**Laboratorio 7**

Ejercicio 1

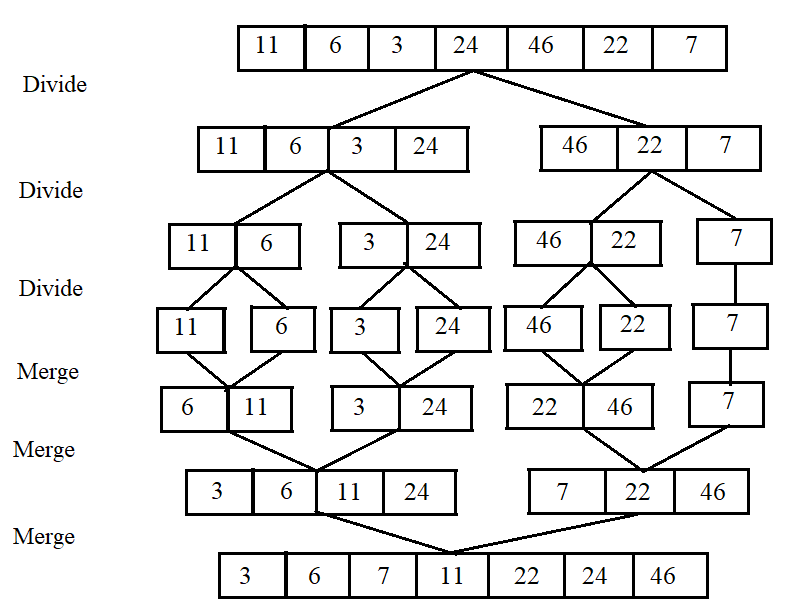
Descargue el código fuente del portal para el laboratorio 7. Implemente el método sort\_elements() de la clase Tuples. La clase contiene un arreglo de Tuple llamado “elements”. Un tuple es una pareja de valores int (first, last). Su método de ordenamiento (cualquiera que aplique) debe ordenar el arreglo “elements” de forma ascendente. Un Tuple se compara con otro a través de la suma de sus valore first y last.

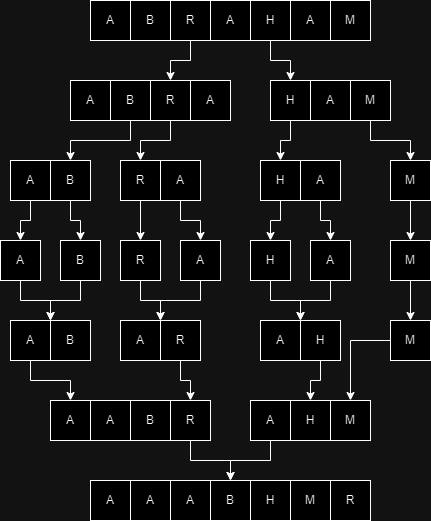
Ejercicio 2

Usted tiene un conjunto de datos con los siguientes valores:

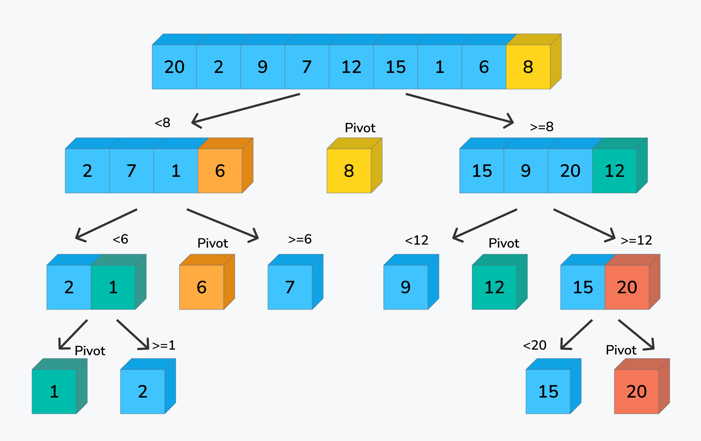
“A” “B” “R” “A” “H” “A” “M”

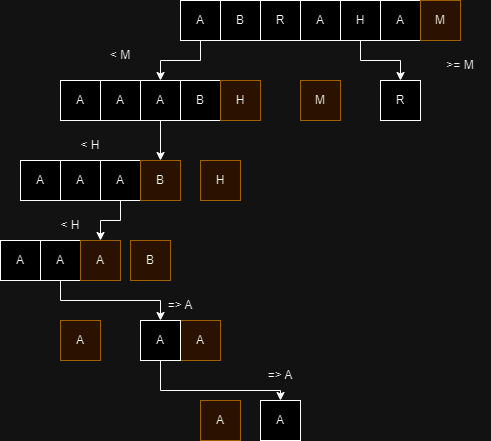
Construya todo el árbol de iteraciones de todos los algoritmos siguientes: Quick Sort, Merge Sort, pasando por todas las fases de cada algoritmo. Sírvase del ejemplo de cada algoritmo. Puede usar <https://draw.io> para realizar los diagramas:

Merge Sort  




Quick Sort





Ejercicio 3

Usted tiene un conjunto de datos con los siguientes valores:

“A” “B” “R” “A” “H” “A” “M”

Construya toda la tabla de iteraciones del algoritmo Insertion Sort para dichos datos. Sirvase del ejemplo y utilice el mismo formato:

Datos de ejemplo:

f b a h c d g e

Iteraciones:

f b a h c d g e

b f a h c d g e

a b f h c d g e

a b f h c d g e

a b c f h d g e

a b c d f h g e

a b c d f g h e

a b c d e f g h

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | B | R | A | H | A | M |
| **A** | **B** | R | A | H | A | M |
| **A** | **B** | **R** | A | H | A | M |
| **A** | **A** | **B** | **R** | H | A | M |
| **A** | **A** | **B** | **H** | **R** | A | M |
| **A** | **A** | **A** | **B** | **H** | **R** | M |
| **A** | **A** | **A** | **B** | **H** | **M** | **R** |

Ejercicio 4

Responda

¿En qué condición QuickSort puede tener una complejidad temporal en términos de Big-O igual a n2? De un ejemplo que cause dicha condición.

**Si la elección del pivote no es óptima y siempre selecciona el elemento más grande o pequeño.**

**Un ejemplo de un arreglo que cause esta condición es un arreglo que ya está ordenado en orden ascendente o descendente. Si QuickSort siempre elige el primer elemento como pivote y el arreglo está ordenado en orden ascendente, el pivote será siempre el elemento más pequeño, lo que provocará divisiones desequilibradas y llevará a una complejidad temporal cuadrática.**

Considere que todos los algoritmos han sido implementados adecuadamente, y que los datos no tienen ningún orden o semi orden aplicado. Bajo qué condición seria cierto que estos tiempos son realmente los que le ha tomado a cada algoritmo ordenar un conjunto de datos N.

QuickSort: 5 milisegundos

InsertionSort: 3 milisegundos

BubbleSort: 3 milisegundos

Bajo la condición que la complejidad de tiempo haya sido *O*(*N*2), ya que tanto Insertion como Bubble tienen la característica de funcionar muy bien con conjuntos de datos pequeños, mientras que Quick es su peor caso, lo cual explicaría esos 2 milisegundos extra.