Estrategias didácticas para la comprensión de técnicas algorítmicas mediante la aplicación del pensamiento computacional

Luis Fernando Maldonado Osorio

Código: 2012203037

UNIVERSIDAD PEDAGOGÍCA NACIONAL

Facultad de Ciencia y Tecnología

Licenciatura en Electrónica

Bogotá – 2020

Estrategias didácticas para la comprensión de técnicas algorítmicas mediante la aplicación del pensamiento computacional

Luis Fernando Maldonado Osorio

Trabajo de grado para optar al titulo de Licenciado en Electrónica

Directora Trabajo de Grado

Linda Alejandra Leal Urueña

UNIVERSIDAD PEDAGOGÍCA NACIONAL

Facultad de Ciencia y Tecnología

Licenciatura en Electrónica

Bogotá – 2020

Tabla de contenido

Tabla de Figuras

Capítulo I

1 Introducción

Este siglo tiene un componente tecnológico que nunca se había visto en la historia de la humanidad. La sociedad vive en un mundo en el cual la tecnología es transversal a todos los ámbitos de la vida, la necesidad de empoderar a las personas con competencias que les permitan dominar e inventar sus propias tecnologías se hace indispensable.

De ahí que, el pensamiento computacional surja como una posibilidad de cambiar la manera de afrontar situaciones de la vida real, asimismo, ayude a tomar desiciones de forma ordenada, secuenciada, lógica y sin ambiguedades, algo que a veces a las personas les resulta difícil aplicar en el desarrollo habitual de sus actividades cotidianas.

En efecto, comprender la esencia del pensamiento computacional, empodera al sujeto frente a un mundo lleno de situaciones que requieren de metodologías de solución efectivas que estimulen las habilidades lingüisticas y numéricas, asimismo la creatividad y el aprendizaje colaborativo.

Por otro lado, existen enormes desafios educacionales en el presente constante en donde las habilidades y destrezas desarrolladas por el pensamiento computacional dará a los estudiantes de la educación K-12 y universitaria, herramientas que los haran menos indefensos, mejor provistos para efrentar esos desafios como puerta de entrada a ese mundo del futuro ya presente. (Artecona *et al*, 2017).

En consecuencia con lo anterior, se presenta este documento como una herramienta lúdica para el desarrollo de capacidades y destrezas que valdría la pena implementar en todo tipo de situación social con el propósito de optimizar tiempo y recursos que pueden llevar a efecto.

El documento esta dividido en dos capítulos, en el primer capítulo se realiza una aproximación a la definición del pensamiento computacional , teniendo en cuenta el desarrollo y las aportaciones de algunos personajes a este concepto. De igual manera, se describen los pilares que están implícitos en éste y se concluye con la solución de tres ejercicios de juegos mentales que precisan estrategias didácticas que se fundamentan en la aplicación de los pilares del pensamiento computacional.

De manera semejante, en el segundo capítulo se analizan paso a paso las técnicas algorítmicas de búsqueda y ordenación, utilizando material audiovisual público y un lenguaje básico que permite a cualquier persona interesada comprender y aplicar estas técnicas en situaciones del diario vivir.

2 Pensamiento Computacional

Para precisar que es el Pensamiento Computacional se cito la definición de Julian Fraillon, profesor y director del Programa Nacional de Evaluación de Australia, quien menciona que el Pensamiento Computacional son las habilidades para reconocer aspectos de la realidad que pueden ser modelados como problemas incluyendo el diseño y la evaluación de soluciones algoritmicas que pueden ser implementadas computacionalmente. (Csizmadia, Curzon, Dorling, Humphreys, Ng, Selby, Woollard, 2015).

Este pensamiento, entonces no debe considerarse como una forma superadora de todos los métodos disponibles para solucionar problemas sino como una forma complementaria, propia de este tiempo, que se puede utilizar en conjunto con la tecnología, entendiendo la tecnología como una condición social que permite vincular de forma interdisciplinar una serie de conocimientos al servicio de necesidades y resolución de problemas.

La figura 1 corresponde a una foto titulada "*Le velo de Tati*" o en español la bicicleta de Tati, tomada por el fotografo frances Robert Doisneau en 1949, en ella se observa un hombre con el uniforme caracteristico al de los ingenieros de la epoca de la industrialización presente en las dos guerras mundiales, de igualmanera, se puede divisar las partes de una bicicleta colocadas de tal manera, que las piezas que se relacionan de forma directa, es decir, que se unen directamente estaban cerca una de la otra.

El proposito de traer a colación esta foto, es evidenciar como ya desde 1949 y mucho antes, el ser humano a elaborado sus propias estrategias y métodos de resolución, para un problema en especifico como lo es la construcción de una bicicleta, pasando desde la idealización de la funcionalidad de cada pieza en un estilo robusto a un estilo ergonómico. Esto le ha permitido al ser humano llegar a diferentes soluciones, asimismo, desarrollar capacidades relativas al Pensamiento Computacional.

Figura 1. Le Velo de Tati, 1949 - Robert Doisneau (Francia, 1912–1994). [1]

[1] Figura Tomada de la página MutualArt link: https://www.mutualart.com/Artwork/Le-velo-de-Tati/A9571A826C675282 consultada noviembre 11 de 2020.

"Un problema es una situación que ubica a quien los resuelve ante la necesidad de desplegar su actividad cognitiva en un intento de búsqueda de estrategias, de elaboración de conjeturas y toma de desiciones"(Azcue, Diez y Lucanera *et al*, 2002, p. 53)

Las habilidades necesarias no solamente para programar sino para afrontar determinado problema, como se ha hecho notar con la contrucción de la bicicleta, no solo están enfocadas en espacios computacionales para entender cómo funciona una computadora y que es capaz de hacer, sino también en desarrollar herramientas y técnicas para analizar problemas cotidianos diseñando sus posibles soluciones. Estos conocimientos y habilidades a los que se hace referencia son las que se conocen actualmente con el nombre de Pensamiento Computacional. (Zapata-Ros, M. 2015).

2.1 Fundamentos del Pensamiento Computacional

Teniendo en cuenta que el Pensamiento Computacional va más allá de programar o codificar e implica todo un proceso previo de formulación y análisis del problema, como también de diseño y de evaluación de soluciones, la *International Society for Technology in Education*[2] entendiendo el pensamiento computacional como un proceso cognitivo que involucra un razonamiento lógico, permeado por el pensamiento, el

[2] La Sociedad Internacional para la Tecnología en Educación, conocida por su sigla en inglés ISTE., es una asociación de docentes que buscan promover el uso eficaz de la tecnología en la enseñanza y la formación de profesores. Su sede está en Washington, Estados Unidos.

lenguaje y la inteligencia plantea cinco capacidades que a continuación se sintetizan de la siguiente manera:

La primera hace referencia a la capacidad de pensar de forma algorítmica, la segunda se centra en la capacidad de pensar en términos de descomposición, la tercera hace énfasis a la capacidad de pensar en generalizaciones, identificando y haciendo uso de patrones. La cuarta comprende la capacidad de pensar en términos abstractos y la elección de buenas representaciones. (Csizmadia. A, 2015)

Por último, es importante destacar que la quinta abarca la capacidad de pensar en terminos de evaluación.

En consecuencia esas cinco capacidades constituyen los pilares del Pensamiento Computacional.

2.2 Pilares del Pensamiento Computacional

2.2.1 Capacidad de pensar de forma algorítmica

Pensar de forma algorítmica representa una actividad cognitiva asociada a la resolución de problemas, a su especificación y a la comunicación de su solución. Por consiguiente, pensar de esta forma, conlleva a idealizar una solución a través de una definición clara de pasos. Estos pasos generelamente son secuenciales y estan sujetos a reglas.

Existen situaciones en la vida cotidiana en las que se puede pensar de forma algorítmica, como por ejemplo, cuando un cheft escribe una

receta para preparar un plato, éste esta creando un algoritmo dado que, otros pueden seguir los pasos y así reproducir la receta de manera correcta. Por otra parte, cuando una persona anota las instrucciones para llegar a un sitio específico, está detallando una secuencia de pasos (un algoritmo) para que otra persona lo pueda ubicar.

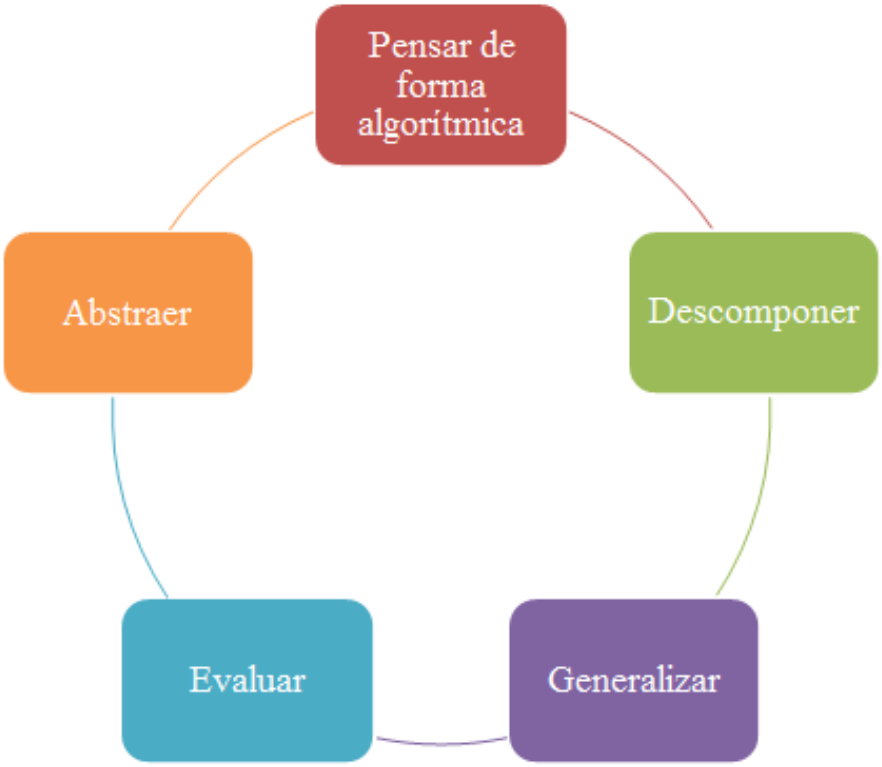


Figura 2. Pilares del Pensamiento Computacional

En el ámbito educativo, el pensamiento algorítmico puede evidenciarce cuando un profesor proporciona un conjunto de instrucciones para desarrollar un experimento en un laboratorio, éste está definiendo un algoritmo, que es seguido por los estudiantes para la obtención de datos para su posterior análisis y aprendizaje.

2.2.2 Capacidad de pensar en términos de descomposición

La descomposición hace referencia a la separación de un problema o cosa en diferentes partes o elementos que lo conforman. Es así como, se facilita entender el conjunto de partes, con la intención de resolver, desarrollar y evaluar por separado el problema.

Esto representa sin lugar a duda, un método para resolver de forma fácil problemas complejos. Puesto que, se comprende mejor el proceso de solución para cada descomposición, obteniendo al final la solución general del problema original, integrada por la suma de sus soluciones específicas.

Un ejemplo de la vida real, en el cual se evidencia el pensar en términos de descomposición es la preparación de la semana cultural en el colegio, se parte de una actividad total, se descompone en actividades, como la difusión del evento, la preparación y ensayo de manifestaciones culturales, la construcción y atención de cada stand, entre otras.

2.2.3 Capacidad de pensar en generalizaciones

Pensar en generalizaciones es pensar en la identificación de patrones, similitudes y conexiones.

En la vida real, al desconponer un problema complejo se suelen encontrar patrones entre los subproblemas.

Los patrones se expresan como características compartidas entre los distintos subproblemas. Cuando se determinan, es posible trabajarlos de manera conjunta y así simplificar el proceso de resolución. De ello resulta necesario decir, que cuando los problemas comparten patrones son más faciles de solucionar, debido a que es posible generalizar soluciones ya diseñadas con anterioridad y aplicarlas a problemas similares.

Por ejemplo: una persona esta a cargo de alimentar los animales de un zoológico, existe comida suficiente y especial para cada uno. Las instrucciones para que la persona alimente a los animales son simples:

- Para alimentar al león, colocar la comida del león en el plato del león.
- Para alimentar al elefante, colocar la comida del elefante en el plato del elefante.
- Para alimentar a la avestruz, colocar la comida de la avestruz en el plato de la avestruz.

Si se analiza cada instrucción se puede afirmar que hay una estructura subyacente común en cada una de éstas. Esta estructura conforma un patrón, que podría expresarse de la siguiente manera:

- Para alimentar al <animal>, colocar la comida del <animal> en el plato del <animal>.

Al encontrar un patrón en las instrucciones que le asignan a la persona del zoológico, se puede realizar un proceso de generalización

simplificando el protocolo de alimentación a partir de una única instrucción genérica.

2.2.4 Capacidad de pensar en términos abstractos

La abstracción es el proceso de hacer un problema más comprensible mediante la reducción de los detalles innecesarios. Este proceso se basa en identificar lo que es importante de algo. Asimismo, desarrollar este tipo de pensamiento da como resultado la construcción de una visión simplificada, que es la idea principal de la asbtracción.

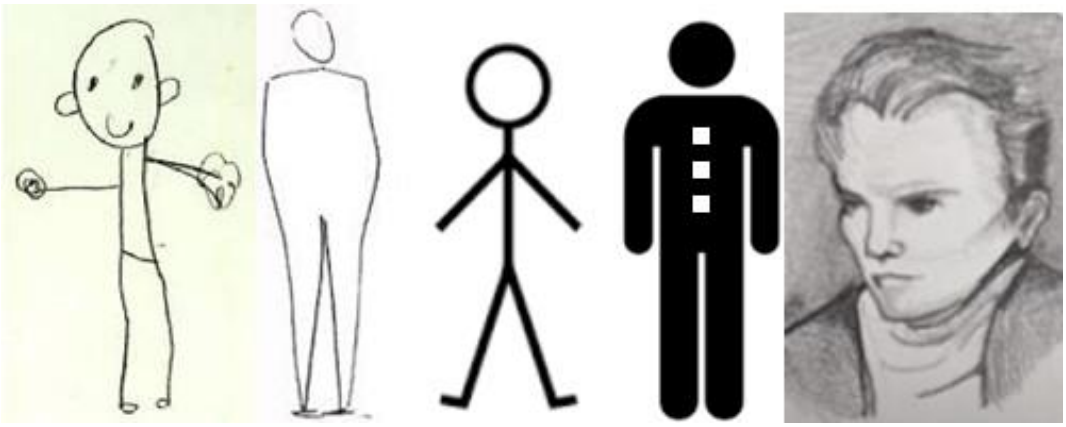


Figura 3. Diferentes representaciones de un ser humano.

El ser humano ha utilizado la abstracción de muchas maneras al retratarse así mismo. Fernando Bordignon y Alejandro Iglesias en su documento titulado "*Introducción al Pensamiento Computacional*" del ministerio de educación argentino[3] ilustran cinco imágenes como se observa en la figura 3.

[3] Bordigon, Fernando. Iglesias Alejandro. *Introducción al pensamiento computacional*. Editorial Educar – UNIPE. Ministerio de Educación Argentina. Año 2020.

Ellos plantean que la primera imagen de izquierda a derecha representa un dibujo de un niño de 4 años, siendo su primer retrato. Después aparece el bosquejo utilizado en planos de arquitectura para dar idea de las dos dimensiones en relación a una persona. La tercera imagen corresponde a un dibujo realizado mediante figuras euclidianas básicas típico que se puede observar en muchos esquemas. En cuarto lugar, se evidencia la imagen de un hombre que habitualmente se utiliza en señalética. Finalmente, se visualiza la imagen de una caricatura que tiene detalles artísticos a fin de comunicar mejor las expresiones del personaje. Estas cinco abstracciones son representaciones mentales de un ser humano.

Figura 4. Mapa troncal Transmilenio

Otro ejemplo de abstracción es el mapa con las diferentes troncales de transmilenio, la característica de este objeto gráfico está dada porque solo muestra detalles necesarios, ya que si es muy detallado puede resultar confuso y si es muy simplificado puede obviar información útil.

Es así como: “la esencia de la abstracción consiste en singularizar un rasgo que, en contraste con otras propiedades, se considera especialmente relevante” (Salvador, 2016).

2.2.5 Capacidad de pensar en términos de evaluación

La evaluación es una experiencia más de aprendizaje y se entiende como un proceso sistemático, continúo y permanente de recolección y análisis de información, esto permite a la persona una vez que ha diseñado una solución analizarla, con la intención de dar respuesta a las tres preguntas que se plantean en la figura 5 y de esta manera lograr que el algoritmo se entienda fácilmente, asimismo, sea eficaz y eficiente, es decir resuelva el problema haciendo el mejor uso posible de los recursos disponibles y por último que cumpla con los criterios de diseño especificados.

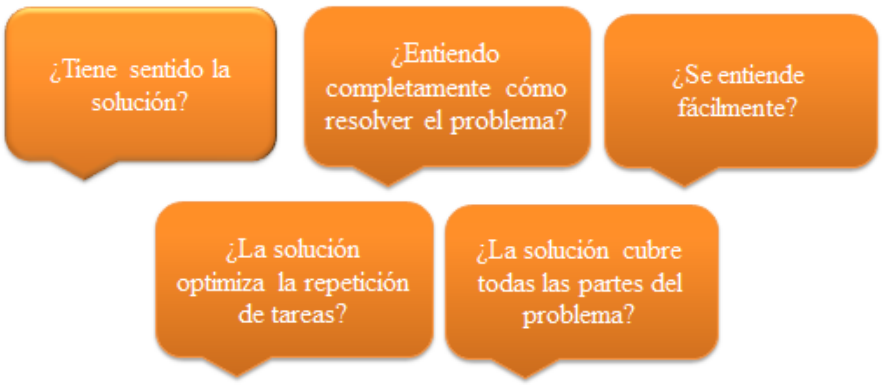


Figura 5. Preguntas para evaluar una solución

Luego de que un algoritmo es evaluado y cumple satisfactoriamente con las preguntas anteriores , se entiende que es una solución correcta, por lo tanto, se puede continuar con la etapa de programación. Es recomendable antes de programar en cualquier lenguaje idealizar,

construir y evaluar el algoritmo, con el fin de eliminar los errores que conllevan a mayores costos y tiempos en el proyecto.

3 Análisis y solución de los ejercicios propuestos

A continuación, se plantean, analizan y solucionan algunos ejercicios que se desarrollaron en el seminario pensamiento comptacional, esto con el objetivo de relacionar la teoría con la práctica. De igual manera, evidenciar como se aplican los pilares del pensamiento computacional en la solución de cada ejercicio.

Los ejercicios componen un recurso didáctico introductorio a conceptos relacionados con el pensamiento computacional. Cada ejercicio permitirá construir, evaluar y comunicar los conocimientos a los que se ha llegado.

3.1 Sopa de letras

Inicialmente se menciona el orden de los pasos que se implementan para resolver la sopa de letras. Por otro lado se plantea la relación entre el desarrollo implementado y los pilares del pensamiento computacional y por último se describe los pasos del algoritmo.

A continuación, se describe detalladamente la secuencia de pasos que se siguió hasta llegar a la solución.

Primero se observó y se leyó el nombre de cada personaje.

Enseguida, se organizaron los nombres de cada personaje alfabéticamente.

El método implementado consiste en desplazar la mirada por cada columna de la sopa de letras, desde la primera casilla correspondiente a la letra T hasta la última casilla correspondiente a la letra P, de tal manera, que el movimiento visual describa la forma del cuerpo de una serpiente.

De modo que con el movimiento visual se ubica la letra inicial del nombre, luego se observa si arriba o abajo, a la derecha o a la izquierda o en sus diagonales se encuentra la segunda letra del nombre.

Se determinó inicialmente que algunos nombres comienzan con la misma letra, es el caso de Ada, Anita y Alan. Esta situación conllevo a que en la primera búsqueda se tuviera que encontrar tres personajes a la vez.

Para facilitar este proceso se reconoció las dos primeras letras de los tres nombres. Es decir, Ad – An – Al, con el fin de aprovechar el método idealizado de manera eficiente y eficaz.

En resumen, fue de esta manera como se encontraron los nombres de los personajes que estaban implícitos en la sopa de letras y se determinó de manera decisiva los que no estaban.

Para la solución de este problema se aplicaron de la siguiente forma los cuatro pilares del pensamiento computacional.

El pilar construcción de algoritmos, abarco los pilares de descomposición, generalización y abstracción. A continuación se explica de qué forma.

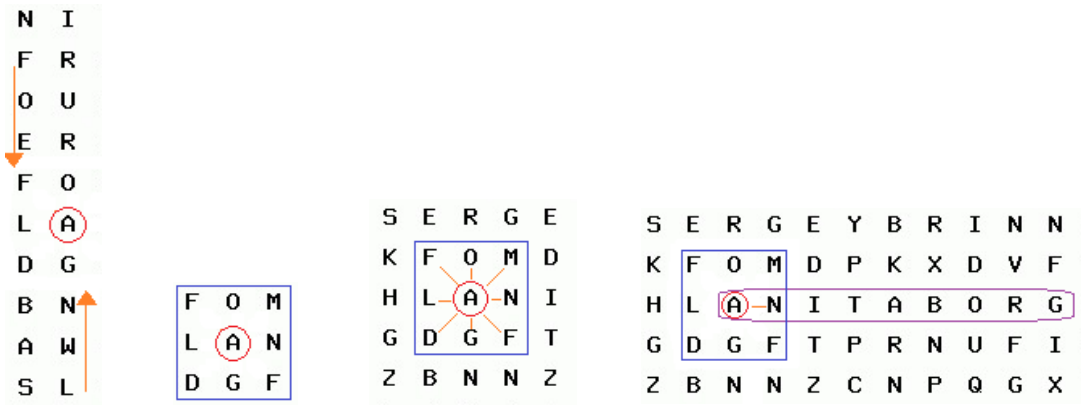
El pilar construcción de algoritmos, se evidencia por medio de la secuencia que se llevó a cabo para el desarrollo del ejercicio, iniciando con la observación, la lectura de cada nombre y la organización alfabética de éstos.

Luego, la búsqueda de la inicial de cada nombre con el propósito de determinar si en los extremos próximos de la letra se encontraba la segunda letra, a fin de formar el nombre del personaje.

Por otra parte, la descomposición se realizó cuando se dividió la sopa de letras en columnas de forma tal, que se observara minuciosamente la inicial del nombre buscado, con la intención de encontrar o descartar la posibilidad según las letras que se encontraban a su alrededor.

La generalización está presente en este ejercicio, cuando la solución planteada puede ser adaptada a la búsqueda de palabras en cualquier sopa de letras.

Por medio del proceso de abstracción se redujo la complejidad, dejando a un lado el movimiento visual aleatorio por el movimiento visual estructurado, dado que éste garantiza encontrar la palabra en un tiempo mínimo, en consecuencia, se filtra únicamente las dos primeras letras de la palabra que se pretende buscar.



a. *b.* *c.* *d.*

Figura 6. Se observa la secuencia del algoritmo que se implemento para encontrar los nombres de cada personaje. La imagen *a* representa el movimiento visual que se realizo hasta llegar a una letra A. Una vez identificada la letra A, se delimito la zona por medio de un cuadrado con el fin de relacionar la primera y segunda letra con el nombre completo. La imagen *c* evidencia las ocho posibles relaciones que existen para obtener el nombre, de las cuales una correspondiente a la segunda letra del nombre ANITABORG. Por último se selecciona por medio de una elipse la opción que forma el nombre.

Algoritmo

1. Organizar las palabras alfabéticamente.
2. Agrupar las palabras que inician con la misma letra.
3. Seleccionar la palabra o palabras a buscar (Palabras que empiezan con la misma letra inicial).
4. Identificar y memorizar la segunda letra de la o las palabras que inician con la misma letra.
5. Iniciar la búsqueda visual de la palabra desde cualquier casilla ubicada en los vértices del cuadrado.
6. Recorrer cada columna o fila en forma de serpiente.
7. Encontrar una letra equivalente a la letra inicial de la palabra que se está buscando.

8. Comparar las letras que se encuentran alrededor de la letra equivalente, encontrando similitud con la palabra buscada.
9. Si la palabra encontrada es diferente a la buscada, ejecuta el inciso 6.
10. Si la palabra encontrada es equivalente a la buscada, revisa si es la última.
11. No es la última palabra, ejecuta el inciso 3.
12. Si es la última palabra, fin del ejercicio.

~~Ada Lovelace~~, ~~Anita Borg~~, ~~Barbara Liskov~~, ~~Dana Ulery~~, ~~Dorothy Denning~~, ~~Fran Allen~~, ~~Grace Hopper~~, ~~Jeannette Wing~~, Alan Turing, ~~Chris Strachey~~, Edgar Codd, ~~Edsger Dijkstra~~, John Von Neumann, Maurice Wilkes, ~~Niklaus Wirth~~, Philip Emeagwali, ~~Sergey Brin~~, Tim Berners Lee, Tony Hoare, ~~Vint Cerf~~

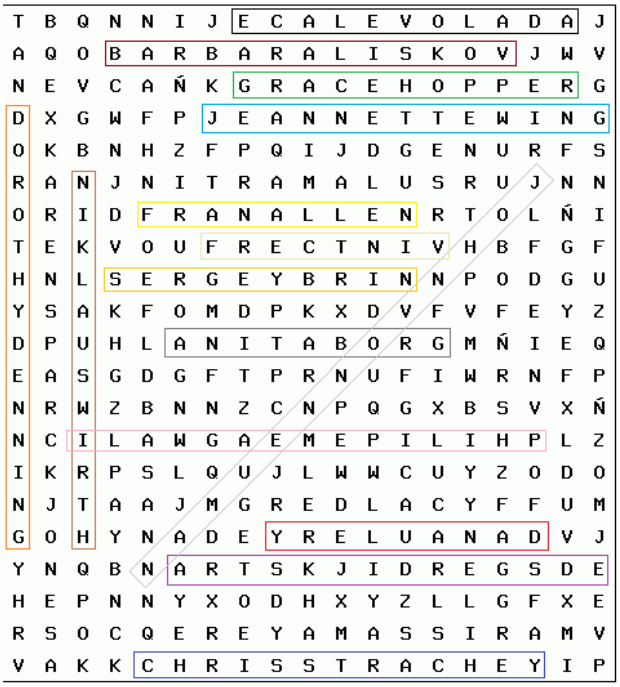


Figura 7. Solución de la sopa de letras. Implementando el algoritmo propuesto se descarto cuatro personajes que no se encuentran en la sopa de letras.

3.2 Bloques Cortados

A continuación se describe el desarrollo sistemático de uno de los tres rompecabezas.

En el primer apartado se encuentra el enunciado correspondiente al ejercicio. Luego, se hace énfasis en algunas claves para la solución del problema. Enseguida, se relaciona el procedimiento implementado con los pilares del pensamiento computacional. Finalmente, se detalla el algoritmo. Cabe señalar que el algoritmo propuesto se puede aplicar a cualquier ejercicio del mismo tipo.

Indicaciones

Hay dos reglas que debe cumplir un rompecabezas de bloques cortados.

Cada área demarcada por las líneas más oscuras debe contener los números del 1 al número de cuadrados en el área. Por ejemplo, el área superior en el primer rompecabezas consta de 5 cuadrados, por lo que esos cuadrados deben llenarse con los números: 1, 2, 3, 4 y 5, sin números repetidos. Si el área tiene dos cuadrados, como el de abajo a la izquierda, del mismo rompecabezas, debe llenarse con los números 1 y 2.

Ningún número puede estar al lado del mismo número en cualquier dirección, ya sea horizontal, vertical o diagonal. Entonces, en la cuadrícula de abajo, el hecho de que haya un 4 en el costado significa que no puede haber un 4 en ninguno de los 5 cuadrados que lo rodean.

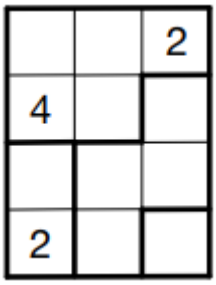


Figura 8. Rompecabezas simple

A continuación, se describe detalladamente la secuencia de pasos que se siguió hasta llegar a la solución.

Primero se identificó las áreas demarcadas, con el objetivo de conocer la serie de números enteros desde *1* hasta *n*, donde n representa el número de casillas que contiene cada área demarcada.

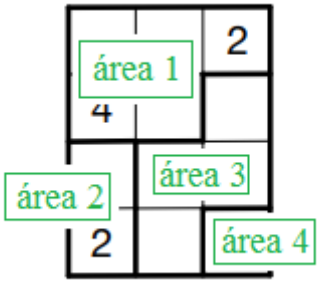


Figura 9. Se observa las cuatro áreas del rompecabezas simple.

Para el área 1 se necesita ubicar los números del *1* al *5*, para el área 2 del *1* al *2*, al igual que para el área 3 del *1* al *5*.

Segundo, se observó si había áreas demarcadas con una sola casilla con el fin de colocar el número 1(Véase imagen a, figura 10).

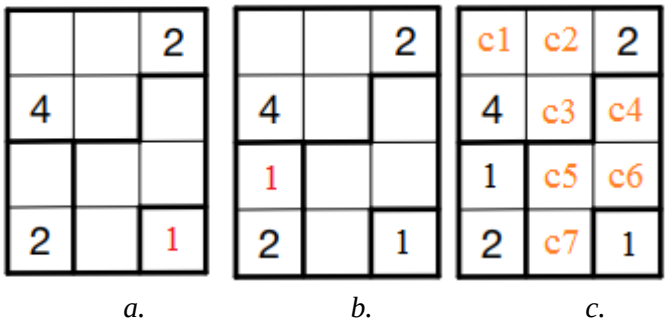


Figura 10. Descripción visual del metodo que se implemento paso a paso para determinar la posición correcta de cada número.

En la imagen b (figura 10) se observa que en el área 2 solo fue necesario ubicar el número *1*. Hasta este punto fue muy fácil colocar los dos números, debido a que esa era la única opción asumiendo las condiciones planteadas en el enunciado.

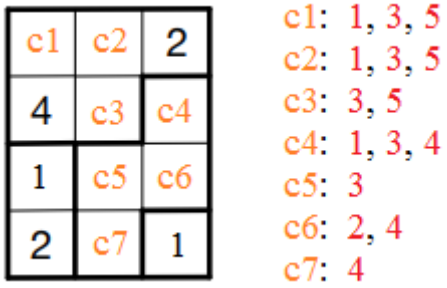


Figura 11. Se muestra los posibles numeros que puede contener cada casilla, teniendo en cuenta las condiciones mencionadas en el encunciado.

Tercero, a cada casilla en blanco se le asigno un nombre con la intención de conocer cuales eran los posibles números que podrian ir en éstas. En la figura 11, se observa el nombre de cada casilla con sus correspondientes números. Los números se obtuvieron al aplicar la rectricción de que ningún número, puede ser rodeado por su número equivalente. Asimismo, fue necesario realacionar la restricción antes mencionada, con la serie de números enteros que debe contener cada área.

Por ejemplo:

La casilla c5 no puede contener los números 1, 2 o 4 porque no cumpliría con la rectricción de que un número tenga su número equivalente en las diagonales y a sus lados, del mismo modo si se relaciona con el área determinada el único número que hace falta para completar la serie y que cumple con las condicones seria el número 3.

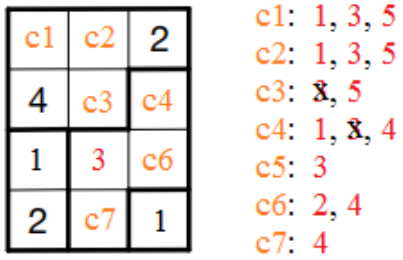


Figura 12. Se ubica en la casilla c5 el número 3

Con la ubicación del número 3 en la casilla c5 se eliminó la posibilidad de que en la casilla c3 y c4 estuviera el número 3. De ahí que en la casilla c3 esté el número 5. Una vez que se ubica en la casilla c7 el número 4, los posibles números van disminuyendo facilitando la correcta ubicación de los demás.

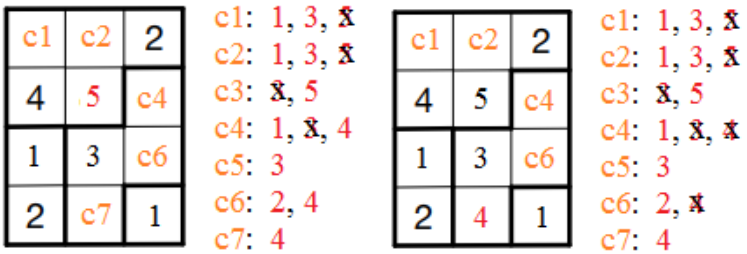


Figura 13. En la imagen *a* se ubica el número 5 en la casilla c3, al igual que en la casilla c7 se ubica el número 4.

Si se observa la imagen b (figura 13), se evidencia que la casilla c4 y c6 deben contener respectivamente los números 1 y 2. Después de lo cual la casilla c2 no podría contener el número 1, porque en la diagonal estaría la casilla c4 con este número. No obstante, la solución correcta para la casilla c2 es el número 3. Por último la casilla c1 debe contener el número 1 para cumplir con los números del área determinada.

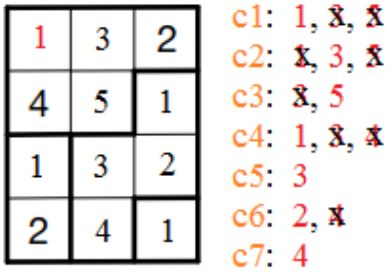


Figura 14. Solución rompecabezas simple.

Para la solución de este ejercicio se aplicaron cuatro pilares del pensamiento computacional.

El pilar construcción de algoritmos, se evidencia por medio de la secuencia que se llevó a cabo para el desarrollo de los rompecabezas, iniciando con la identificación de las áreas demarcadas y la serie de números enteros que contiene cada una. Si existen áreas con una o dos casillas, se completa con un número que cumpla con las condiciones.

Después se listan los posibles números que puede contener cada casilla. Por último, se crean relaciones teniendo en cuenta las restricciones, dado que es necesario, minimizar los posibles números de cada casilla incógnita.

Por otra parte, la descomposición se realizó cuando se dividió el rompecabezas por áreas demarcadas de forma tal, que permitiera observar minuciosamente qué números venían a priori ubicados en las casillas y así

poder determinar por medio de las relaciones creadas, los números que faltan por ubicar con un nivel de dificultad mucho menor.

La generalización está presente en este problema, cuando la solución planteada puede ser adaptada a la búsqueda de los números de cualquier rompecabezas.

Por medio del proceso de abstracción se redujo la complejidad, identificando las relaciones entre casillas, dejando a un lado la suposición cándida de cualquier número correspondiente a una casilla.

Algoritmo

1. Identificar las áreas demarcadas
2. Determinar si hay áreas compuestas únicamente por una casilla. Si es así colocar el número 1 dentro de ésta.
3. Determinar si hay áreas compuestas únicamente por dos casillas y dentro de cualquiera de las dos el número 1 o el número 2. Si es así colocar el número que falta.
4. Dar un nombre a cada casilla en blanco.
5. Realizar una lista con los nombres de las casillas que colindan con los números que vienen ubicados a priori, junto a los posibles números que pueden ir dentro de éstas.
6. Crear relaciones entre el área demarcada y la restricción, de modo que ningún número este rodeado por su número equivalente.
7. Ubicar el número obtenido por medio de las relaciones creadas dentro de la casilla correspondiente.

8. Revisar la lista y descartar los números que no cumplen con las restricciones.
9. Repetir el inciso 6 y 7 hasta completar el rompecabezas.

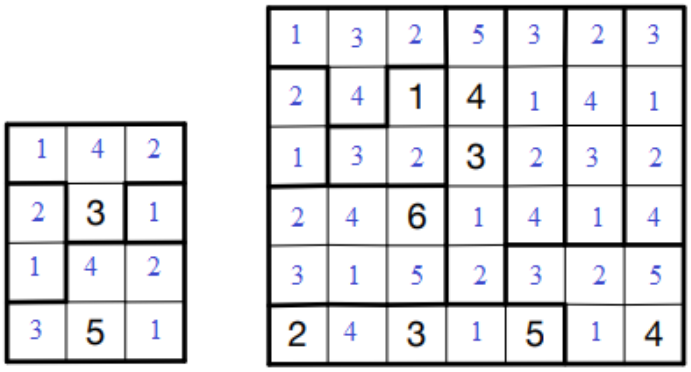


Figura 15. Solución de un rompecabezas simple y uno complicado solucionado con el algoritmo propuesto.

3.3 Cuadrícula

De igual manera que en el ejercicio anterior, en este ejercicio se describen los pasos que se siguieron para solucionar la cuadrícula. Asimismo, se evidencia la relación que hay entre los pilares y el proceso realizado para concluir con el algoritmo que se propone al final.

La siguiente es una cuadrícula similar a un crucigrama donde cada cuadrado debe completarse con un dígito del 1 al 9. Cada bloque de dígitos horizontal o vertical debe sumar el número dado a la izquierda o arriba, respectivamente. Todos los dígitos en cada bloque deben ser diferentes.

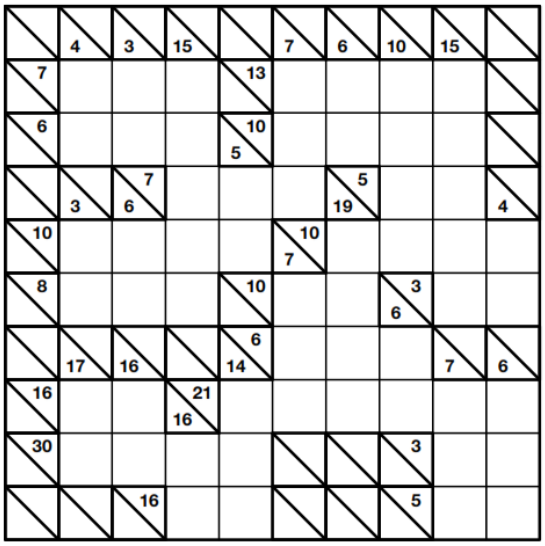


Figura 16. Ejercicio cudrícula para solucionar.

A continuación, se ilustra el desarrollo sistemático de la cuadricula. Cabe señalar que el algoritmo propuesto se puede aplicar a cualquier ejercicio del mismo tipo.

Para solucionar el ejercicio de la cuadricula se llevaron a cabo los siguientes pasos.

1. Se leyó el enunciado con el fin de identificar cuáles son las condiciones o restricciones que se debe tener en cuenta para solucionar el ejercicio. Por ejemplo:

- Cada cuadrado debe completarse con un dígito del 1 al 9
- Cada bloque de dígitos horizontal o vertical debe sumar el número dado a la izquierda o arriba.
- Todos los dígitos en cada bloque deben ser diferentes.

2. Se encerró con un círculo rojo los números horizontales y con un círculo azul los números verticales, con intención de ordenarlos de menor a mayor.

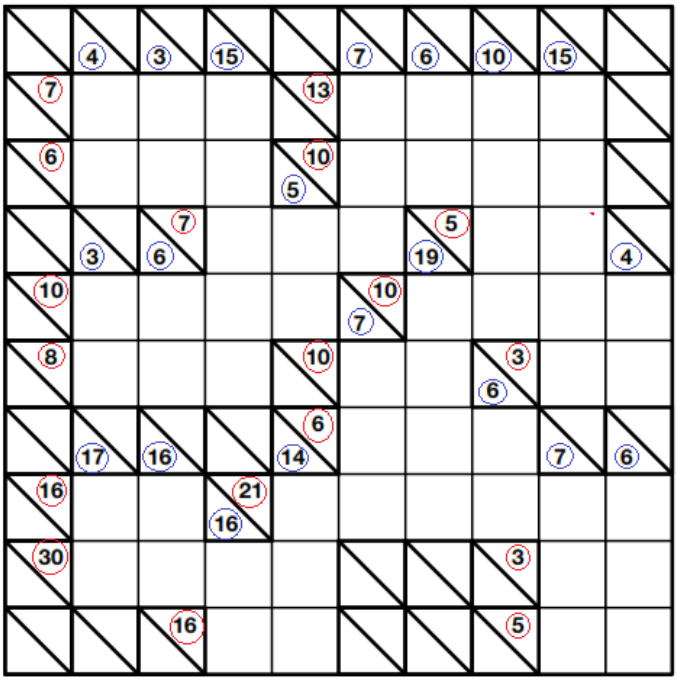


Figura 17. Se observan los números de la cuadrícula encerrados con un círculo rojo (horizontal) y azul (vertical).

3. Se realizó una lista con todos los números y sus posibles combinaciones, teniendo en cuenta las condiciones que se mencionan en el inciso 1.

Números Horizontales		Números Verticales	
3 = 1 + 2	(2 casillas)	3 = 1 + 2	(2 casillas)
5 = 4 + 1	(2 casillas)	4 = 1 + 3	(2 casillas)
3 + 2		5 = 4 + 1	(2 casillas)
		3 + 2	
		6 = 1 + 5	(2 casillas)
		2 + 4	
6 = 1 + 2 + 3	(3 casillas)		
		6 = 1 + 2 + 3	(3 casillas)
7 = 1 + 2 + 4	(3 casillas)	7 = 1 + 2 + 4	(3 casillas)
8 = 1 + 2 + 5	(3 casillas)	10 = 1 + 2 + 3 + 4	(4 casillas)
1 + 3 + 4			
10 = 1 + 2 + 3 + 4	(4 casillas)	14 = 1 + 6 + 7	(3 casillas)
		1 + 4 + 9	
10 = 1 + 9	(2 casillas)	2 + 5 + 7	
2 + 8		2 + 4 + 8	
3 + 7		3 + 5 + 6	
4 + 6		3 + 4 + 7	
13 = 1 + 3 + 4 + 5	(4 casillas)	15 = 1 + 2 + 3 + 4 + 5	(5 casillas)
1 + 2 + 4 + 6		16 = 7 + 9	(2 casillas)
1 + 2 + 3 + 7		17 = 8 + 9	(2 casillas)
16 = 7 + 9	(2 casillas)	19 = 1 + 2 + 7 + 9	(4 casillas)
21 = 1 + 2 + 3 + 4 + 5 + 6	(6 casillas)	2 + 3 + 6 + 8	
		1 + 3 + 6 + 9	
		3 + 4 + 5 + 7	

Figura 18. Se observan los números de la cuadrícula con las posibles combinaciones que se pueden utilizar dependiendo las casillas que conforman cada bloque.

4. Conociendo las posibles combinaciones de cada número establecido en la cuadrícula, se plantearon relaciones de intersección de manera que al sumar los números se obtuviera el resultado de arriba y de la izquierda.

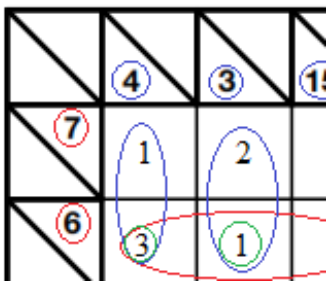


Figura 19. Se muestra un ejemplo de tres relaciones creadas que al sumar los números de cada bloque dieran como resultado el número de la casilla de arriba y el número de la casilla de la izquierda.

Para la solución de este ejercicio se aplicaron cuatro pilares del pensamiento computacional.

El pilar construcción de algoritmos, se evidencia por medio de la secuencia que se llevó a cabo para el desarrollo de la cuadricula, iniciando con la identificación de las condiciones o restricciones en el enunciado, a fin de encerrar en un círculo rojo o azul los números dependiendo la posición horizontal o vertical en la que se encuentren.

Del mismo modo se determinó las posibles combinaciones que daban como resultado el número preestablecido por la casilla de arriba o la casilla de la izquierda

Por otra parte, la descomposición se realizó cuando se dividió los números dependiendo de su posición horizontal o vertical de manera tal, que se precisara qué relaciones se tenían que formar para obtener un resultado equivalente al de la casilla.

La generalización está presente en este problema, cuando la solución planteada puede ser adaptada a la búsqueda de los números que sumados vertical u horizontalmente den como resultado el número presente en la casilla de arriba o en la de la izquierda.

Por medio del proceso de abstracción se redujo la complejidad, identificando las relaciones entre casillas de manera más precisa, de esta manera se minimizó las posibles combinaciones hasta quedar con la combinación que cumpliera con las condiciones mencionadas en el enunciado del problema.

Algoritmo

1. Identificar las condiciones o restricciones del problema.
2. Encerrar con un círculo rojo los números horizontales y con un círculo azul los números verticales.
3. Realizar una lista con cada número vertical u horizontal.
4. Al lado de cada número colocar las posibles combinaciones que sumados den como resultado el número de la casilla de arriba y el número de la casilla de abajo.
5. Analizar la intersección que se genera cuando se intercepta la combinación horizontal con la vertical, de manera tal, que la sumatoria del bloque cumpla las condiciones del enunciado.
6. Rellenar las casillas con la combinación obtenida en el inciso anterior.
7. Repetir el inciso 5 y 6 hasta completar las casillas en blanco de la cuadrícula.

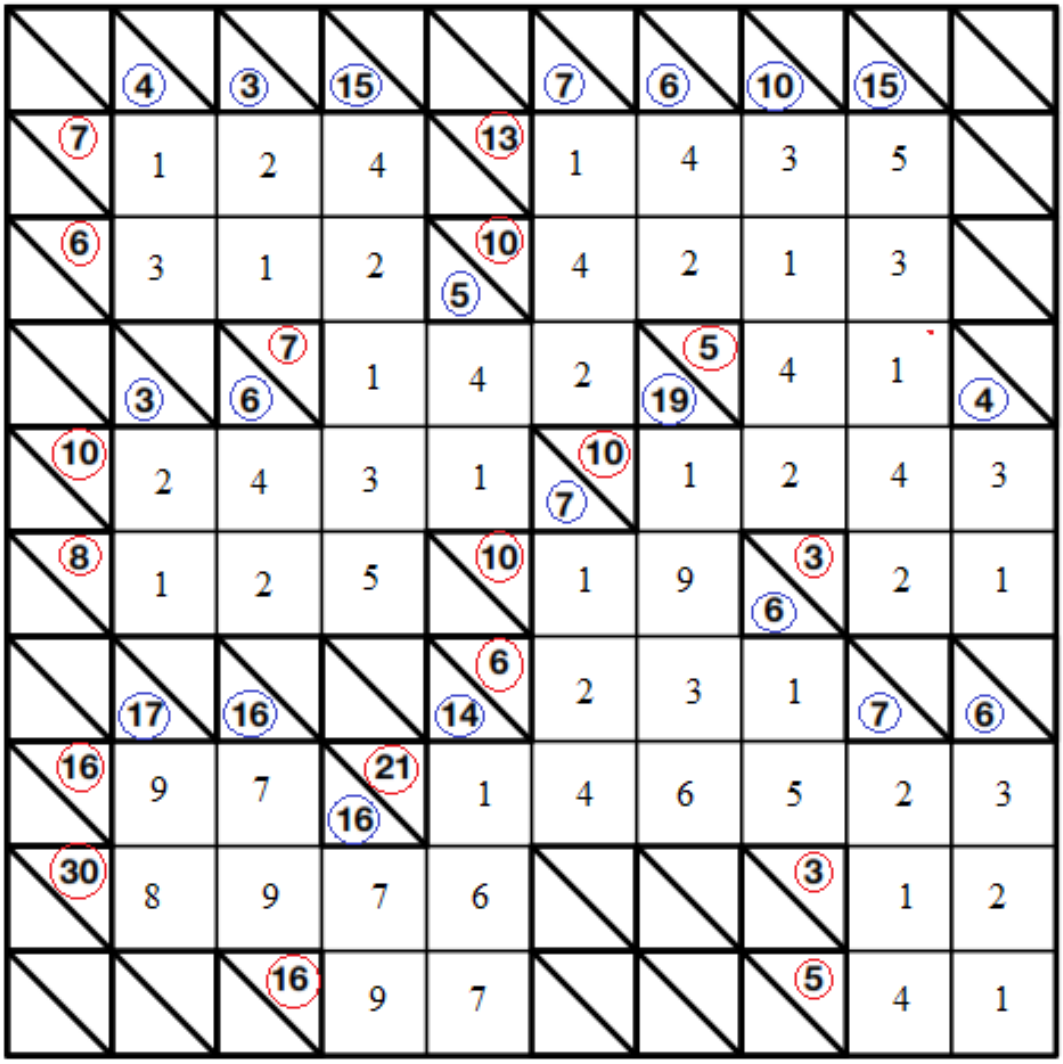


Figura 20. Solución de la cuadrícula implementando el algoritmo propuesto.

Capítulo II

4 Técnicas algorítmicas para la solución de problemas

Antes de desarrollar algunos ejemplos que instruyen la aplicación de las técnicas algorítmicas más relevantes para la solución de problemas, es indispensable realizar una aproximación a la definición del pensamiento algorítmico, dado que éste, constituye una habilidad fuertemente vinculada con la capacidad de abstracción , descomposición y generalización , siendo estos últimos pilares del pensamiento computacional.

El pensamiento algorítmico es un proceso intelectual complejo que optimiza la resolución de problemas, mediante la creación de una serie de pasos lógicos sistemáticos. Este proceso no está resolviendo una respuesta específica, en su lugar, resuelve cómo construir una colección de instrucciones ejecutables precisas, cuya ejecución paso a paso conduce a una meta predefinida en un número finito de pasos replicables.

Las técnicas algorítmicas son empleadas en la solucion de problemas que requieran ordenar o buscar un objeto u objetos dentro de una conjunto. Éstas se basan en la selección de criterios como: la primera letra que compone el nombre o la ponderación que adopta el objeto cuando se le asigna un número.

Estas técnicas estan conformadas por algoritmos de búsqueda y de ordenación.

4.1 Algoritmos de búsqueda

Los algoritmos de búsqueda se encuentran implícitos en situaciones que vivenciamos diariamente. Uno de los primeros algoritmos que se enseña a los estudiantes de primaria, es el algoritmo de división, por medio de la aplicación de éste, se encuentra el cociente. El algoritmo de división permite no solo a las personas sino también a los computadores, resolver problemas con un conjunto sistemático de pasos lógicos.

Por otro lado, se encuentra Google, un motor de búsqueda basado en parte por el algoritmo de PageRank el cual ordena un conjunto de resultados, de modo que la mayoria de las veces, el resultado que se busca está en la parte superior de la página principal.

Existen diferentes algoritmos de búsqueda, la selección de alguno de éstos, depende de factores como lo son la cantidad, la organización y la disposición de los objetos en su conjunto o listado. Asimismo, depende de si los objetos a ordenar pueden ser accedidos de modo aleatorio o deben ser accedidos de modo secuencial.

4.1.1 Búsqueda Lineal

El algoritmo de búsqueda lineal, es el algoritmo más sencillo, se fundamenta en la comparación entre dos objetos adyacentes hasta encontrar el elemento deseado. Si el conjunto contiene bastantes objetos este algoritmo no es eficiente.

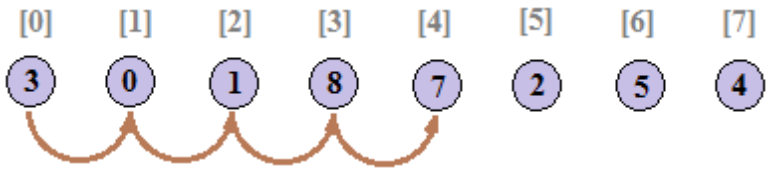


Figura 21. Algoritmo de búsqueda lineal aplicado para encontrar el número 7 en un arreglo de ocho posiciones.

En la figura 21 se observa un ejemplo en el cual, se aplica el algoritmo de búsqueda lineal, con el fin de encontrar el número 7, en consecuencia, con este algoritmo se ejecutan nueve pasos para hallar el número (pregunta si es el número 7 y salta si no lo es). Conviene destacar que si el arreglo de números fuera más grande y el número 7 estuviera en una posición más lejana la eficiencia de este algoritmo seria ínfima por esta razón no es muy utilizado.

4.1.2 Búsqueda Aleatoria

Con el algoritmo de búsqueda aleatoria se encuentra un objeto en una lista sin necesidad de interactuar con todos los objetos de ésta. Por lo general, se suele implementar cuando la lista contiene bastantes elementos. Su eficiencia con respecto al algoritmo de búsqueda lineal es mayor en algunos casos.

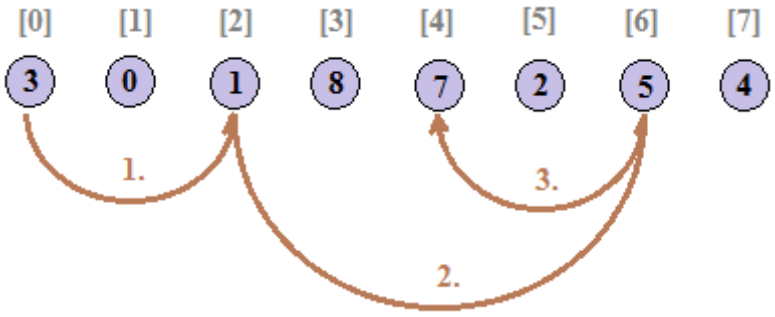


Figura 22. Algoritmo de búsqueda aleatoria aplicado para encontrar el número 7 en un arreglo de ocho posiciones.

Para encontrar el número 7 utilizando el algoritmo de búsqueda aleatoria fue necesario ejecutar tres pasos como evidencia la figura 22. No obstante, la naturaleza de este algoritmo es aleatoria, no se conoce exactamente cuantos pasos tarda en encontrar *x* número en un arreglo de *n* elementos.

4.1.3 Búsqueda Binaria

El algoritmo de búsqueda binaria se basa en la ejecución de dos pasos.

En el primer paso, se selecciona el objeto que está en la mitad de la lista total de objetos. En el segundo paso, se compara el objeto buscado con el objeto que está en la mitad, si el objeto buscado hace parte de la cantidad de objetos que están antes que el objeto de la mitad, se eliminan los objetos que están ubicados después de éste. No obstante, si el objeto buscado hace parte de la cantidad de objetos que están después que el objeto de la mitad, se eliminan los objetos que están ubicados antes de éste. Luego se elige de nuevo el objeto que ahora se encuentra ubicado en el la mitad de la cantidad no eliminada en el paso anterior y se compara nuevamente con el objeto buscado, con la intención de ir descartando la cantidad de objetos que

se encuentran en la parte izquierda o derecha del objeto de la mitad. Los pasos anteriores se repiten sucesivamente. Este algoritmo es muy eficiente.

Con el propósito de comprender el algoritmo de búsqueda lineal se idealiza el siguiente ejercicio: existen dos jugadores, el participante A y B, la función del participante B es pensar un número entre 0 y 127, de ahí que la función del participante A sea encontrar este número.

Figura 23. Intervalo cerrado desde el número 0 hasta el número 127.

En primer lugar, se asume que el número 7 es el número que pensó el participante B.

Enseguida se aplica el algoritmo de búsqueda de modo que el participante A se ubica en el medio de los 128 números, es decir, en el número 64, donde el límite izquierdo es 0 y el límite derecho es 127.

Figura 24. Primer intervalo de números con los límites laterales y el elemento central.

El participante A le pregunta al participante B ¿el número es 64?, el participante B diría no, entonces como método para reducir las posibilidades e ir minimizando el intervalo, el participante A le pregunta al participante B: ¿el número a adivinar es menor o mayor al número 64?, el participante B diría es menor a 64.

En ese instante el intervalo cerrado se minimiza desde el número 0 hasta el número 63.

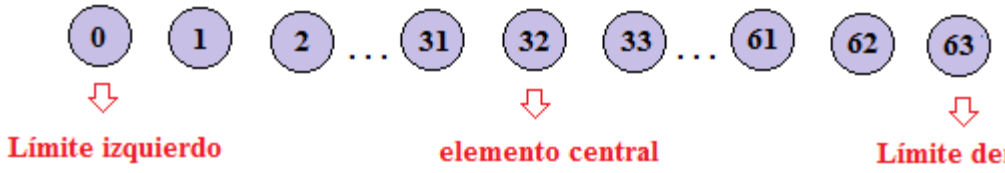


Figura 25. Segundo intervalo de números con los límites laterales y el elemento central.

Después de eso, el participante A se ubica en el número 32, es decir, en el medio del nuevo intervalo. Luego, le pregunta al participante B, ¿el número a adivinar es 32?, el participante B diría no. Teniendo en cuenta lo anterior el participante A le pregunta al participante B ¿el número a adivinar es menor o mayor a 32?, éste responde, es menor a 32. Siendo así como el intervalo cerrado pasa a contener el número 0 hasta el número 31.

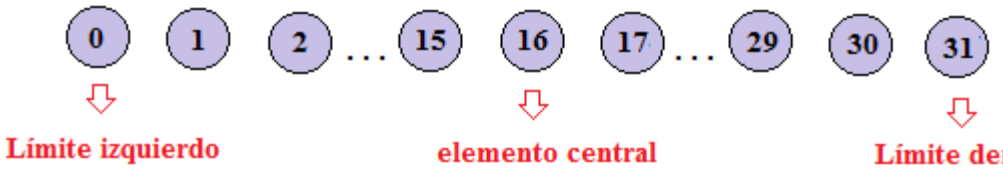


Figura 26. Tercer intervalo de números con los límites laterales y el elemento central.

Acto seguido, el participante A se sitúa en el número 16 y le pregunta al participante B, ¿el número a adivinar es 16?, el participante contesta: el número no es 16, el participante A menciona ¿el número es menor o mayor a 16?, el participante B responde, es menor, esto conlleva a descartar 16 posibilidades obteniendo un intervalo cerrado desde 0 hasta 15.

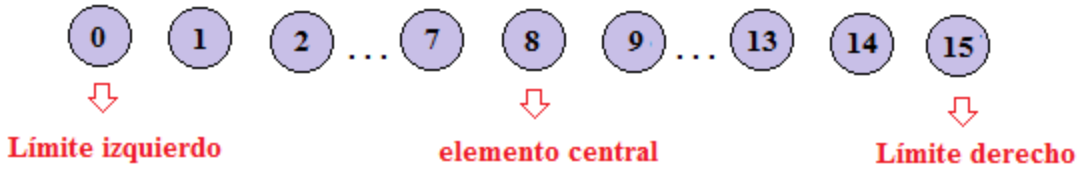


Figura 27. Cuarto intervalo comprendido entre 0 y 15.

El participante A le pregunta al participante B ¿el número a adivinar es 8?, éste responde: no es 8, seguidamente el participante A le pregunta al participante B si el número a adivinar es menor o mayor que 8, el participante B responde que es menor al número 8, en consecuencia vuelve y se reduce el intervalo cerrado conteniendo los números desde 0 hasta 7.

Figura 28. Quinto intervalo, cuando el número total de elementos en un intervalo es par se puede elegir cualquiera de los dos números enteros que acotan el centro. Para esta situación se eligió el número 3.

De inmediato el participante A pregunta al participante B ¿el número a adivinar es 3?, dando como repuesta el participante B que 3 no es el número a adivinar, una vez que el participante A sabe que el número 3 no es el número a adivinar y que el número es mayor que 3, el intervalo cerrado se minimiza a los números 4, 5, 6 y 7. Independientemente escoja el número 5 o 6, el participante A pregunta si es alguno de esos dos números, la respuesta del participante B será que no. Esto conllevará a que el participante A escoja el intervalo cerrado conformado por los números 6 y 7. Cualquiera que sea la selección de los dos números posibles el participante A sabrá que el número a adivinar es 7.

Como se evidencia anteriormente el participante A tuvo que preguntarle seis veces al participante B si era correcto el número que éste seleccionaba para llegar a la respuesta. El algoritmo de búsqueda binaria es un método de búsqueda que minimiza intervalos repetidamente hasta encontrar el número de manera rápida.

A diferencia del algoritmo de búsqueda lineal, el algoritmo de búsqueda binaria es más eficiente, puesto que, el número de instrucciones u operaciones elementales es menor a la cantidad que requiere ejecutar el otro algoritmo para encontrar un número en un intervalo de números ordenados.

Número de elementos examinados		
Tamaño del arreglo	**Algoritmo de búsqueda binaria**	**Algoritmo de búsqueda secuencial**
1	1	1
10	4	10
100	7	100
1000	11	1000
10000	14	10000
100000	18	100000
1000000	21	1000000

Figura 29. En la tabla se compara la eficiencia de la búsqueda binaria con la búsqueda lineal, respecto al número de ejecuaciones en el peor de los casos. De igual manera, se comprueba como para conjuntos de datos grandes la búsqueda binaria sigue siendo eficiente mientras que la lineal se va degradando.

Los datos de la tabla se obtuvieron al dividir cada intervalo en aproximadamente la mitad teniendo en cuenta el límite inferior y superior. Es necesario resaltar, que si el intervalo no es impar, se puede elegir alguno de los números enteros que estén acotando la mitad de ese intervalo. Igualmente se puede escoger cualquier intervalo de la parte izquierda o de la derecha, dado que solo es necesario conocer el número total de intervalos que se necesitan reducir para encontrar el número.

Además, se idealizo que si se realizaban las divisiones con los números en base 10 no iba a ser muy clara la respuesta. Respecto a lo

anterior, se iba convirtiendo los números a base 2 (binarios) a medida que se iba dividiendo por la mitad , hasta llegar a 1, el 1 significa que se ha encontrado el número. Finalmente, se obtiene el total de divisiones necesarias para adivinar el número.

4.2 Algoritmos de ordenación

Los algoritmos de ordenación permiten como su nombre lo dice, organizar un conjunto de datos en algún orden o secuencia específica, tal como creciente o decreciente para datos numéricos o el orden alfabético para datos compuestos por caracteres. De igual manera los algoritmos de ordenación permutan los elementos del conjunto de datos hasta conseguir dicho orden. Para ello se basan en dos operaciones básicas: la comparación y el intercambio.

En este documento se mencionan los siete algoritmos de ordenación más conocidos.

4.2.1 Ordenación por selección

El algoritmo de ordenación por selección toma elementos de la parte derecha y los coloca en la parte izquierda, empezando por el menor elemento desordenado y lo intercambia con el que ocupa su posición. Así, en la primera iteración se busca el menor elemento y se intercambia con el que ocupa la posición cero en el arreglo. En la segunda iteración, se busca el menor elemento entre la posición uno y

el final y se intercambia con el elemento en la posición uno. De esta manera las dos primeras posiciones del arreglo están ordenadas y contienen los dos elementos menores dentro del arreglo. Este proceso continúa hasta ordenar todos los elementos del arreglo.

De modo que se pueda comprender el funcionamiento de este algoritmo se planteó la siguiente ilustración:

La descripción de este algoritmo se basa en el video llamado: Select-sort with Gypsy folk dance – Link: https://www.youtube.com/watch?v=Ns4TPTC8whw&ab_channel=AlgoRythmics

Se tiene una fila de diez bailarines, cada uno de ellos tiene un número entero del 0 al 9, estos números están ubicados de forma aleatoria de la siguiente manera:

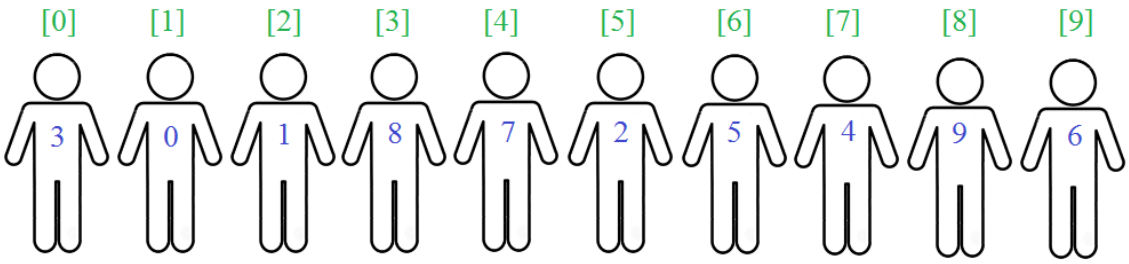


Figura 30. El arreglo contiene 10 posiciones, cada posición está expresada por un número dentro de corchetes.

El algoritmo de ordenación por selección se fundamenta básicamente en la comparación secuencial entre dos números, esto quiere decir, que existen dos posibilidades que el número A sea mayor o menor que el número B.

Posibilidad I

Si el número A es menor que el número B, entonces el número B conservará su posición y el número A continuará la comparación con el siguiente número.

Posibilidad II

Si el número A es mayor que el número B, entonces el número A ocupará la posición del número B.

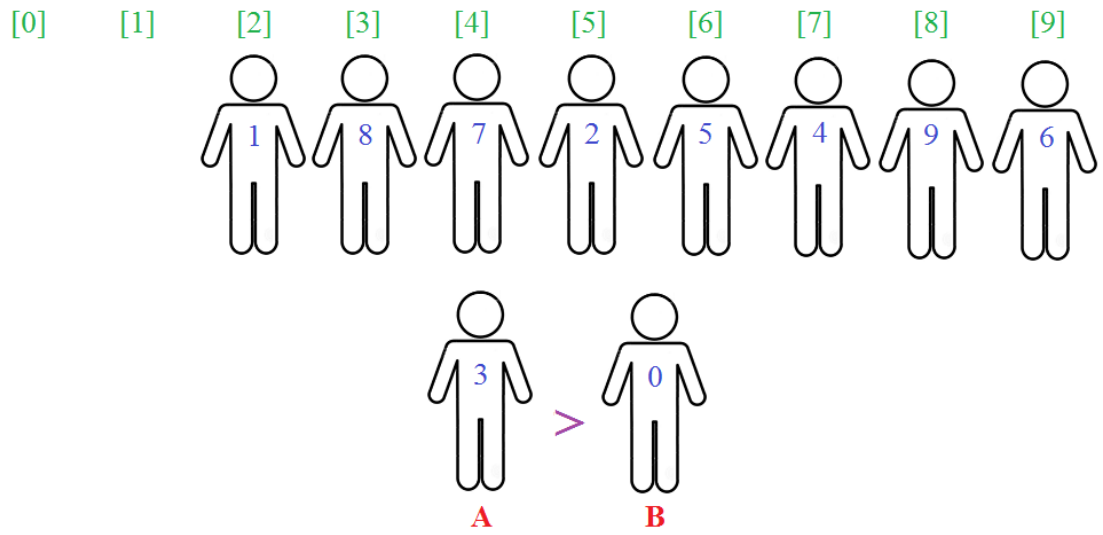


Figura 31. Primera iteración entre el número 3 y el número 0. A y B definen la posición en el arreglo según el resultado de la comparación.

Como el número 3 es mayor que el número 0, A ocupa la posición de B. En consecuencia ahora A pasa a ser el número 0 y B el número 1, como se observa en la figura 32. Enseguida se repite el mismo procedimiento de comparación entre 0 y los demás números.

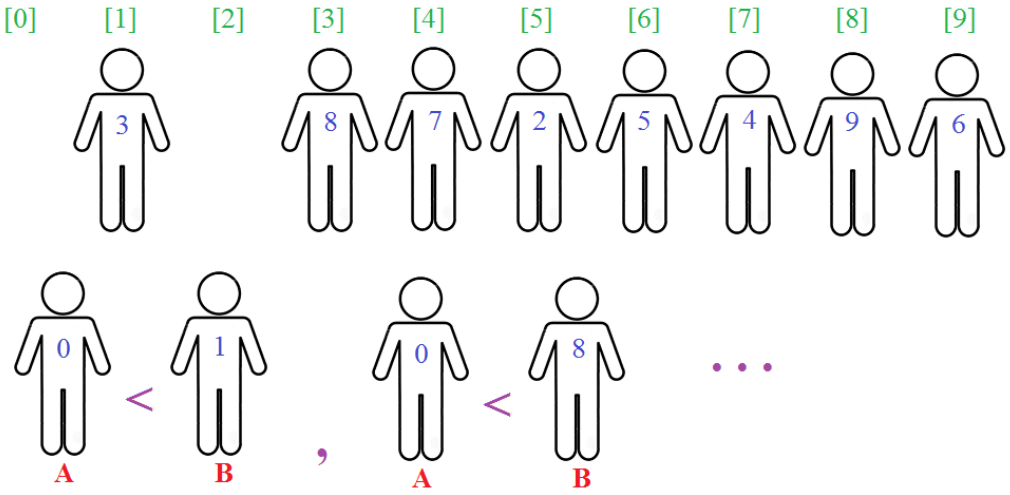


Figura 32. En la figura se observa las iteraciones sucesivas que se llevan a cabo con el objetivo de definir las posiciones en el arreglo según el resultado de cada comparación.

El número 0 es menor que los nueve números restantes, éste ocupa la posición cero en el arreglo, de este modo se ubica al primer bailarín.

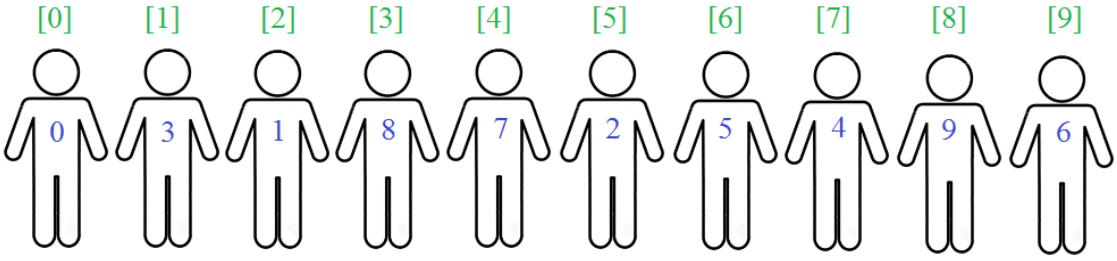


Figura 33. Arreglo resultante de las iteraciones anteriores.

Luego se toma al número 3 como A y se compara con B que sería el número 1. Como el número 3 es mayor que el número 1, este último toma la posición uno del arreglo. Este procedimiento se repite sucesivamente teniendo en cuenta las dos posibilidades planteadas al principio del ejercicio.

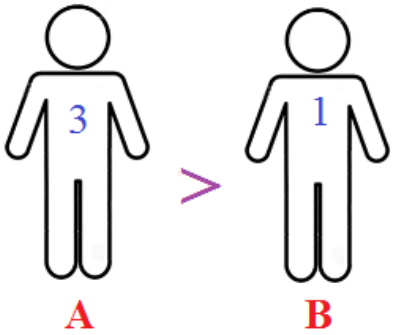


Figura 34. Se observa la compración entre el bailarín número 3 y el bailarín número 1.

En conclusión el arreglo de bailarines queda organizado de la siguiente forma:

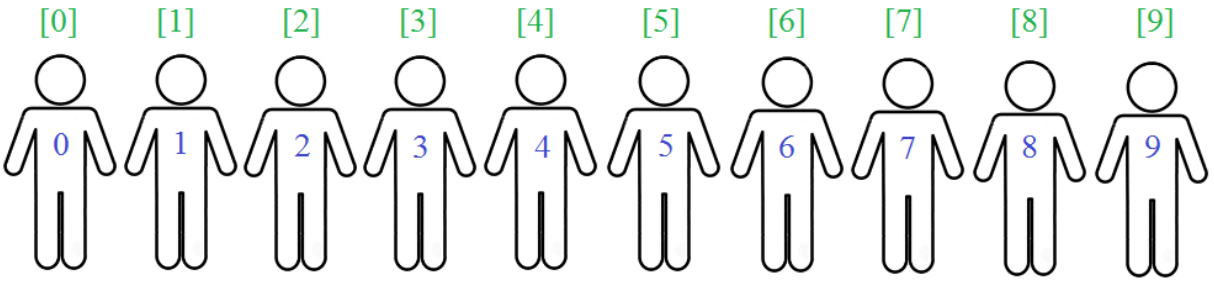


Figura 35. Arreglo ordenado por medio del algoritmo de selección. Cada bailarín ocupa la posición correcta.

Se tuvieron que comparar cuarenta y cinco veces a los bailarines para ordenarlos de menor a mayor.

4.2.2 Ordenación Burbuja

La ordenación burbuja se basa en recorrer el arreglo desde la posición cero hasta la posición *n* un cierto número de veces comparando pares de valores que ocupan posiciones adyacentes. Si ambos datos no están ordenados, se intercambian. Esta operación se

repite *n-1* veces, siendo *n* el tamaño del arreglo. Al final de la última pasada el elemento mayor estará en la última posición.
En la segunda operación, el segundo elemento llegará a la penúltima y así sucesivamente. (Algoritmos de ordenación y búsqueda, s.f).

La comparación de los dos elementos conlleva a realizar varias revisiones en toda la lista hasta organizarlos de menor a mayor o viceversa.

La descripción de este algoritmo se basa en el video llamado: Bubble-sort with Hungarian folk dance – Link: https://www.youtube.com/watch?v=lyZQPjUT5B4&ab_channel=AlgoRythmics

Con la intención de comprender este algoritmo se utilizará el mismo ejemplo anterior.

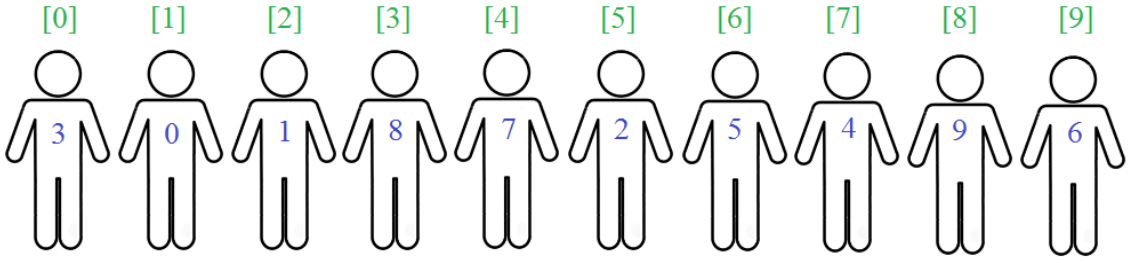


Figura 36. Arreglo inicial de bailarines

Se tiene una fila de diez bailarines, cada uno de ellos tiene un número entero del 0 al 9, estos números están ubicados de forma aleatoria de la siguiente manera:

Cuando se comparan dos números en consecuencia existen dos posibilidades de modo que el número A sea mayor o menor que el número B.

Posibilidad I

Si el número A es menor que el número B, el número A y el número B conservarán la misma posición.

Posibilidad II

Si el número A es mayor que el número B, entonces el número A ocupará la posición del número B.

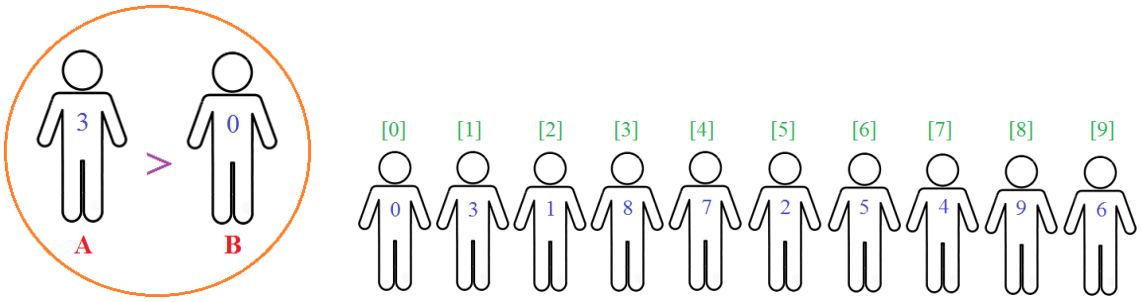


Figura 37. Primera comparación entre el número 3 y el número 0, ubicados respectivamente en las posiciones uno y cero del arreglo.

El primer paso es comparar los dos primeros números, es decir el número 3 y el número 0, como el número 3 es mayor que el número 0 el bailarín A toma la posición del bailarín B, de manera que los diez bailarines quedan ubicados como se evidencia en la figura 37.

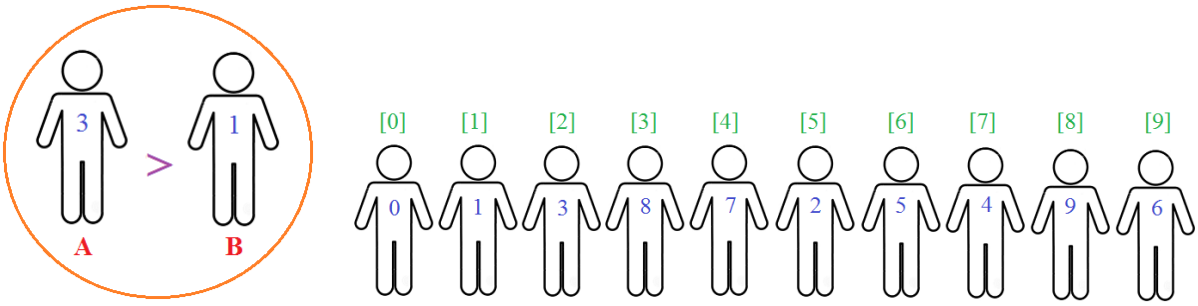


Figura 38. Segunda comparación entre el número 3 y el número 1, ubicados respectivamente en las posiciones dos y uno del arreglo.

De forma semejante que en el primer paso, se compara el número 3 con el número 1 de ahí que, el bailarín A ocupe la posición del bailarín B, obteniendo como resultado el arreglo de bailarines que se visualiza en la figura 38.

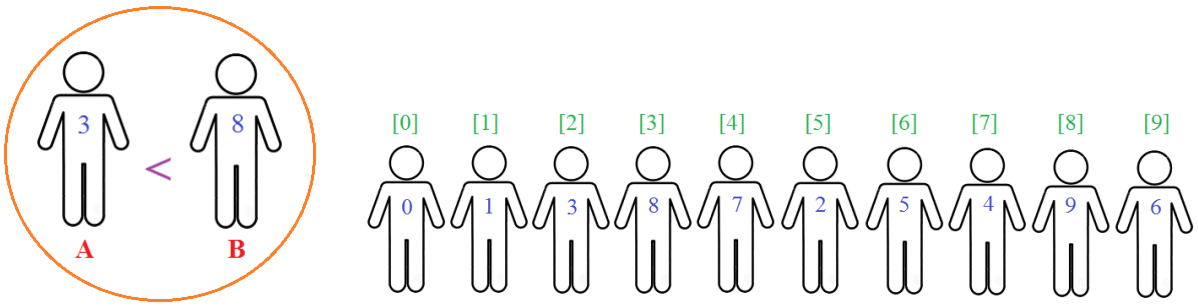


Figura 39. Tercera comparación entre el número 3 y el número 8, ubicados respectivamente en las posiciones dos y tres del arreglo.

En el terce paso, el bailarín A es el número 3 y el bailarín B el número 8, a diferencia de las dos situaciones anteriores el número del bailarín A es menor que el número del bailarin B, a causa de esto, los dos bailarines conservan la misma posición en el arreglo.

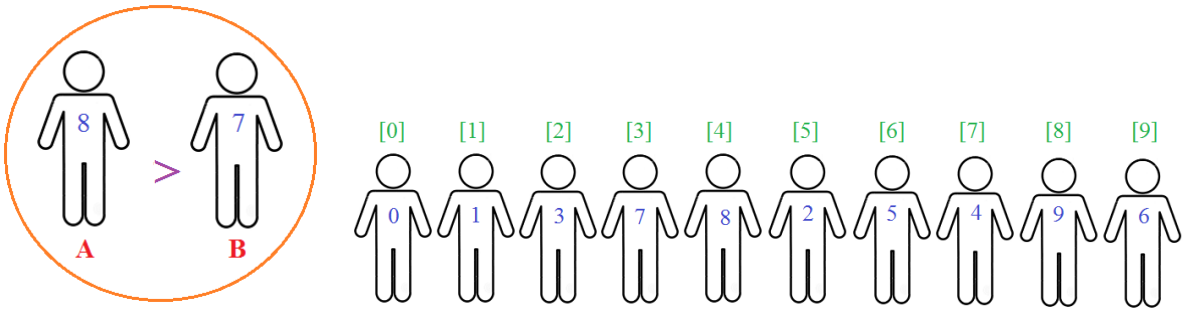


Figura 40. Cuarta comparación entre el número 8 y el número 7, ubicados respectivamente en las posiciones cuatro y tres del arreglo.

En el cuarto paso, se repite la situación del primer y segundo paso, como el número del bailarín A es mayor que el número del bailarín B, éstos intercambian de posición.

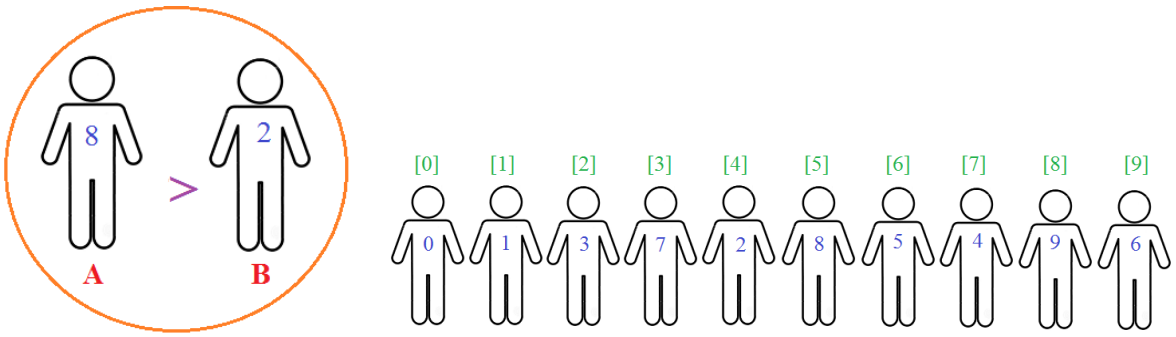


Figura 41. Quinta comparación entre el número 8 y el número 2, ubicados respectivamente en las posiciones cinco y cuatro del arreglo.

En el quinto paso, el número del bailarín A vuelve a ser mayor que el número del bailarín B, esto conlleva a que intercambien sus posiciones.

Por consiguiente el bailarín con el número 2 y el bailarín con el número 8 ocupa la posición cuatro y cinco en el arreglo respectivamente. Como se ha mostrado el nombre de este algoritmo, se debe a que el bailarín cuyo número es mayor sube a la posición final del arreglo, al igual que las burbujas de aire en un depósito

suben a la parte superior. Para ello deben realizar un recorrido paso a paso desde su posición inicial hasta la posición final del arreglo.

En conclusión, este procedimiento se repite veintidós veces para ordenar de menor a mayor los números de cada bailarín.

4.2.3 Ordenación por Inserción

El algoritmo de ordenación por inserción utiliza un método muy similar al algoritmo de ordenación por selección, de forma semejante toma un elemento de la parte no ordenada para colocarlo en su lugar. El primer elemento del arreglo es comparado con el segundo elemento, el elemento que sea menor se ubica delante del elemento mayor. Se repite esta operación sucesivamente de tal modo que se va colocando cada elemento en la posición correcta. (Algoritmos de ordenación y búsqueda, s.f).

Para colocar el elemento en su lugar, se debe encontrar la posición que le corresponde en la parte ordenada del arreglo y crear un espacio de forma que se pueda insertar. Es preciso señalar que para encontrar la posición se puede hacer una búsqueda secuencial desde el principio del arreglo hasta encontrar un elemento mayor que el dado. Del mismo modo, para crear un espacio es necesario desplazar los elementos pertinentes una posición a la derecha.

La descripción de este algoritmo se basa en el video llamado: Insert-sort with Romanian folk dance – Link: https://www.youtube.com/watch?v=ROalU379l3U&ab_channel=AlgoRythmics

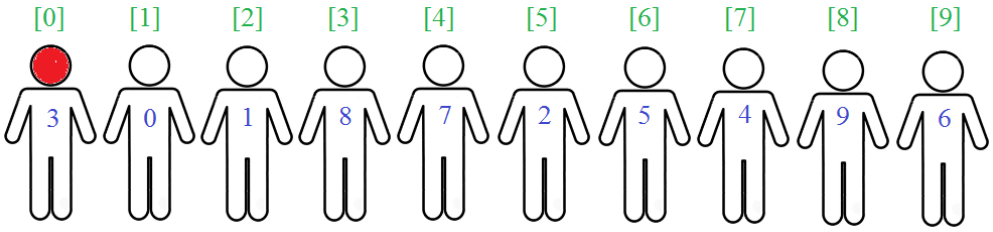


Figura 42. Arreglo inicial de bailarines. Para ordenar a los bailarines se implementará el algoritmo de ordenación por inserción.

En la figura 42 está representado el arreglo con diez bailerines , cada uno ubicado de manera aleatoria con su respectivo número. El primer paso que indica el algoritmo de ordenación por inserción, consiste en seleccionar al bailarín ubicado en la posición cero del arreglo, en este ejemplo seria el bailarín número 3. (Vesase figura 42).

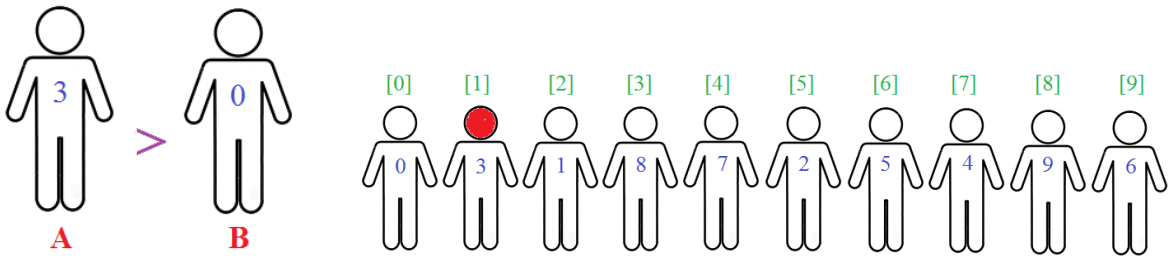


Figura 43. En la parte izquierda se observa la primera comparación que se realiza con el fin de determinar la nueva posición de los dos primeros bailarines en el arreglo.

Para empezar, se compara el número del bailarín A con el número del bailarín B. Como el número 3 es mayor que el número

0, éste se desplaza a la posición cero del arreglo y el bailarín número 3 a la posición uno.

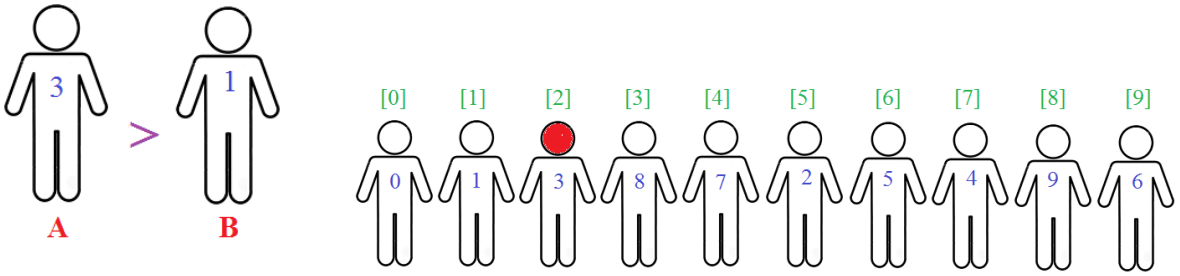


Figura 44. En la parte derecha se evidencia como el bailarín número 3 se desplaza de la posición cero a la posición dos del arreglo después de realizar dos comparaciones con el bailarín adayacente de cada iteración.

El bailarín B ahora pasa a ser el bailarín número 1, éste se compara con el bailarín A que sigue siendo el bailarín número 3. Dado que, el número 3 es mayor que el número 1, los dos bailarines intercambian posición como se observa en el arreglo de la figura 44.

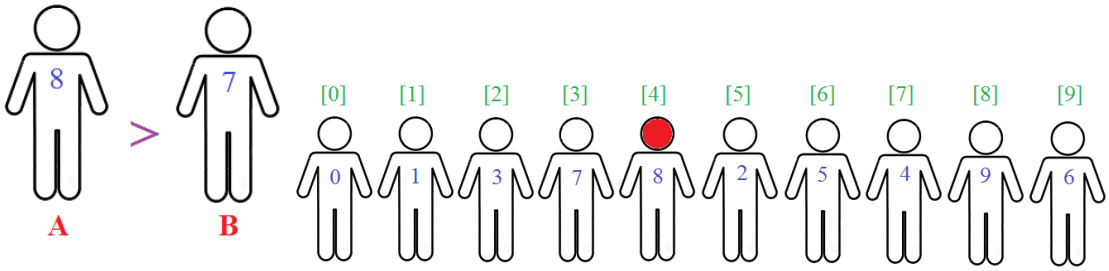


Figura 45. Se divisa el bailarín número 8 como nuevo referente.

Nótese que el bailarín número 3 ha sido el bailarín referente en las dos comparaciones anteriores, debido a que su número es mayor respecto a los dos bailarines que ahora ocupan la posición cero y uno del arreglo. No obstante, en la tercera comparación el número referente deja de ser el 3 para ser el 8 (Ver figura 45). A causa

de esto, en el siguiente paso se lleva a cabo la comparación entre el bailarín número 8 y el bailarín número 7.

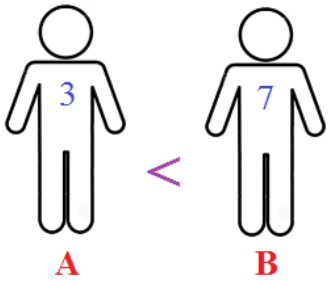


Figura 46. Comparación entre el bailarín número 3 y el bailarín número 7.

Luego de comparar y ubicar al número 8 y al número 7, el siguiente paso es realizar la comparación entre el bailarín número 3 y el bailarín número 7, por causa de que alguno de los números ubicados en las tres primeras posiciones del arreglo puede ser mayor que el número 7. Para este ejercicio no se tiene en cuenta esta comparación, puesto que de antemano, se sabe que los tres primeros números son menores que 7. Sin embargo, se hace énfasis en la importancia de la realización de esa comparación cuando se pretende implementar este tipo de algoritmo en algún lenguaje de programación.

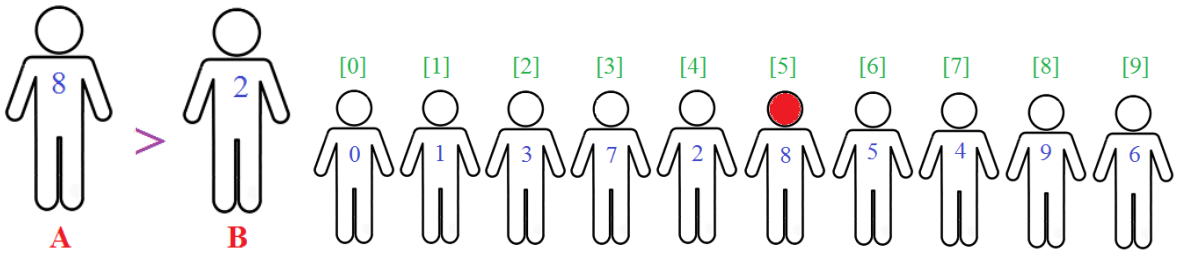


Figura 47. Comparación entre el bailarín número 8 y el bailarín número 2.

Tan pronto como se lleven a cabo las comparaciones en cada iteración entre el bailarín referente y los bailarines adyacentes, éste

se ira desplazando hacia la derecha , de tal forma que se ubique en la posición correcta. Por otro lado, los bailarines que se van desplazando de derecha a izquierda también tienen que compararcen. La figura 28 ilustra tres comparaciones que se realizan, con el objetivo de ubicar a cada bailarín de forma ascendente.

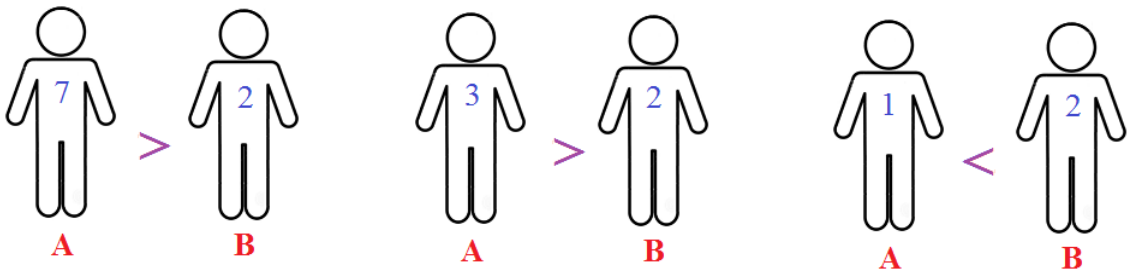


Figura 48. Para poder ubicar al bailarín número 2 en la posición dos del arreglo es necesario realizar la comparación de éste con los bailarines 7, 3 y 1.

Enseguida, se compara al bailarín referente con el bailarín adyacente, es decir, con el bailarín número 5. Al establecer la relación de semejanza entre los dos bailarines, el bailarín número 5 pasa a ocupar la posición cinco del arreglo como se observa en la figura 49. Después de eso, se realizan dos comparaciones entre el bailarín número 5 y los bailarines 7 y 3 de modo que, el bailarín 5 ocupe la posición cuatro del arreglo.

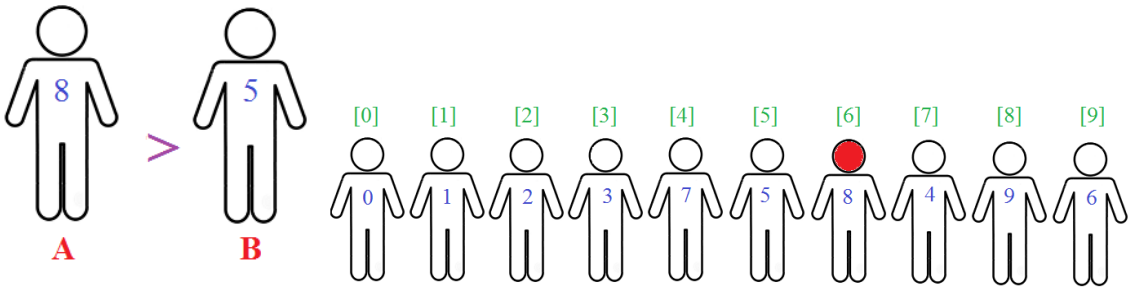


Figura 49. Comparación entre el bailarín número 8 y el bailarín número 5.

Una vez hecho lo anterior, se llevan a cabo dos comparaciones entre el baialrín número 8 y los bailarines número 4 y 9 respectivamente, cada comparación se realiza en una iteración diferente. El bailarín número 4 se desplaza a la posición cuatro, después de haberse comparado con los bailarines 5 y 7. De manera semejanta sucede con el bailarín número 6, el cual pasa a ocupar la posición seis en el arreglo. Finalmente, del resultado de las comparaciones anteriores, se obtiene el arreglo ordenado de menor a mayor.

4.2.4 Ordenación por Mezcla o Merge Sort

Igual que los algoritmos anteriores, el algoritmo de ordenación por mezcla se fundamenta en la realización de comparaciones entre elementos de una lista vistos como un arreglo. La idea de este algoritmo es dividir el arreglo por la mitad, cada mitad se ordena de forma recursiva, comparando los pares de elementos implícitos en éste. A medida que se van comparando los elementos, éstos van

ordenándose en otro arreglo. A continuación las dos mitades ya ordenadas se mezclan formando una secuencia clasificada.

La descripción de este algoritmo se basa en el video llamado: Merge-sort with Transylvanian-saxon (German) folk dance – Link: https://www.youtube.com/watch?v=XaqR3G_NVoo&ab_channel=AlgoRythmics

Con el fin de presentar de forma didáctica el algoritmo de ordenación por mezcla se desarrolla el siguiente ejemplo:

Se cuenta nuevamente con un arreglo de diez bailarines posicionados aleatoriamente como se observa en la figura 50.

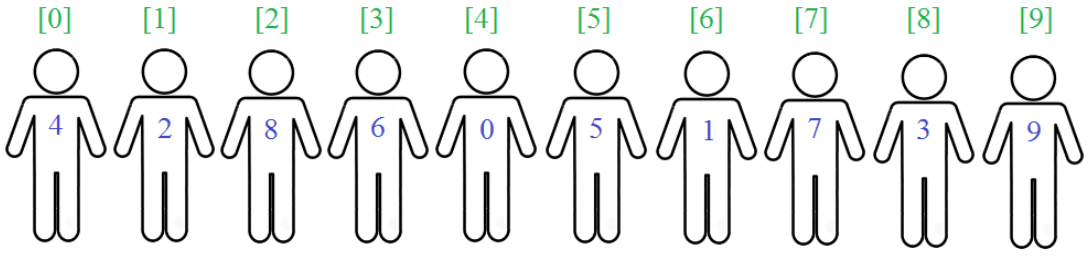


Figura 50. Se observa a los diez bailarines ubicados inicialmente de forma aleatoria cada uno con un número de 0 a 9.

En primer lugar, se divide el arreglo original en dos partes iguales o aproximadamente iguales. Para este ejemplo en especifico, se crearon dos subgrupos de cinco bailarines.

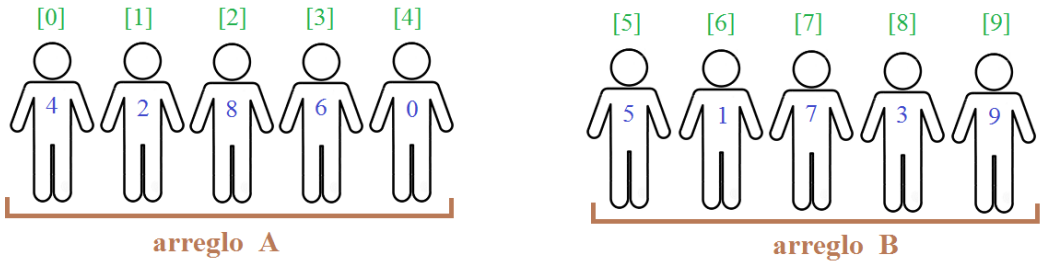


Figura 51. El arreglo de la figura 50 se dividió en dos arreglos llamados arreglo A y arreglo B.

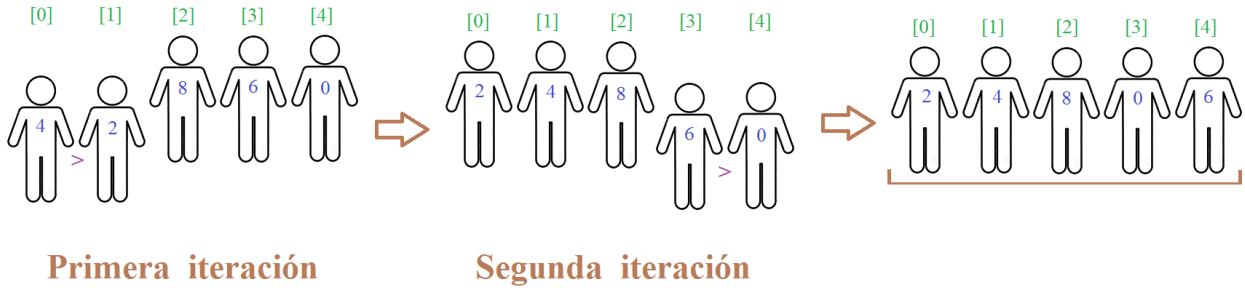


Figura 52. Resultado de la primera y segunda iteración o ejecución, aplicando el algoritmo de ordenación por mezcla.

Después de haber separado el arreglo original en dos sub-arreglos, se comparan los números de los dos bailarines que están ubicados en las posiciones cero y uno del arreglo, es decir, el bailarín número 4 y el bailarín número 2. (Ver figura 52)

Obsérvese, que en la segunda iteración los bailarines número 2 y número 4 se encuentran ordenados de forma ascendente como resultado de la comparación realizada en la primera iteración. Seguidamente, se comparan los bailarines número 6 y número 0 obteniendo el arreglo que se visualiza en la lateral derecha de la figura 52.

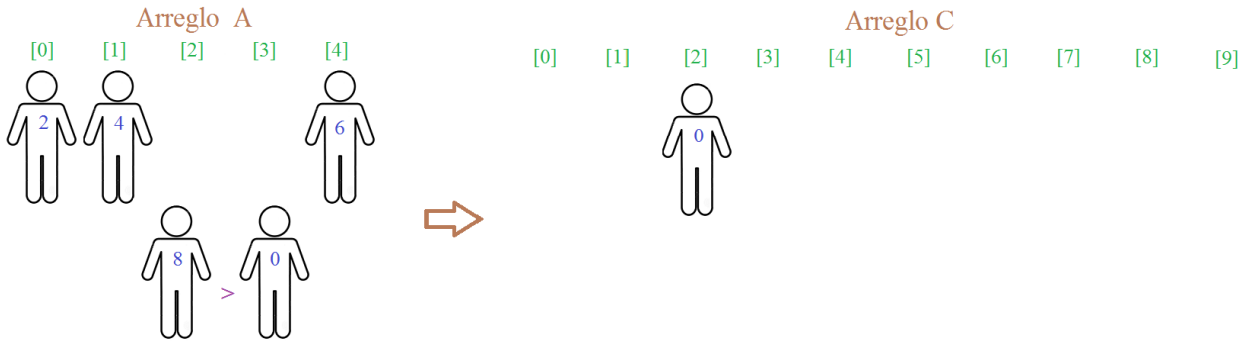


Figura 53. Resultado de la tercera ejecución, aplicando el algoritmo de ordenación por mezcla.

Con la intención de ordenar de manera rápida a los bailarines, se implementa un arreglo C con la misma cantidad de posiciones del arreglo original. El bailarín número 8 es comparado con el bailarín número 0, luego se ubica en la posición dos del arreglo C, siendo la misma posición que ocupará posteriormente en el arreglo A.

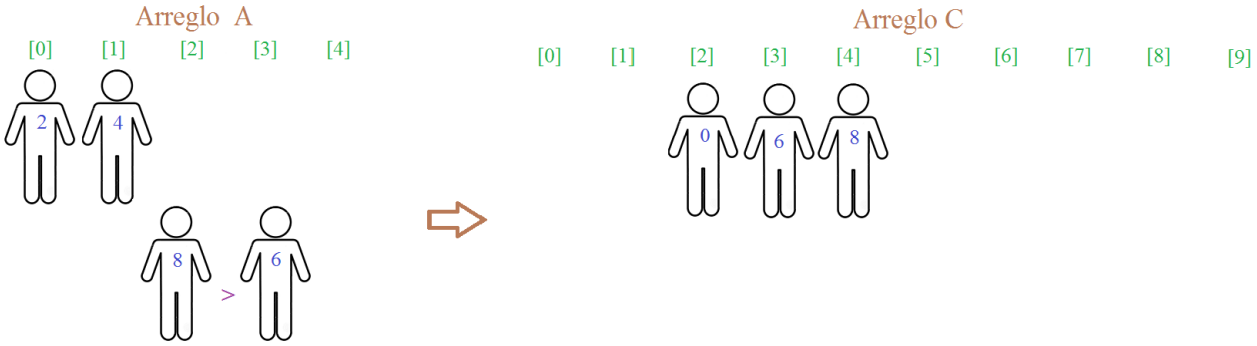


Figura 54. Resultado de la cuarta ejecución, aplicando el algoritmo de ordenación por mezcla.

Una vez que se encuentra ubicado el bailarin número 0 en el arreglo C, de inmediato se compara al bailarín número 8 con el bailarín número 6, esto conlleva a ubicar de manera ascendente a los tres bailarines, de ahí que este algoritmo facilite ir comparando e ir ubicando a cada bailarín.

La figura 55 muestra cómo queda ordenado el arreglo A, como resultado de las primeras iteraciones

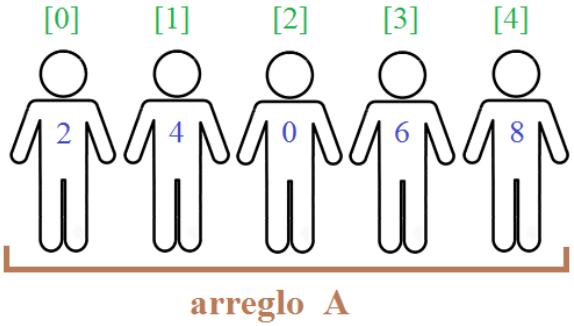


Figura 55. Resultado de las cuatro ejecuciones iniciales descritas anteriormente aplicando el algoritmo de ordenación por mezcla.

Es preciso señalar que en un primer instante se comparaban los bailarines que estaban adyacentes. De ahí que en un segundo instante, se lleven a cabo las comparaciones que se mencionan a continuación.

Específicamente, se hace referencia a la comparación entre el bailarín número 2 y el bailarín número 0. Como se muestra en la figura 56 el resultado de esta comparación permite ubicar al bailarín número cero en la primera posición del arreglo C.

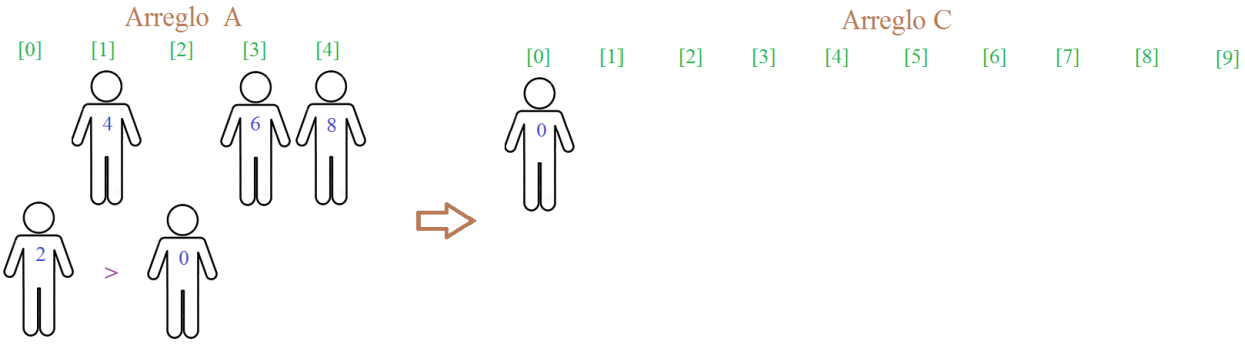


Figura 56. En la ejecución cinco se compara n los dos bailarines y según corresponda , lo sitúa en el arreglo C.

Del mismo modo, se compara al bailarín número 2 y al bailarín número 6. En consecuencia con esta acción se ubica al bailarín número 2 en la segunda posición del arreglo C.

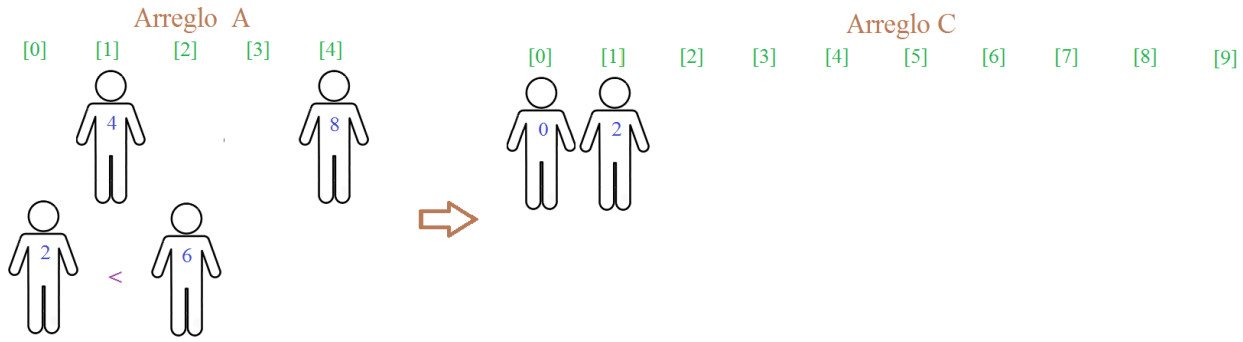


Figura 57. En la ejecución seis se comparan los dos bailarines y según corresponda, lo sitúa en el arreglo C.

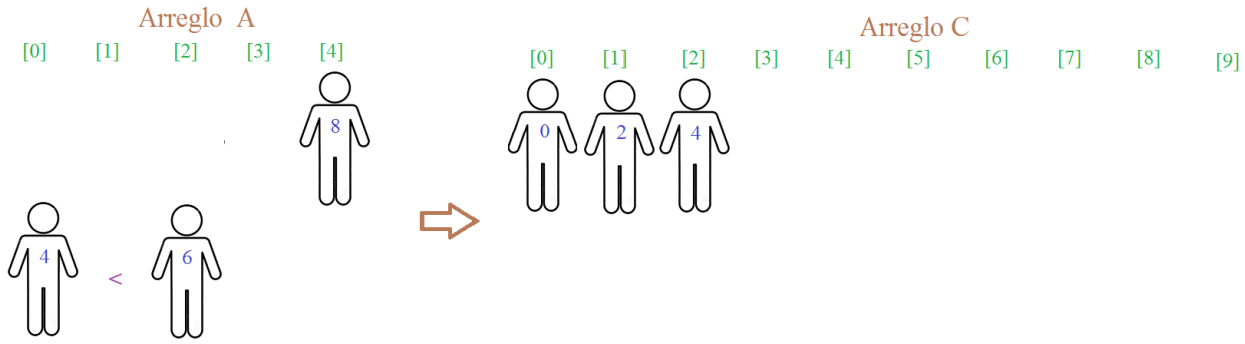


Figura 58. Ejecuación siete. Se observan los tres primeros bailarines ubicados de forma ascerndente en el arreglo C.

De forma semejante, se repite la misma acción anterior entre el bailarín número 4 y el bailarín número 6. En síntesis se obtiene el arreglo A ordenado de menor a mayor. De manera análoga, se ordena el arreglo B repitiendo el mismo procedimiento anterior. Para finalizar la aplicación de este algoritmo se entrelaza el arreglo A y el arreglo B.

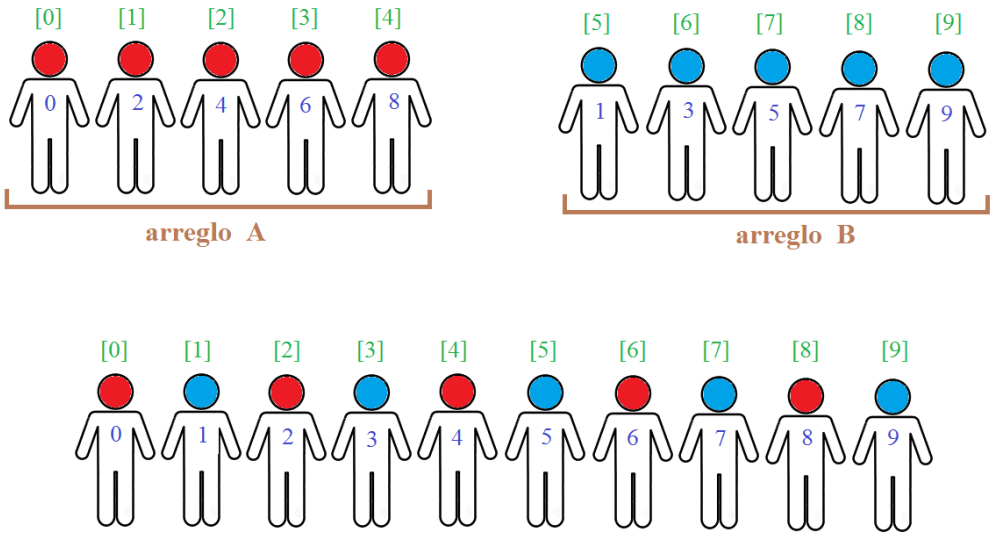


Figura 59. Dado que los dos arreglos estan ordenados ascendentemente, se mesclan los bailarines de manera alterna. De ahí que este algoritmo reciba el nombre de ordenación por mezcla.

En efecto se necesitan menos pasos para construir una lista ordenada a partir de dos listas también ordenadas.

4.2.5 Ordenación por montículos – Heapsort

A diferencia del algoritmo de ordenación por mezcla, el algoritmo de ordenación por montículos no es recursivo, esto quiere decir, que no es un algoritmo que expresa la solución de un problema en términos de una llamada a sí mismo.

Este algoritmo almacena todos los elementos en un arreglo, por medio de la implementación de estructuras conocidas como montículos, que hacen referencia a una clase especial de árbol. Este tipo de árbol generalmente se dibuja de forma triangular, cuyos vértices están representados por círculos o nodos que poseen datos. Cada nodo simboliza una posición en el arreglo.

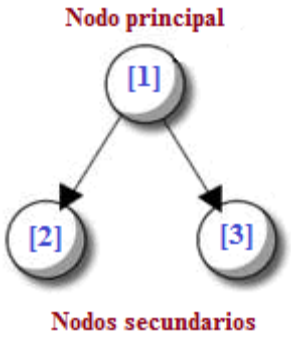


Figura 60. Estructura general de un montículo, también conocida como árbol binario.

La cantidad de nodos depende de la cantidad de posiciones en el arreglo. Por ejemplo: si el arreglo contiene siete posiciones la estructura monticular debe formar las mismas. Conviene señalar que ningún nodo puede tener más de dos extensiones o subárboles. De ahí que, en un árbol binario se pueda tener cero, uno o dos subárboles.

Con respecto al procedimiento que se debe llevar a cabo, el algoritmo se fundamenta en la ordenación por comparación, realizando esta acción en un primer instante entre los dos nodos de la parte inferior . El nodo inferior con el número mayor, será comparado con el nodo principal en un segundo instante. En consecuencia, el número mayor se ubicará en la raíz o nodo principal de la estructura monticular.

Otra particularidad que se debe tener en cuenta, es que este algoritmo utiliza la propiedad de la raíz para ordenar el arreglo. Una vez que el elemento mayor este en la raíz o nodo principal, se quita éste y se coloca al final del arreglo. Con los datos que sobran, se crea otro montículo y se

repite el mismo procedimiento hasta que todos los datos estén ordenados.

La descripción de este algoritmo se basa en el video llamado: HEAP-sort with Hungarian (MEZOSÉGI) folk dance – Link: https://www.youtube.com/watch?v=Xw2D9aJRBY4&ab_channel=AlgoRythmics

A fin de comprender este algoritmo se desarrolla el siguiente ejemplo:

Se tienen siete bailarines posicionados de forma monticular y ubicados aleatoriamente como se observa en la figura 61.

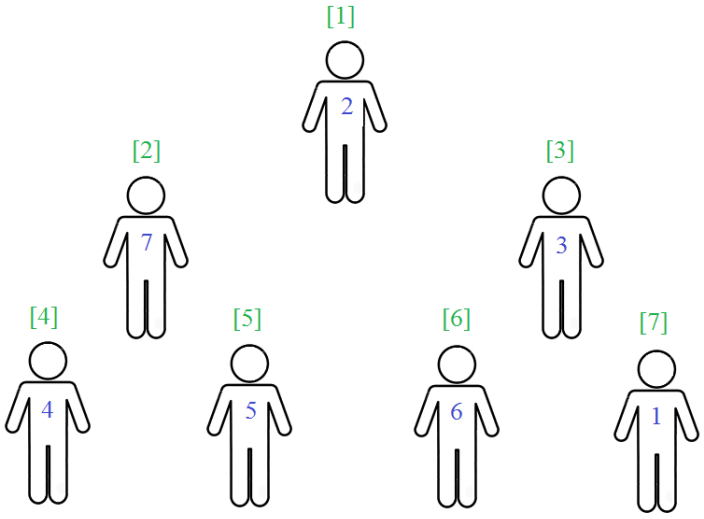


Figura 61. Se evidencia tres estructuras la primera está integrada por los bailarines con los números 2, 7 y 4, la segunda por los bailarines con los números 7, 4 y 5 y la última por los bailarines con los números 3, 6 y 1.

La posición uno se le concoce con el nombre de raiz, ésta representa el nodo principal del triangulo formado por la posición dos y tres. Del mismo modo la posición dos constituye el nodo principal en el triangulo formado por las posiciones

cuatro y cinco. Así como la posición tres figura como nodo principal del triangulo hecho con las posiciones seis y siete. Teniendo claro lo anterior se realiza la primera y segunda comparación como se muestra en la figura 62.

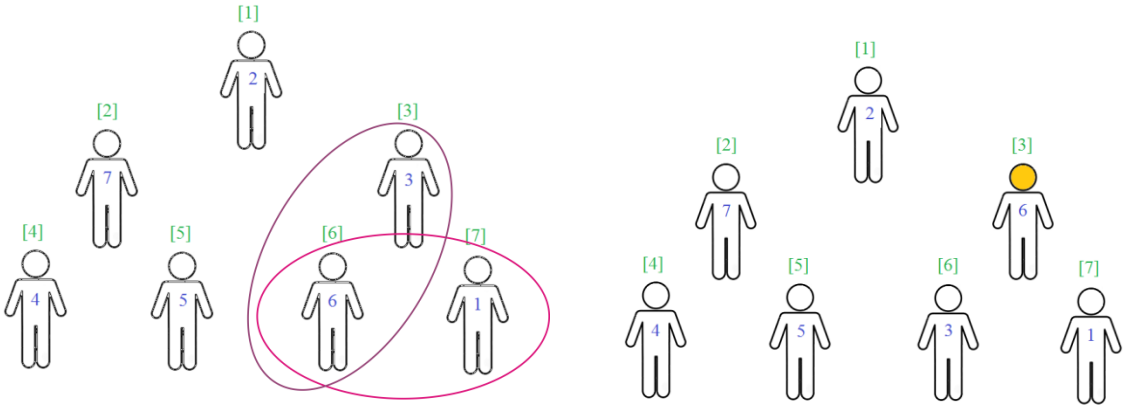


Figura 62. Las elipses encierran a los bailarines que se comparan en el primer y segundo turno. Dada estás dos comparaciones, se ubica al bailarín número 6 en el nodo principal del tercer triangulo.

La primera comparación se realiza entre el bailarín número 6 y el bailarín número 1. Como consecuencia, de que el número 6 es mayor que el número 1 se lleva a cabo la segunda comparación entre el bailarín número 6 y el bailarín número 3. Esta comparación ubica al bailarín número 6 en la posición tres de la estructura monticular.

Como se ha mostrado en el figura 62 se cumple con la parte inicial del procedimiento mencionado en la descripción inicial de este algoritmo.

La tercera y cuarta comparación se observa en la figura 63, consecuentemente ninguno de los bailarines cambia de posición

puesto que, el bailarín con el número mayor respecto a las posiciones dos, cuatro y cinco, se encuentra ubicado en el nodo principal de la estructura monticular.

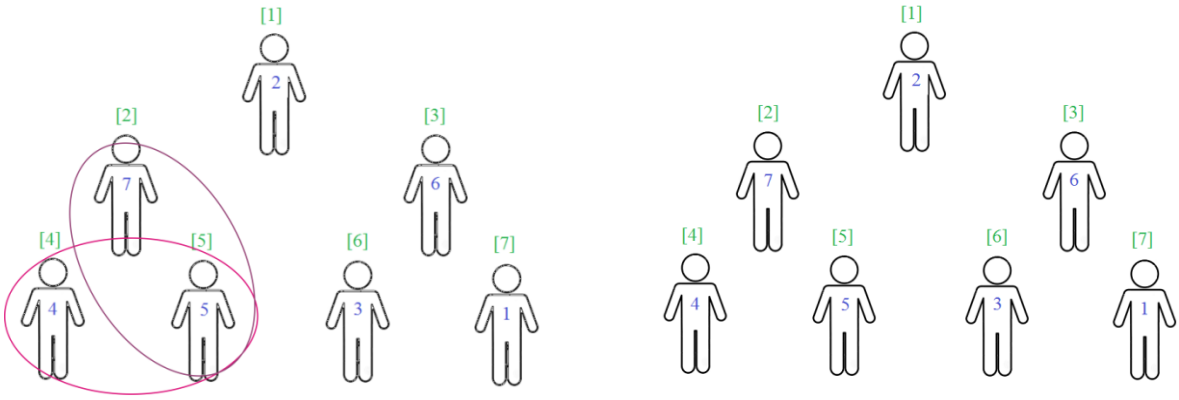


Figura 63. Las elipses encierran a los bailarines que se comparan en el tercer y cuarto turno.

La última estructura monticular a comparar para extraer al primer bailarín, se encuentra en la parte superior, la integran las posiciones uno, dos y tres. Siguiendo el orden estipulado, la quinta comparación se da entre el bailarín número 7 y el bailarín número 6. Luego se intercambia al bailarín número 2 por el bailarín número 7.

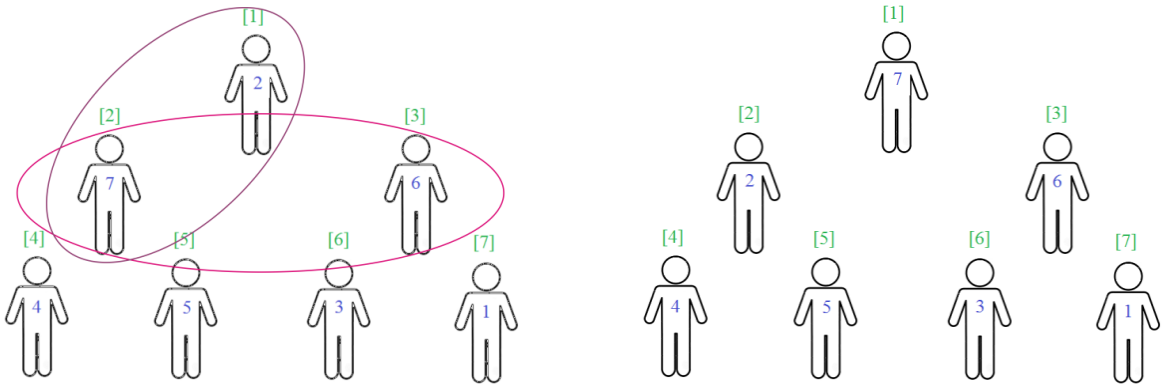


Figura 64. Las elipses encierran a los bailarines que se comparan en quinto y sexto turno.

A consecuencia de lo anterior, el bailarín con el número mayor ya se encuentra en la raíz de la estructura monticular. Sin embargo, aún no se puede retirar, porque una de las condiciones que permite ubicar al número de la raíz en el arreglo, exige que cada nodo principal deba contener el número mayor de los subárboles. Por este motivo, si se observa la posición dos de la estructura que se encuentra a la derecha, (figura 64) no cumple con lo mencionado anteriormente.

Con el fin de aplicar el algoritmo de forma correcta, se vuelve a comparar al bailarín número 4 con el bailarín número 5 y nuevamente a este último con el bailarín número 2. Luego de estas comparaciones cada nodo principal contiene el número mayor de su subárbol, por consiguiente se puede retirar al bailarín número 7 para ubicarlo en el arreglo.

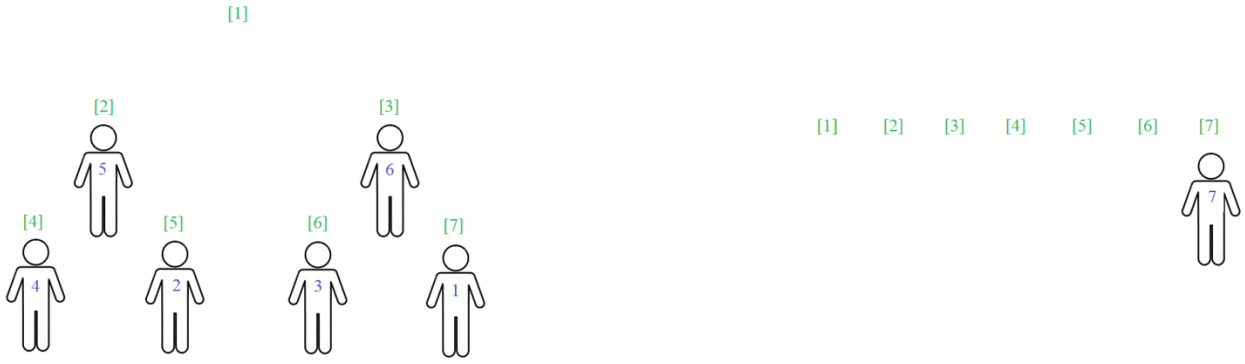


Figura 65. Se retira al bailarín número 7 de la estructura monticular y se ubica en la posición siete del arreglo.

La posición uno o raíz de la estructura monticular, es ocupada por el bailarín que se encuentra en la posición siete de esta

misma estructura (Ver figura 66). De lo anterior resulta necesario decir, que cada vez que se retire al bailarín que se encuentra en la raíz, de inmediato esta posición será ocupada por el bailarín que se encuentra en la misma posición la cual va a ocupar el bailarín saliente pero en la estructura monticular.

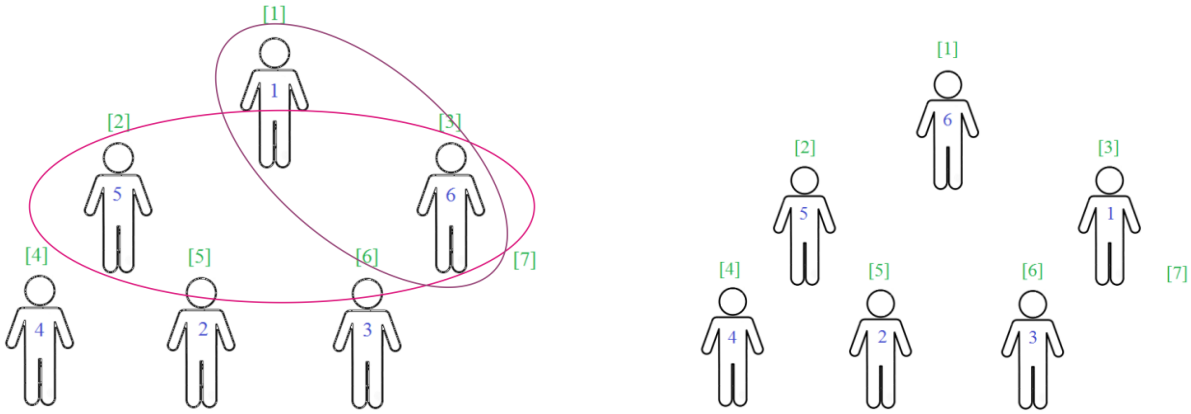


Figura 66. El bailarín número 1 pasa de estar en la posición siete a ocupar la posición uno de la estructura monticular.

Una vez que el bailarín número 1 ocupa la posición uno, se realizan dos comparaciones una seguida de la otra.

La primera tiene como objetivo definir cuál bailarín entre el 5 y el 6 intercambiará la posición con el bailarín número 1.

Con base en el resultado anterior la segunda comparación se lleva a cabo entre el bailarín número 1 y el bailarín número 6, teniendo en cuenta que el bailarín número 6 es el mayor de los tres bailarines que componen el subárbol. Éste pasa a la posición uno como se muestra en la parte derecha de la figura 66.

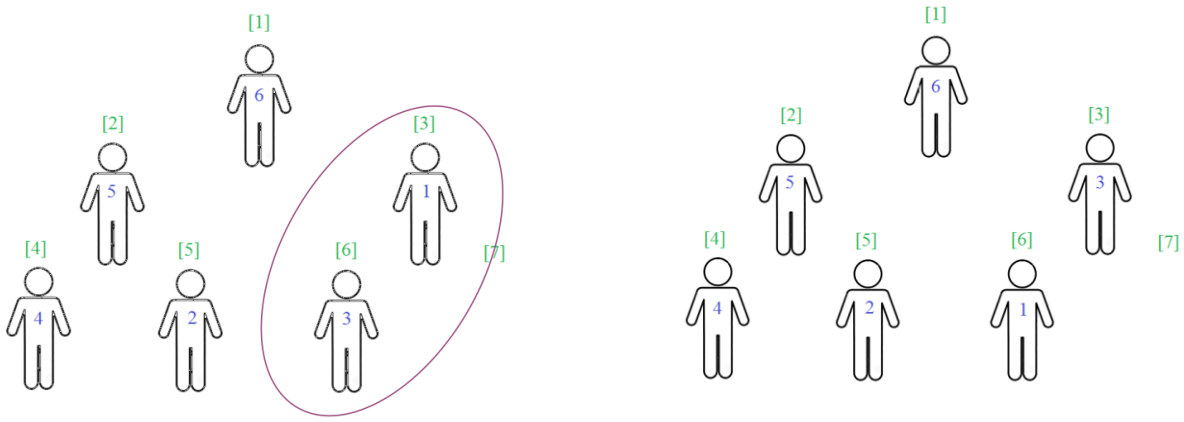


Figura 67. En la figura se evidencia sólo una comparación ya que el subárbol número tres únicamente contiene dos bailarines. Luego de realizar la comparación el bailarín número 3 pasa a la posición tres de la estructura.

Antes de retirar al bailarín número 6 de la raíz, es necesario inspeccionar si el nodo principal de cada subárbol contiene el número mayor de las tres posiciones que lo conforman. Observada la figura 67 es indispensable invertir la posición del bailarín número 3 y el bailarín número 1. Del resultado de esta acción, se obtiene la estructura monticular que se muestra en la parte derecha de la figura 67.

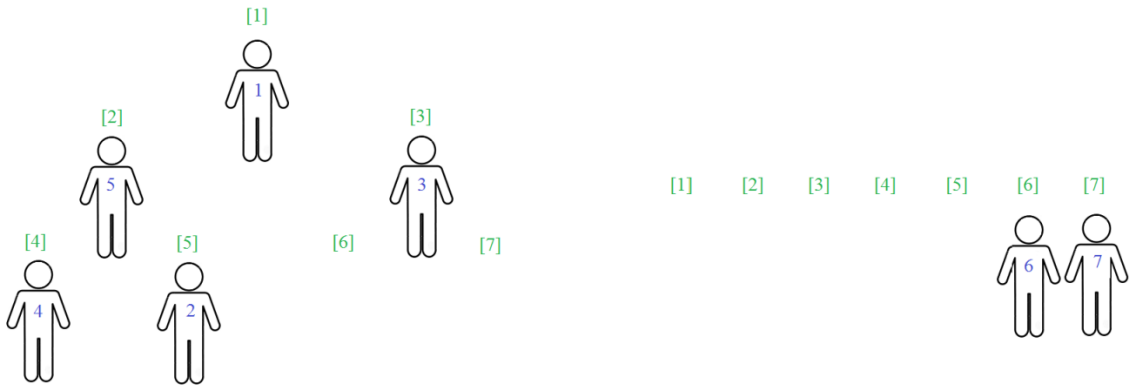


Figura 68. Nótese que en la estructura monticular las posiciones seis y siete se encuentran vacías. Debido a la ubicación de los bailarines 6 y 7 en el arreglo.

La misma metodología se implementa realizando dos comparaciones entre el bailarín número 5 y los bailarines número 3 y número 1, de ahí que en el siguiente turno se pueda retirar al bailarín número 5 de la estructura monticular. Tan pronto se retire al bailarín número 5 la posición de éste es ocupada por el bailarín número 2.

Para poder extraer al bailarín número 4, primero se compara éste con el bailarín número 3. Enseguida cambia de posición con el bailarín número 2, por lo que el bailarín número 4 ocupa la posición raíz.

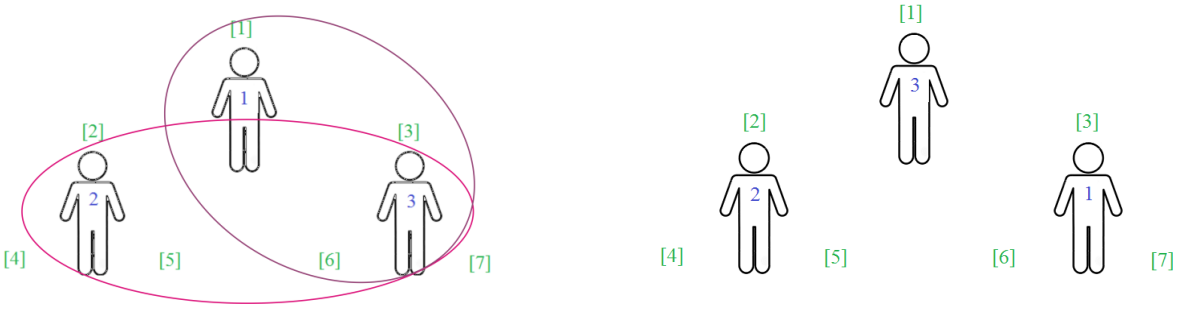


Figura 69. El bailarín número 3 pasa de estar en la posición tres a ocupar la posición uno de la estructura monticular.

La estructura monticular pasa de tener tres subárboles a tener uno, en suma quedan tres bailarines para ubicar en el arreglo.

La figura 69 evidencia la primera comparación que se realiza entre el bailarín número 2 y el bailarín número 3. Siendo mayor el número 3 ocupa la posición raíz y se procede a retíralo.

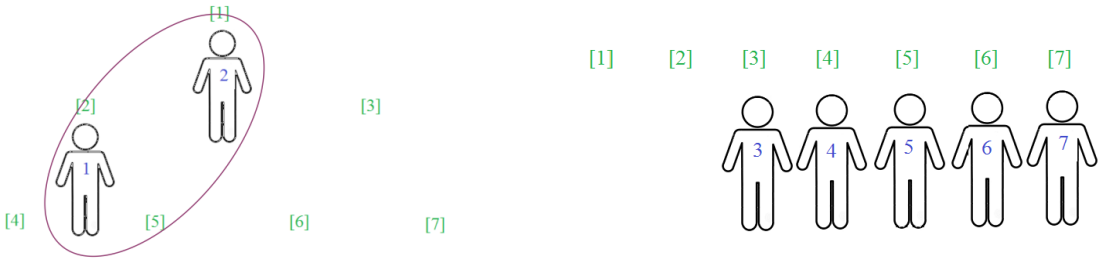


Figura 70. El bailarín número 2 pasa de estar en la posición dos a ocupar la posición uno de la estructura monticular.

Finalmente, el bailarín número 1 pasa a la posición raíz y quedando sólo en la estructura monticular se retira y ubica en la posición uno del arreglo, es de esta forma, como siguiendo el procedimiento antes descrito se logra ordenar los siete bailarines.

En síntesis, este algoritmo resulta ser más rápido que los algoritmos de ordenación por burbuja y ordenación por inserción. Siempre que se inserte o elimine elementos de un montículo, es necesario cuidar que no se destruya la propiedad de orden de éste.

Como se evidencio en el ejercicio analizado anteriormente lo que se hace generalmente es construir rutinas de filtrado ya sean ascendentes o descendentes que tomen el elemento del montículo que viola la propiedad de orden y los muevan diagonalmente por el árbol hasta encontrar una posición en el cual se respete el orden entre los elementos del montículo.

4.2.6 Ordenación rápida – Quicksort

El algoritmo de ordenación rápida al igual que el algoritmo de ordenación por mezcla son frutos de la técnica de resolución de algoritmos *divide y vencerás*. Esta técnica se basa en dividir el problema en subproblemas más pequeños para resolverlos cada uno por separado aplicando la misma técnica y al final unir las soluciones. (Algoritmos de ordenación y búsqueda, s.f).

La metodología de ejecución que a continuación se expone hace que este algoritmo sea el algoritmo de ordenación más rápido en la práctica.

El primer paso consiste en elegir un elemento P llamado pivote del arreglo de datos. El segundo paso es aplicar la comparación entre el pivote (P) y cada uno de los elementos del arreglo, a causa de esto se consigue particionar el arreglo de datos en tantas divisiones como sea posible. La cantidad de divisiones depende del número de elementos y la ubicación de cada uno de ellos en el arreglo. Estas particiones tienen como fin ordenar de manera rápida los elementos que las componen.

Por último, se lleva a cabo el paso *divide y vencerás*, lo cual quiere decir unir todas las soluciones para formar el arreglo con todos los elementos ordenados.

Como parte de la descripción del algoritmo se desarrollarán a continuación dos ejemplos.

La descripción del primer ejemplo se basa en el video llamado: Quick-sort with Hungarian (Küküllomenti legényes) folk dance – Link: https://www.youtube.com/watch?v=ywWBy6J5gz8&ab_channel=AlgoRythmics

El primer ejemplo trata de un arreglo con 10 bailarines ubicados de manera aleatoria como se observa en la figura 71.

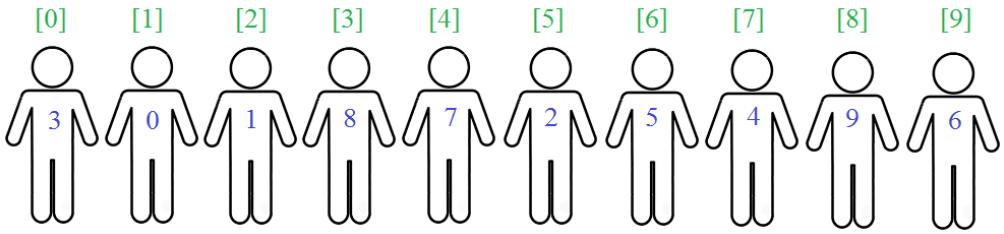


Figura 71. Arreglo inicial de bailarines. Desde este punto se parte con el propósito de ordenarlo por medio del algoritmo de ordenación rápida.

Con base en la metodología descrita anteriormente se selecciona como pivote (P) al bailarín número 3. Después de eso, se compara al bailarín número 3 con cada uno de los bailarines, empezando con el bailarín que se encuentra ubicado en la posición nueve del arreglo. Las comparaciones se llevan a cabo hasta el bailarín número 2 como se observa en la figura 72. Debido a que el número 3 es mayor al número 2, esto conlleva a que los dos bailarines inviertan su posición creando una partición.

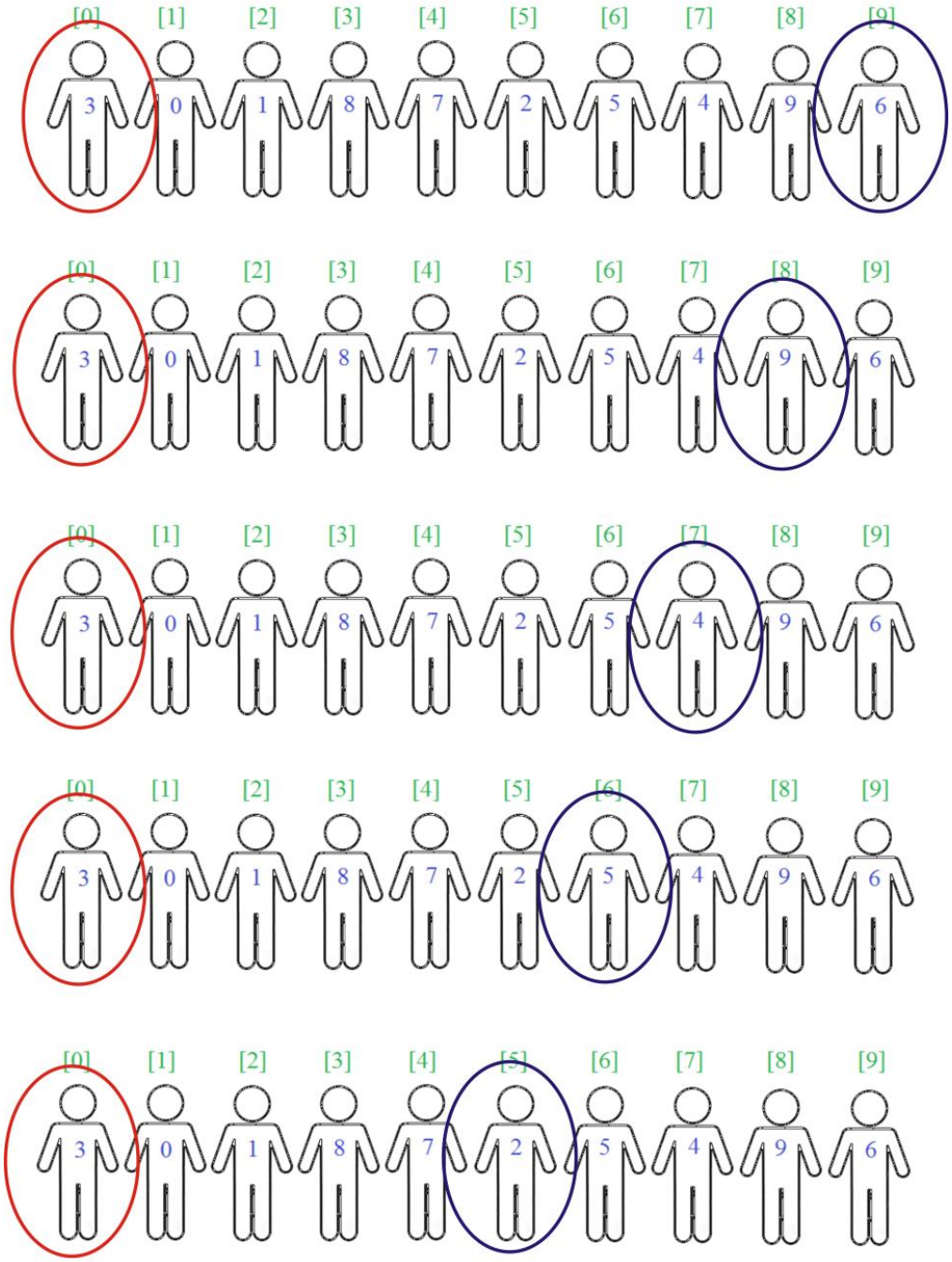


Figura 72. Se observan las primeras cinco comparaciones. Cada una de éstas muestra dos óvalos, el rojo encierra al bailarín pivote y el azul al bailarín que se compara con el pivote en cada ejecución.

La partición se divide en dos intervalos el primero comprendido por los bailarines 2, 0, 1, 8 y 7, y el segundo por los bailarines 5, 4, 9 y 6.

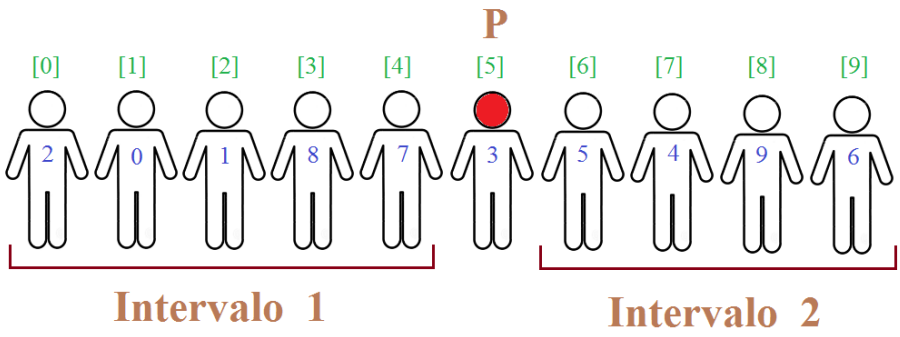


Figura 73. Partición del arreglo.

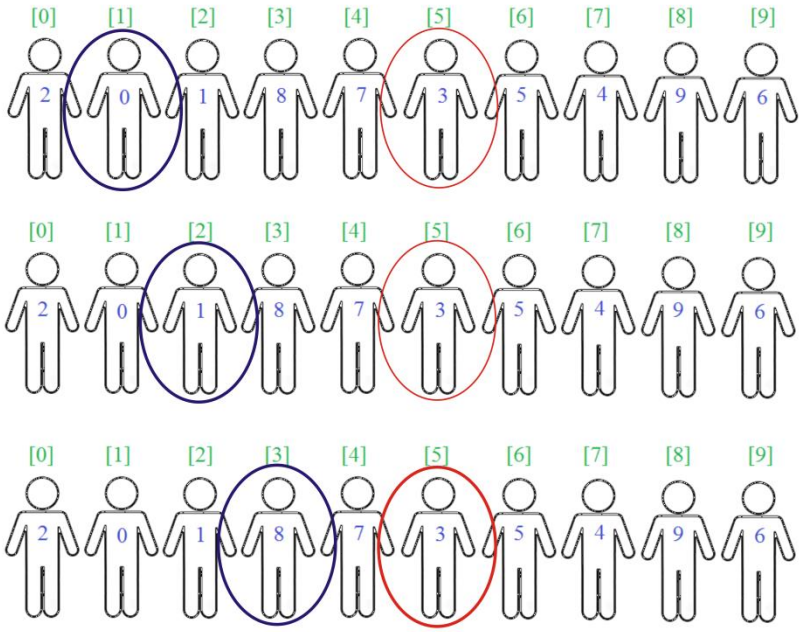


Figura 74. Observan las tres comparaciones siguientes.

Ahora el bailarín pivote se encuentra ubicado en la posición cinco. Este desplazamiento favorece al máximo la velocidad del algoritmo, puesto que, reordena los bailarines de cada intervalo de forma tal que los bailarines que aparecen antes del pivote sean menores a éste, y los que aparecen después del pivote sean mayores a éste.

Por ejemplo: realizando la comparación entre el bailarín número 8 y el bailarín número 3 (ver figura 74), se obtiene

como resultado un intercambio de posiciones, logrando que todos los bailarines que se encuentran antes del pivote sean menores a éste.

Nótese que para completar las comparaciones entre el bailarín número 3 y cada uno de los bailarines del arreglo es necesario comparar a éste con el bailarín número 7. En consecuencia varían los intervalos de la figura 73.

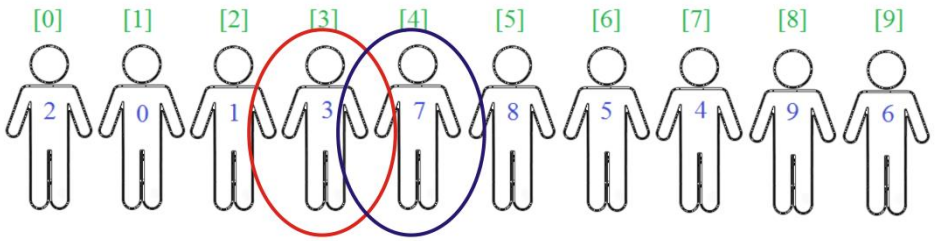


Figura 75. Comparación entre el bailarín número 3 y el bailarín número 7.

Tan pronto como se haya hecho todas las comparaciones entre el pivote y los demás bailarines, se selecciona un nuevo pivote para el intervalo de números menores.

Igual que en el procedimiento anterior, se escoge al bailarín número 2 como pivote, de ahí que, éste se compara con el bailarín número 1 y con el bailarín número 0.

Es preciso señalar que, sólo en la primera comparación los bailarines intercambian sus posiciones, ya que el bailarín pivote se encuentra ubicado en una posición menor que el bailarín número 1.

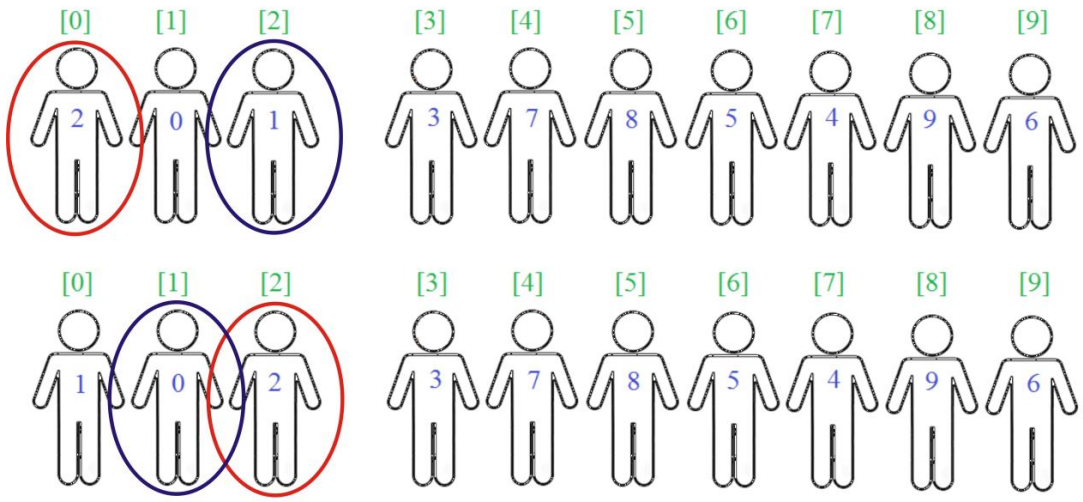


Figura 76. En la segunda comparación, los bailarines no cambian de posición, debido a que el bailarín número 2 ocupa una posición mayor al bailarín número 0.

Realizadas las posibles comparaciones entre el bailarín pivote número 2 y los dos bailarines restantes, se puede afirmar que sólo faltaría llevar a cabo una sola comparación entre el bailarín número 1 como pivote y el bailarín número 0.

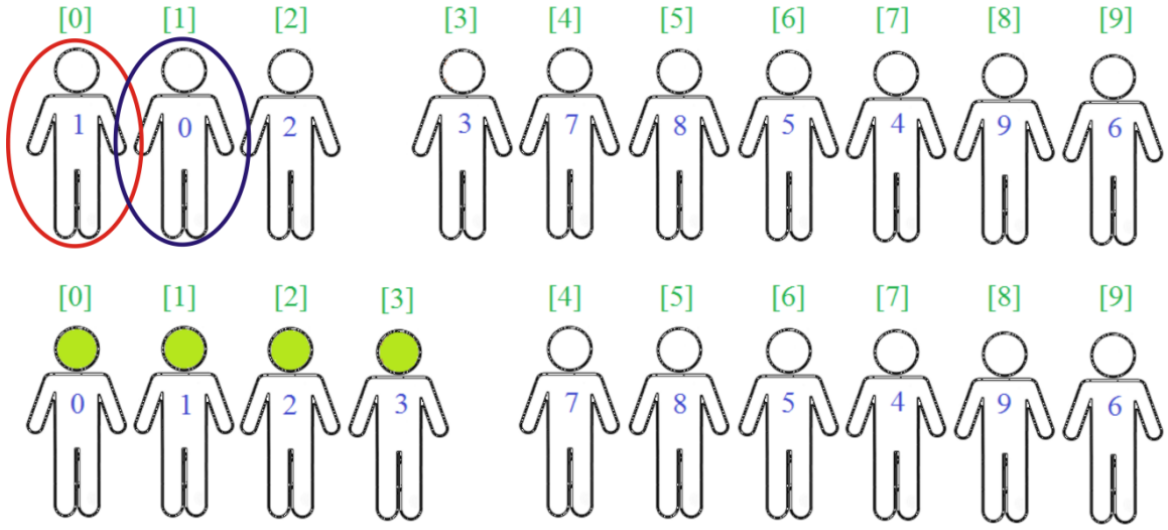


Figura 77. Con la intención de seguir el mismo patrón de selección respecto al pivote se selecciona al bailarín número 1. De ahí que luego se compara con el bailarín número 0.

La figura 77 muestra los primeros cuatro bailarines ordenados de menor a mayor seguidos por un intervalo con seis bailarines. Seguidamente, se escoge al bailarín número 7 como pivote y se empieza a comparar con cada uno de los bailarines restantes, teniendo en cuenta que si el bailarín tiene un número menor que el del pivote de inmediato invierten su posición.

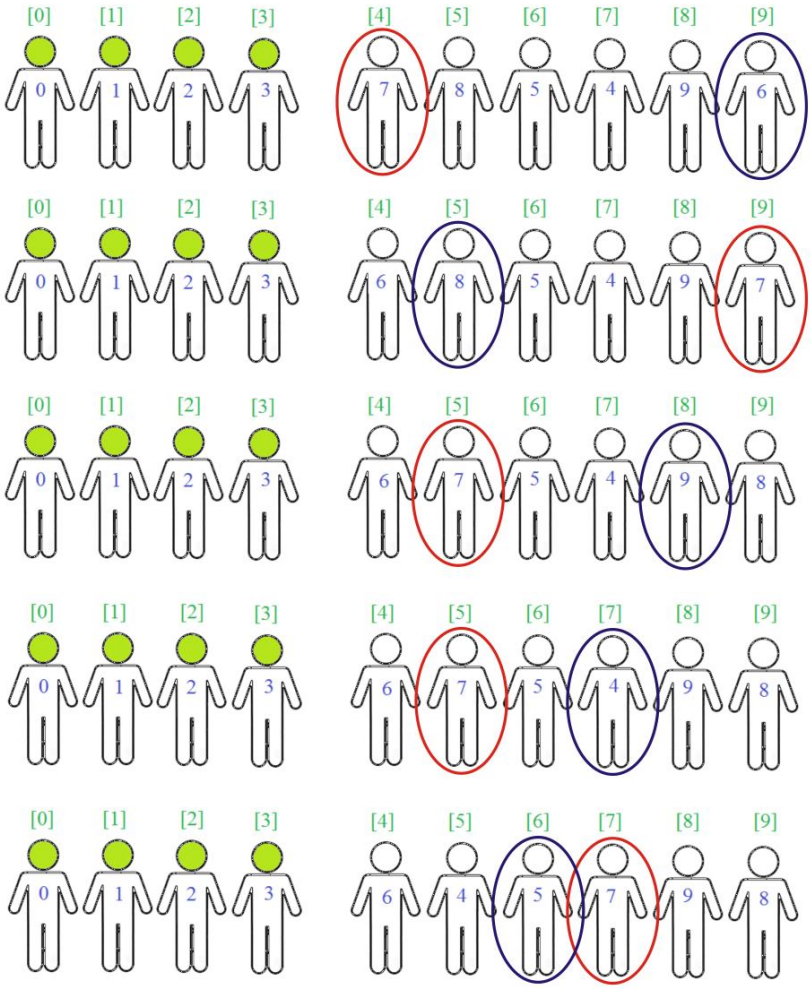


Figura 78. El bailarín número 7 realizó cinco comparaciones, siempre con respecto al número más lejano que no haya sido comparado previamente por el mismo bailarín pivote

En la secuencia que muestra la figura 78 se observan dos cambios de posición. El primero se llevó a cabo entre el bailarín número 7 y el bailarín número 6 y el segundo entre el bailarín número 7 y el bailarín número 4, en consecuencia los bailarines con números menores al del bailarín pivote, quedan ubicados en la parte izquierda, de forma tal que la partición se pueda dividir en tres intervalos.

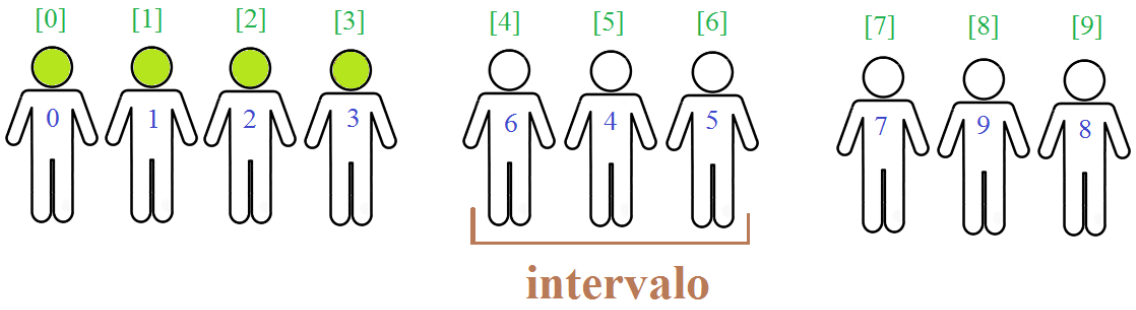


Figura 79. Se observa el nuevo intervalo conformado por los bailarines número 6, 4 y 5.

Para este nuevo intervalo, se selecciona al bailarín número 6 como pivote, de ahí que, el resultado de la comparación respecto al bailarín número 5 conlleve a que éstos dos inviertan la posición en el arreglo.

En la siguiente ejecución, son comparados los bailarines 6 y 4. En consecuencia con el resultado, estos conservan su posición en el arreglo. De ello resulta necesario decir que las posiciones de los bailarines 4 y 5 se encuentran invertidas, por lo tanto se divide de nuevo el arreglo como se muestra en la figura 80.

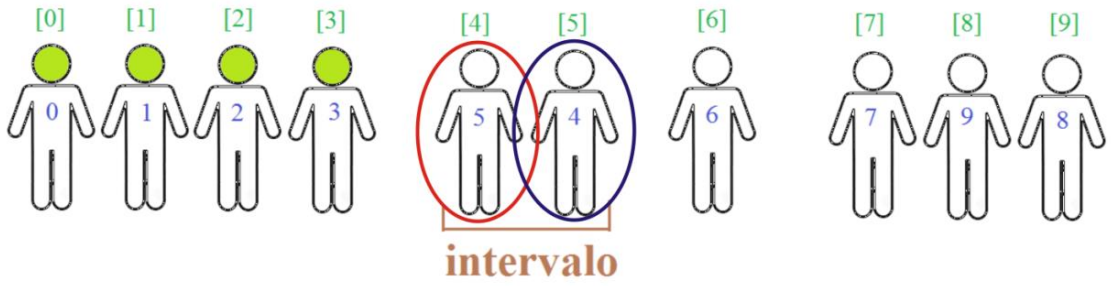


Figura 80. Se comparan los bailarines 5 y 4 con el fin de ordenar las primeras ocho posiciones del arreglo.

Una vez dividido el arreglo se crea un nuevo intervalo integrado solo por dos bailarines el número 4 y el número 5. Siguiendo el proceso anterior, se escoge como bailarín pivote al número 5, enseguida se comparan y se intercambian de posición.

Como resultado de estas ejecuciones se consiguen ordenar de menor a mayor los primeros ocho bailarines.

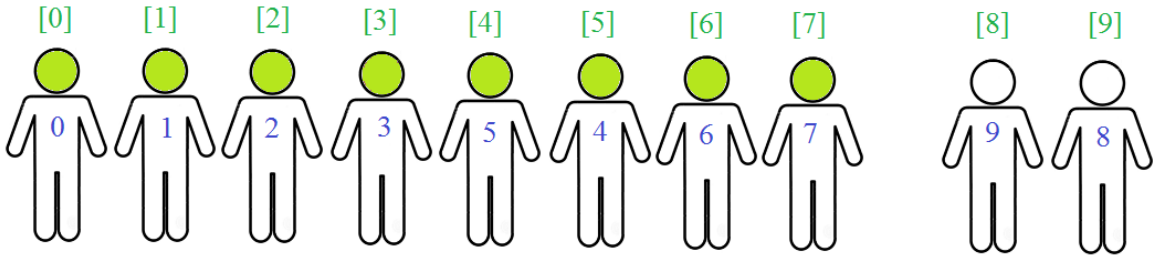


Figura 81. Con las veinte ejecuciones anteriores se ha conseguido ordenar los bailarines desde la posición cero hasta la posición siete.

Con el fin de concluir la aplicación del algoritmo, se realiza la última comparación eligiendo como pivote al bailarín número 9, consecuentemente, éste cambia de posición con respecto al bailarín número 8.

Finalmente, se logra ordenar ascendentemente el arreglo con los diez bailarines.

El segundo ejemplo describe una situación, en la cual se necesita ordenar ocho objetos de menor a mayor teniendo en cuenta el nombre de cada uno de ellos.

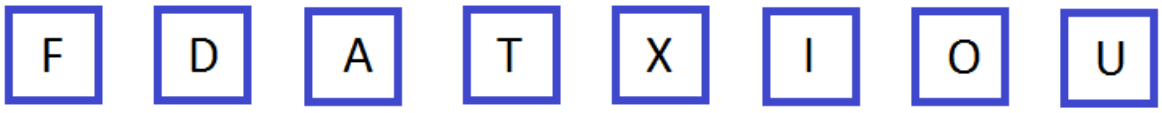


Figura 82. Los ocho cuadrados que se observan contienen la letra inicial del nombre de cada objeto. Asimismo, están ubicados aleatoriamente.

El primer paso consiste en seleccionar el pivote.

En este caso se seleccionó la casilla con la letra T como se evidencia en la figura 83.

Dado que la fila de letras esta desordenada, se podría elegir la primera letra como pivote de forma semejante que en el primer ejercicio. Sin embargo, para este ejercicio se seleccionó la letra que estaba aproximadamente en la mitad con la intención de ilustrar diferentes estrategias ya que es improbable que un pivote al azar de una partición mala en una lista de elementos desordenados.

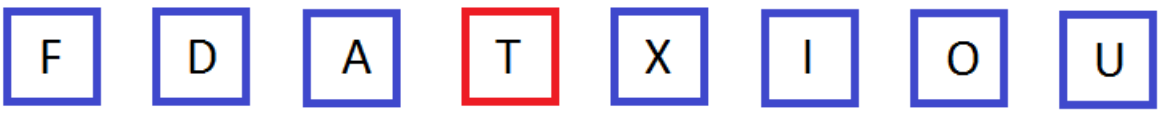


Figura 83. Luego de elegir a la letra pivote se compara con cada una de las letras.

En el segundo paso se ubican los objetos que están en la parte derecha e izquierda de la letra T, teniendo en cuenta algún sistema de ponderación. Esto quiere decir, que si se

elige como sistema de ponderación la inicial del nombre de cada uno de los objetos que conforman la lista, las letras que son menores a T serán ubicadas a la izquierda del pivote. En consecuencia, las demás letras quedarán ubicadas a la derecha.

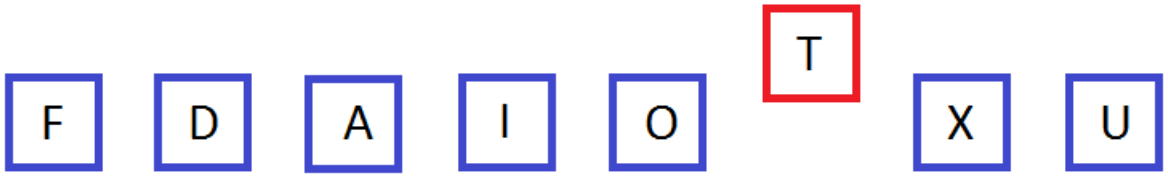


Figura 84. Se observa que la letra F, D, I, A y O fueron ubicadas a la izquierda de la letra T en una ejecución diferente. Las acciones anteriores llevan implícitamente la comparación entre el pivote y las demás letras con base en el abecedario.

En el tercer paso, se vuelve a seleccionar una nueva letra como pivote, utilizando el mismo principio de selección anterior, siempre y cuando haya más de dos letras en cualquiera de los lados divididos por el pivote que se seleccionó en el paso anterior.

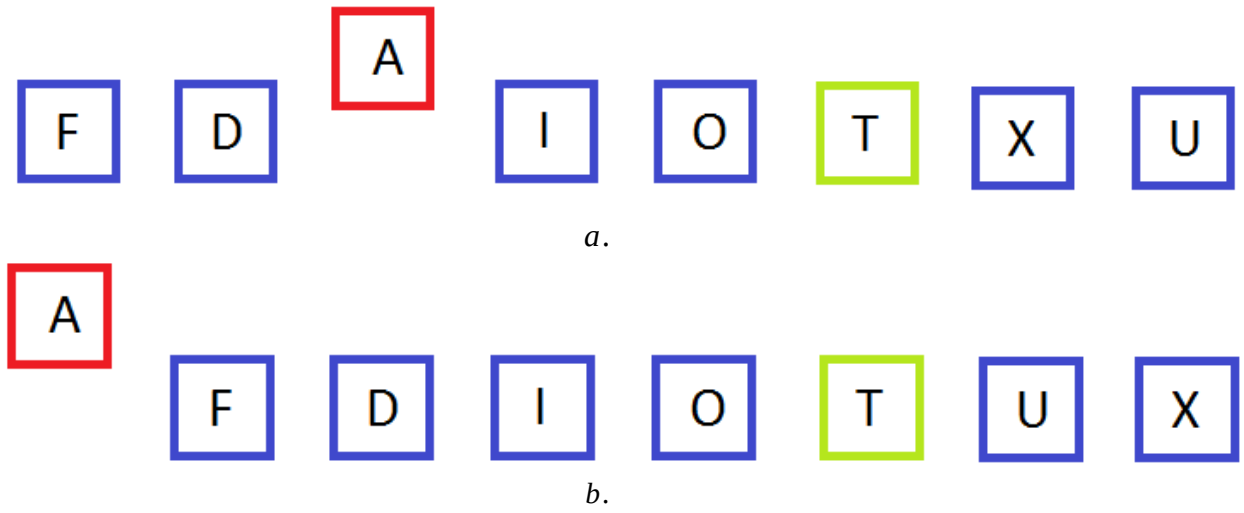


Figura 85. En el figura *a* se observa la letra A como pivote, ésta se compara con las letras de su parte izquierda y se ubica en la primera posición. Por otro

lado, la figura b muestra la letra A ubicada en la primera posición, después de compararse con las letras F y D.

Luego de seleccionar la letra A como pivote se vuelve a aplicar el paso número dos. Obviamente, cada desplazamiento se realiza después de comparar las dos letras en cada turno.

Como se observa en la figura 85 entre las letras A y T se encuentran cuatro letras lo cual conlleva a seleccionar una letra de estas como pivote. Para este caso se seleccionó la casilla con la letra I. No obstante, se hubiera podido seleccionar la casilla con la letra D. (Véase figura 85)

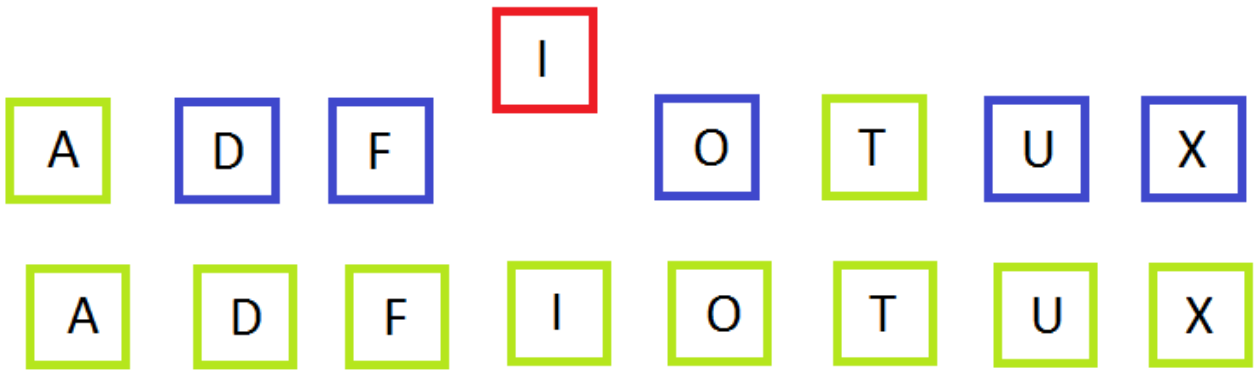


Figura 86. La letra I se compara con las letras que están en su parte izquierda de tal manera que queden organizadas alfabéticamente como se observa en la fila de letras inferior.

Finalmente, este algoritmo de ordenación rápido permite ordenar de menor a mayor o viceversa, de manera eficiente y eficaz una lista de objetos con base en algún sistema de ponderación.

4.2.7 Ordenación por incrementos – Shellsort

Este algoritmo es una generalización del algoritmo de ordenación por inserción, teniendo en cuenta que éste mejora la ordenación por inserción comparando elementos separados por un espacio o intervalo cerrado de varias posiciones. Por consiguiente, esto permite que un elemento se desplace pasos más grandes hacia la posición correcta. (EcuRed, 2012).

Los desplazamientos múltiples sobre los datos se hacen con tamaños de espacio o intervalos cerrados cada vez más pequeños. Al estar leyendo el arreglo de elementos utiliza un solo tipo de comparación abstracta, es así, como se determina cual elemento va primero, cual le sigue y quien va al final del arreglo. De ahí que resulta fácil deshacer más de una inversión en cada intercambio, hecho por el cual este algoritmo gana velocidad.

En el proceso se compara los elementos con saltos de mayor tamaño, pero con incrementos decrecientes.

El último paso de este algoritmo es un simple ordenamiento por inserción.

A fin de comprender el funcionamiento de este algoritmo se considera un arreglo integrado por 10 bailarines, los cuales están ubicados aleatoriamente.

La descripción de este algoritmo se basa en el video llamado: Shell-sort with Hungarian (Székely) folk dance – Link: https://www.youtube.com/watch?v=CmPA7zE8mx0&ab_channel=AlgoRythmics

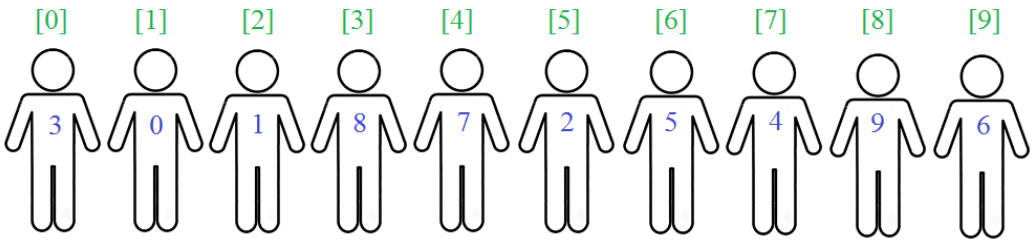


Figura 87. Se observa el arreglo inicial con los diez bailarines ubicados aleatoriamente.

En primer lugar se dividen los bailarines en dos grupos con la condición de que cada grupo tengan igual cantidad de bailarines. El primer grupo está conformado por los bailarines 3, 0, 1, 8 y 7. De igual manera el segundo grupo lo integran los bailarines 2, 5, 4, 9 y 6. (Véase figura 88).

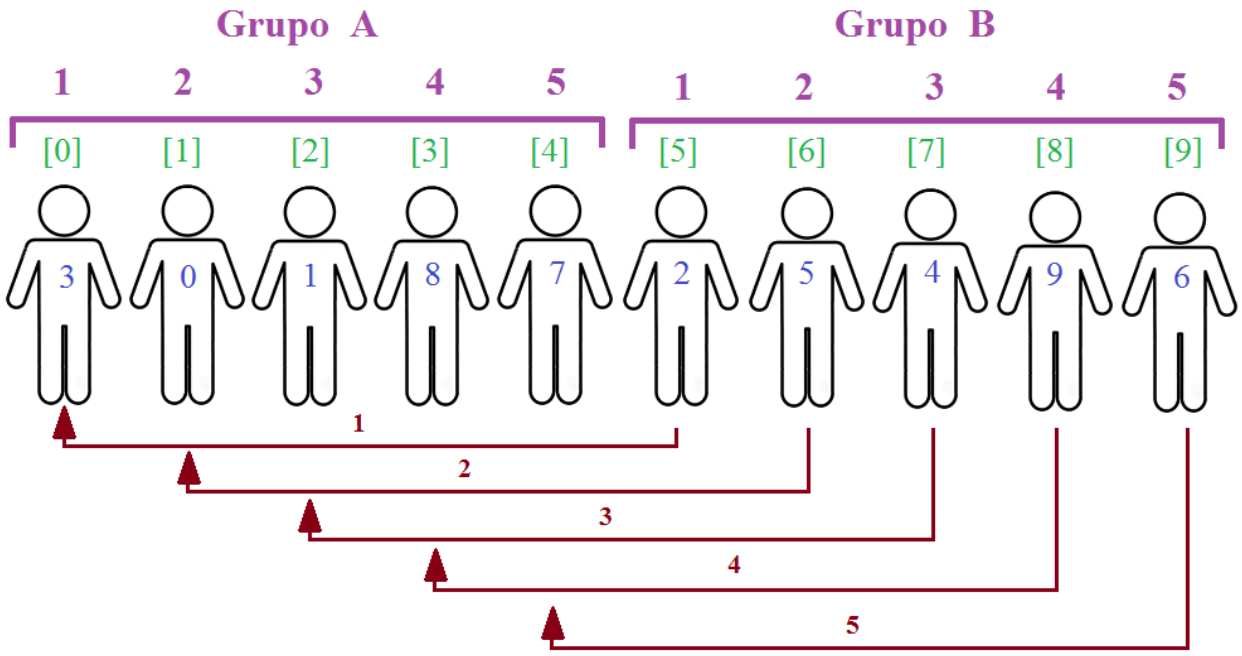


Figura 88. Se observan las primeras cinco ejecuciones. La intención de cada ejecución es comparar al bailarín que ocupa igual posición en los dos grupos. Lo anterior significa que en la primera ejecución se compara al bailarín número 2 respecto al bailarín número 3, como evidencia la flecha 1 de la parte inferior. Nótese que los dos bailarines mencionados anteriormente ocupan la misma posición en su grupo (Los números en la parte superior color purpura representan la posición de los bailarines en cada uno de los grupos).

Una vez conformado los dos grupos, se procede a comparar a los dos bailarines que ocupan el primer puesto en su grupo. Esto quiere decir, que en la primera ejecución se comparan a los bailarines 2 y 3, en la segunda ejecución a los bailarines 5 y 0, en la tercera ejecución a los bailarines 4 y 1; y así sucesivamente como indican las flechas en la parte inferior de la figura 88.

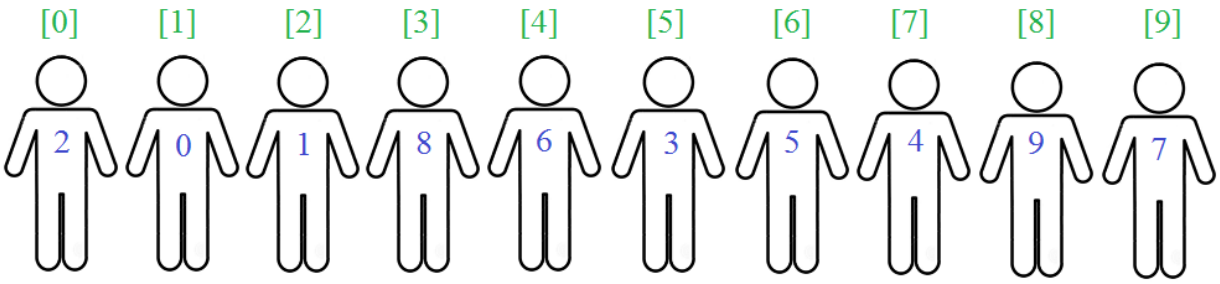


Figura 89. Muestra el arreglo resultante luego de realizar las cinco primeras comparaciones.

Tan pronto se realizan las comparaciones, el bailarín 3 con el bailarín 2 y el bailarín 7 con el bailarín 6 invierten de posición. Esta acción se da debido, a que el número 3 y el número 7 ocupan una posición inferior con respecto al número 2 y al número 6.

De lo anterior resulta necesario decir que siempre que se comparen dos bailarines, éstos invertirán su posición si y sólo si el número mayor ocupa una posición inferior al número menor en el arreglo.

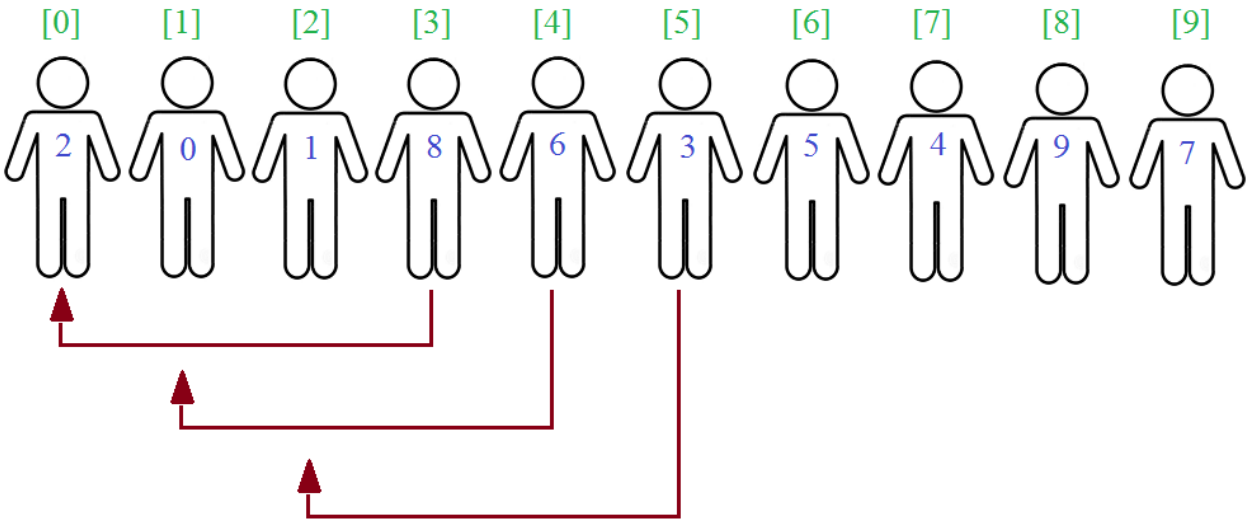


Figura 90. Se visualizan las tres primeras comparaciones. Obsérvese que ahora sólo se encuentran dos bailarines en medio de los que se van a comparan en cada turno.

En segundo lugar, se reduce a la mitad el intervalo cerrado formado por los bailarines que se encuentran en medio de los dos bailarines que se van a comparar.

Si se observa la figura 88 en medio de los dos bailarines que se comparan en las primeras cinco ejecuciones se encuentran 4 bailarines. Por ejemplo: cuando se comparó al bailarín número 3 y al bailarín número 2, en medio de éstos, estaban los bailarines 0, 1, 8 y 7, estos cuatro bailarines representan un intervalo cerrado. Por esta razón, ahora el intervalo cerrado sólo deberá contener 2 bailarines en medio de cada comparación. De ahí que se establezcan las comparaciones evidenciadas en la figura 90.

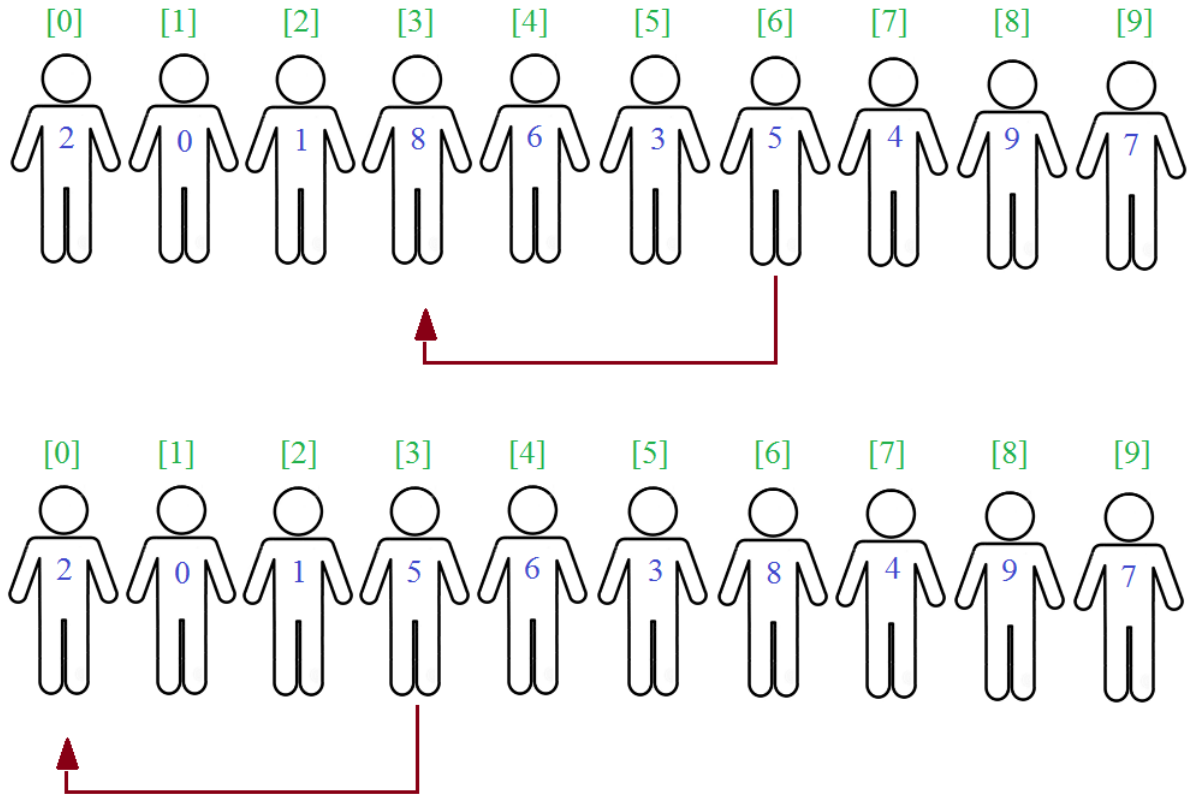


Figura 91. Se realiza la comparación entre los bailarines 5 y 2. Como el número 2 es menor que el número 5, éstos permanecen en la misma posición del arreglo.

Seguidamente se compara al bailarín número 5 con el bailarín número 8, como el número 8 es mayor que el número 5 y además el bailarín número 8 ocupa una posición inferior a la que ocupa el bailarín número 5, éstos cambian de posición. En la parte inferior de la figura 91, se evidencia que el bailarín número 5 se ubica ahora en la posición tres del arreglo y se compara con el bailarín número 2.

Es importante destacar que cada vez que se invierta la posición de dos bailarines en un turno, el que ocupe la posición inferior, se seguirá comparando con el bailarín que le corresponda, hasta que no haya intercambio de posición entre éstos.

Un ejemplo de lo mencionado anteriormente, son las figuras 91 y 92, en la primera se evidencia que el bailarín número 5 se compara con el bailarín número 8, éstos intercambian de posición. Luego, se realiza la comparación entre los bailarines 5 y 2. Dado que 2 es menor que 5 y que el bailarín número 2 está ubicado en una posición inferior a la del bailarín número 5, éstos permanecen en la misma posición en el arreglo.

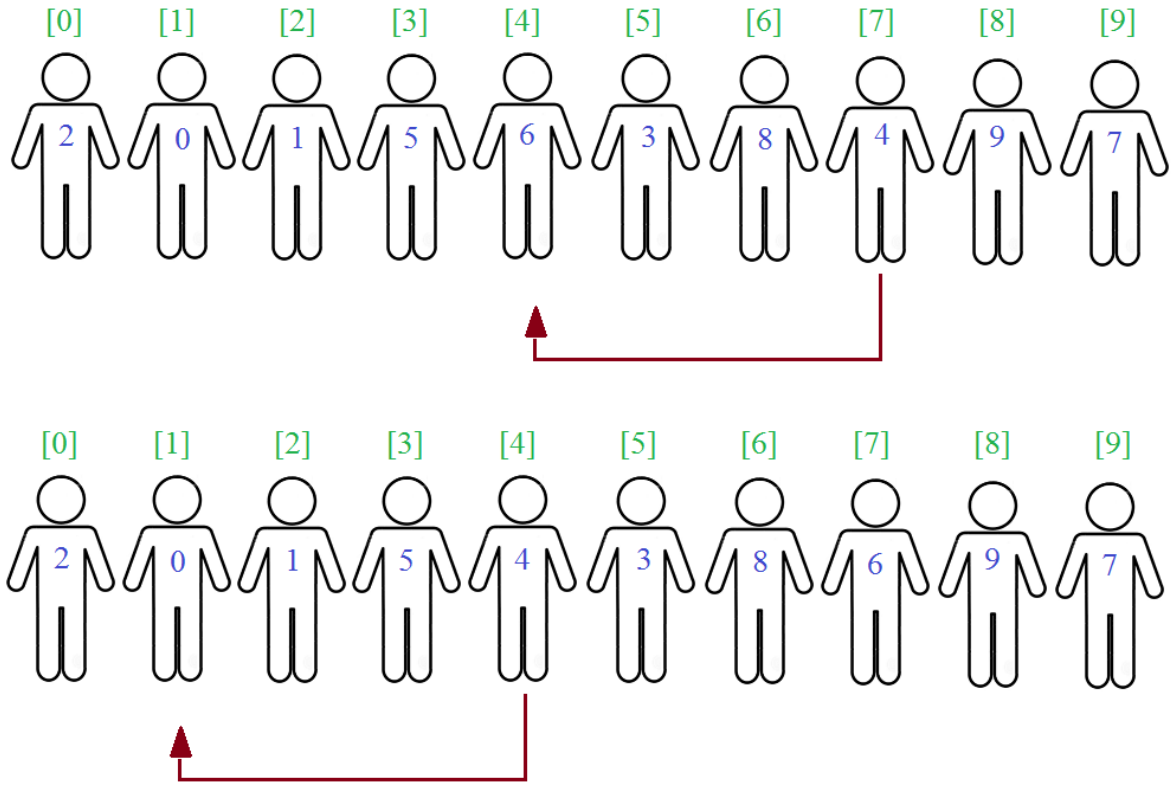


Figura 92. Al igual que en la figura anterior, en ésta se ilustra la comparación que hay entre el bailarín número 4 y el bailarín número 6. Debido a que existe el intercambio de posición entre éstos, el bailarín número cuatro se compara con el bailarín número 0.

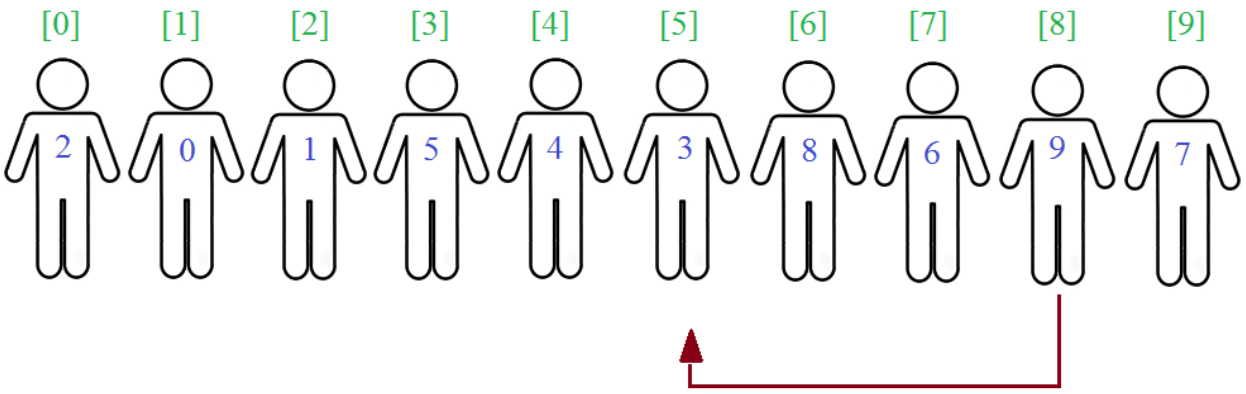


Figura 93. Se realiza la comparación entre los bailarines 9 y 3. Como el número 3 es menor que el número 9, éstos permanecen en la misma posición del arreglo.

De forma similar se comparan los bailarines 9 y 3, como no se cumplen las dos condiciones para que entre ellos intercambien de ubicación, éstos permanecen en la misma posición del arreglo. A diferencia de la comparación anterior, los bailarines 7 y 8 si cambian de posición. El bailarín número 7 se ubica en la posición seis del arreglo. Esto conlleva a que éste, sea comparado con el bailarín número 5.

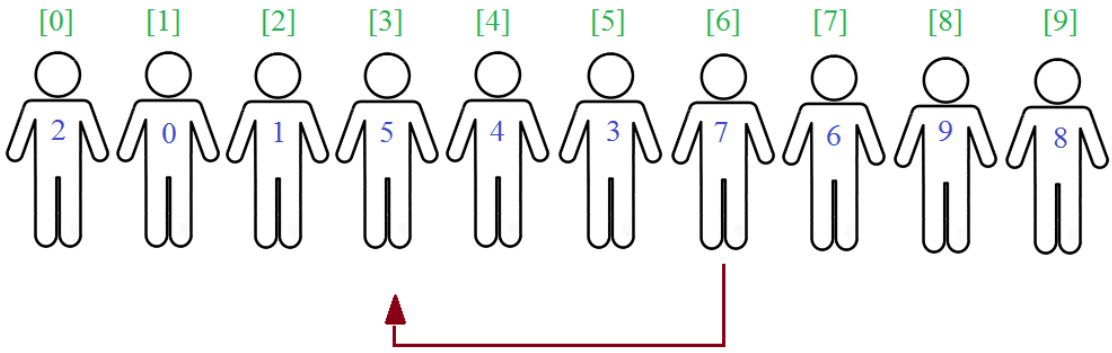


Figura 94. Se realiza la comparación entre los bailarines 7 y 5. Como el número 5 es menor que el número 7, éstos permanecen en la misma posición del arreglo.

Finalmente, se aplica la misma metodología que en el algoritmo de ordenación por inserción, la cual se fundamenta en la

comparación de los bailarines adyacentes. A continuación se ilustra, nuevamente la metodología. Como se visualiza en la figura 95 se inicia comparando los bailarines 2 y 0. Estos dos bailarines invierten su posición siempre que el número mayor ocupe una posición menor con respecto al número menor.

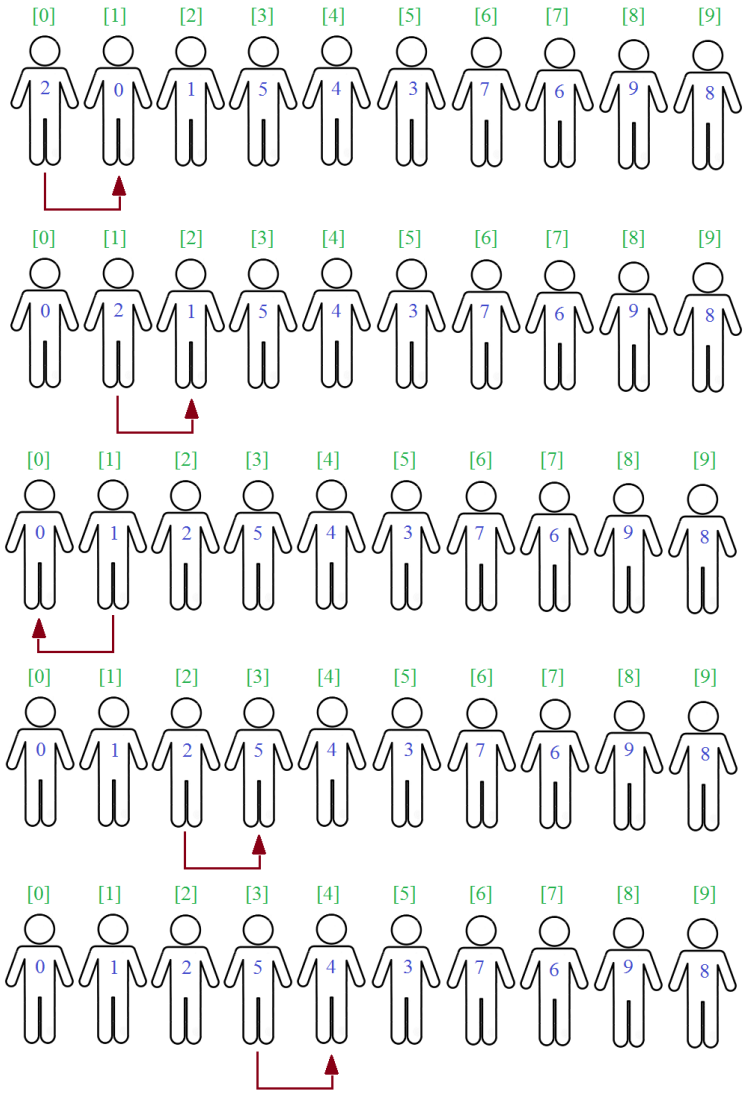


Figura 95. Se muestran las primeras cinco comparaciones realizadas a los bailarines adyacentes.

Luego el bailarín número 2 se compara con el bailarín número 1, esta acción conlleva a que inviertan su posición. En

efecto, el bailarín número 1 se ubica en la posición uno del arreglo.

Después de eso, el bailarín número 1 se compara con el bailarín número 0, con el objetivo de ubicar en la posición definitiva al bailarín número 1.

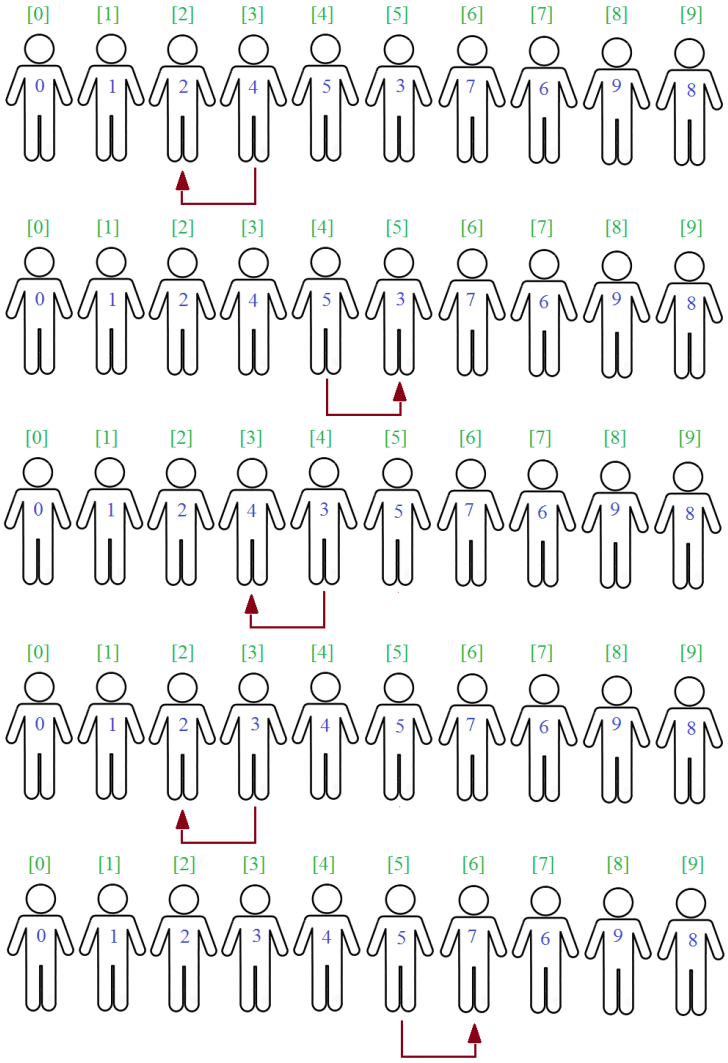


Figura 96. Se visualizan cinco comparaciones, en la primera cuarta y quinta los bailarines conservan la misma posición en el arreglo. En cambio en la segunda y tercera el bailarín número 3 cambia de posición en el arreglo dos veces.

Una de las comparaciones que muestra la figura 96 se da entre los bailarines 5 y 3, como el número 3 es menor que el número 5, el bailarín número 3 se ubica en la posición cuatro del arreglo.

En el siguiente turno el bailarín número 3 se compara y cambia de posición con el bailarín número 4.

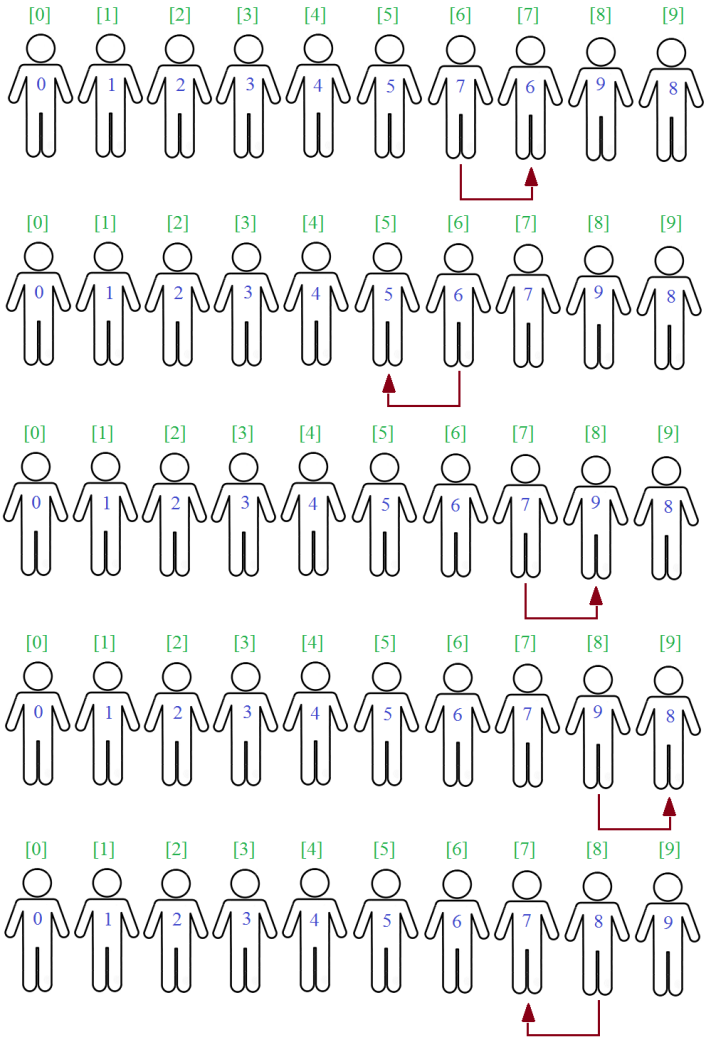


Figura 97. Esta figura muestra el desplazamiento que realizó el bailarín número 6 después de compararse con los bailarines 7 y 5. Asimismo el intercambio de posición entre los bailarines 9 y 8.

El bailarín número 3 es comparado con el bailarín número 2, conservando estos dos la posición en el arreglo.

Es de esta manera como se consigue ordenar los seis primeros bailarines.

En la figura 97 se observan las últimas cinco comparaciones. Del mismo modo que antes, se ejecutan las iteraciones con el objetivo de ubicar a los tres últimos bailarines, teniendo en cuenta las condiciones mencionadas en los párrafos anteriores.

En conclusión, entre más aumente la cantidad de objetos que se deban ordenar mayor es la cantidad de iteraciones que es necesario realizar, en comparación con el algoritmo de ordenación burbuja, éste reduce la cantidad de iteraciones al dividir en mitad e ir ordenando.

5 Conclusiones

Los ejercicios presentes en este documento, describen estrategias y métodos de resolución que permiten desarrollar capacidades relativas al pensamiento computacional. De ahí que esto permita que la persona adquiera habilidades y destrezas que en un momento determinado van a ser útiles.

Comprendiendo que el pensamiento computacional va mucho más allá que la programación de computadores y que adoptar este pensamiento beneficia a todas las profesiones, no se hizo hincapié en un lenguaje de programación especifico, sino se planteó un desarrollo general que ilustra de forma amigable la aplicación de algunos conceptos tales como la recursividad, las iteraciones, la lógica y algunas técnicas algorítmicas para la resolución de problemas.

Los algoritmos de búsqueda y ordenación son una forma de pensar computacionalmente que propicia el análisis y la relación de ideas para la búsqueda u organización de objetos. Entender su funcionamiento conlleva a que la persona aprenda a descomponer un problema, encontrar patrones o similitudes y generar soluciones creativas.

6 Referencias bibliográficas

Algoritmo de Ordenamiento Shell (14 de diciembre 2020) EcuRed enciclopedia colaborativa en red del gobierno de Cuba Link https://www.ecured.cu/Algoritmo_de_Ordenamiento_Shell

Algoritmos de ordenación y búsqueda, (s.f). Obtenido de: http://biolab.uspceu.com/aotero/recursos/docencia/TEMA%208.pdf Universidad Cardenal Herrera – España.

Artecona, F., Bonetti, E., Darino, C., Mello, F., Rosá, M. Scópise, M. (2017). *Pensamiento Computacional*. Un aporte para la educación de hoy. Editorial Fundación Telefónica, Movistar. Fuente https://issuu.com/eduticpe/docs/pensamientocomputacionalfte

Azcue, M., Diez, M. L., Lucanera, V., Scandroli, N., (2002). *Resolución de un problema complejo utilizando un elemento de naturaleza heurística.* Facultad de Ciencias Veterinarias de la Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina. Número 37/6, 10-2-06.

Caipa, S. C., Torres, E, W. (2016, 9 de mayo). Metodología POLYA en resolución de problemas. Departamento de Matemáticas Gimnasio Campestre. Colombia.- Palabra Maestra Fuente https://www.compartirpalabramaestra.org/academia/alianza-gimnasio-campestre-compartir/metodologia-polya-en-resolucion-de-problemas

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., Woollard, J. (2015). *Pensamiento Computacional. Guía para profesores*. Computing At School. Link http://www.codemas.org/wp-content/uploads/2016/04/ Pensamiento-computacional-Guía-para-profesores.pdf

Salvador, P. (2016, 6 de febrero). *LA ABSTRACCIÓN EN FOTOGRAFÍA*. Fuente https://www.pedrosalvador.es/2016/02/06/la-abstracci%C3%B3n-en-fotograf%C3%ADa/#:~:text=La%20esencia%20de%20la%20abstracci%C3%B3n,una%20abstracci%C3%B3n%20de%20la%20forma.

Zapata-Ros, M. (2015, 15 de septiembre). *Pensamiento computacional: una nueva alfabetización digital*. RED, Revista de Educación a Distancia. DOI: 10.6018/red/46/4 Fuente http://www.um.es/ead/red/46/zapata.pdf