Universidad Rafael Landívar
Programación Avanzada

Laboratorio 7

Ejercicio 1

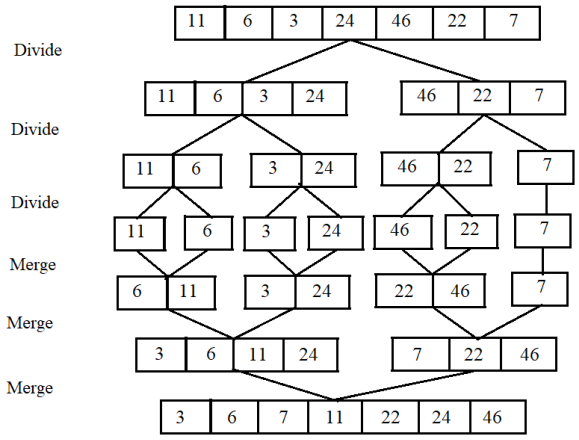
Descargue el código fuente del portal para el laboratorio 7. Implemente el método sort_elements() de la clase Tuples. La clase contiene un arreglo de Tuple llamado “elements”. Un tuple es una pareja de valores int (first, last). Su método de ordenamiento (cualquiera que aplique) debe ordenar el arreglo “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:

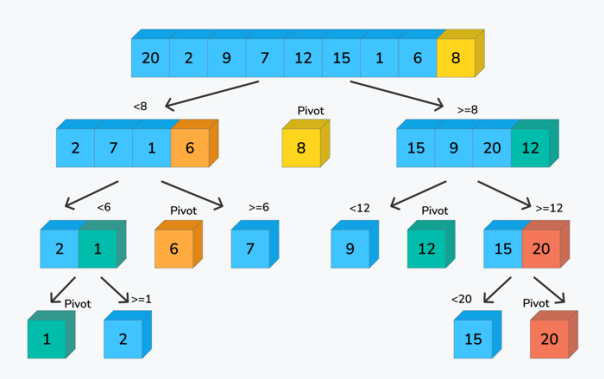
“A” “B” “R” “A” “H” “A” “M”

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:

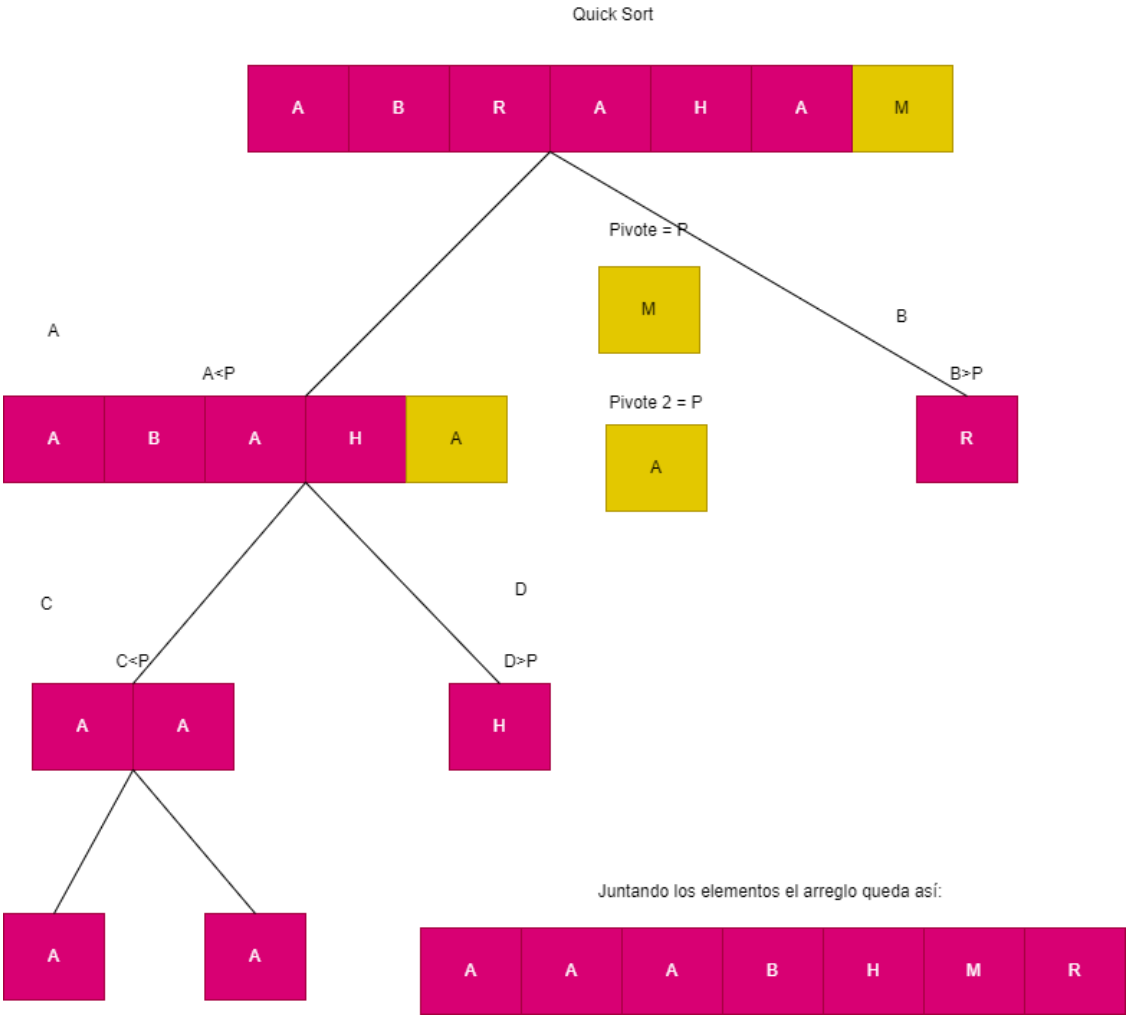
Merge Sort

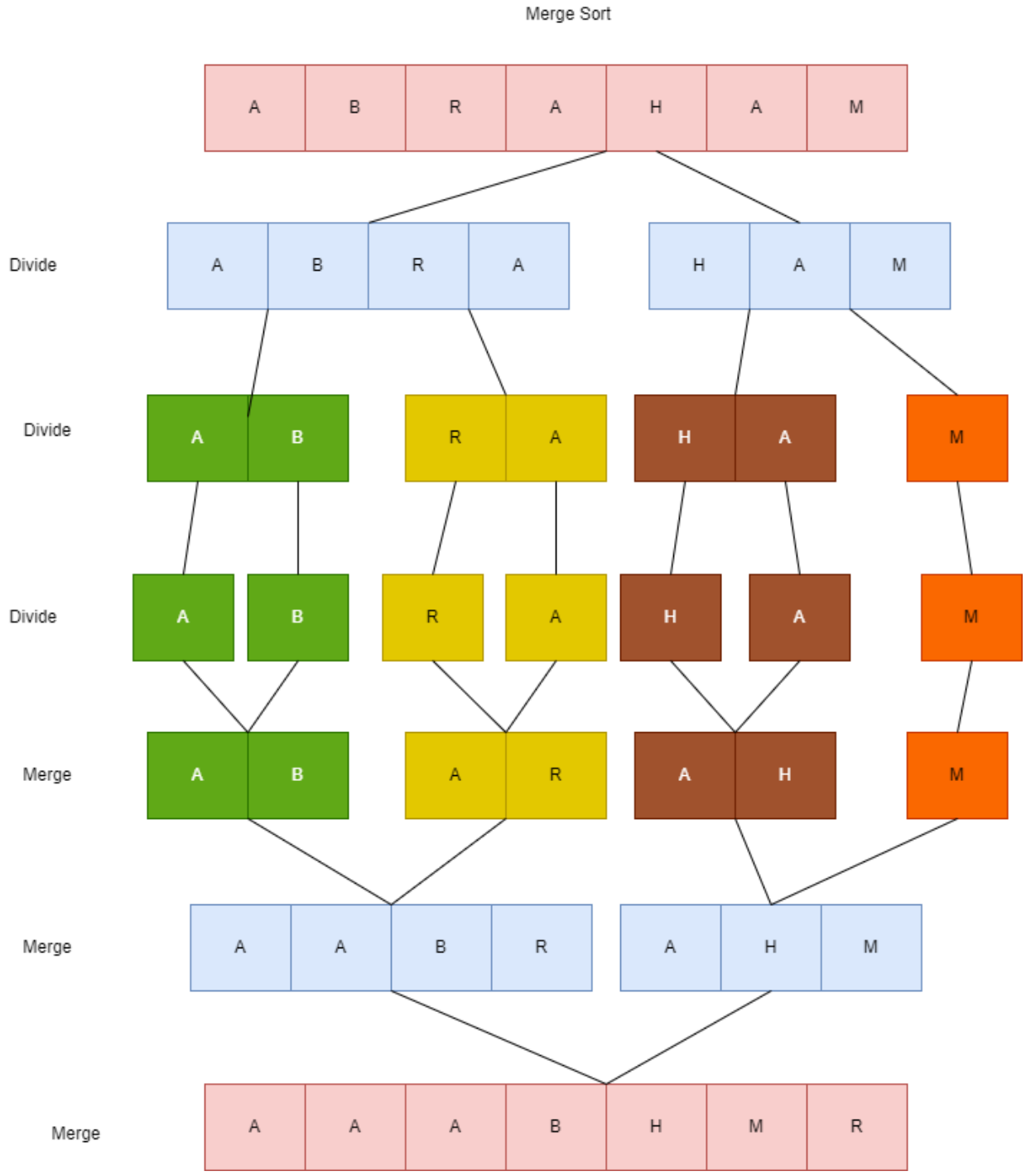


Quick Sort



Respuesta:


Quick Sort
A B R A H A M
Pivote = P
M
A
B
A<P
B>P
A B A H A
Pivote 2 = P
A
R
C
D
C<P
D>P
A A
H
A
A
Juntando los elementos el arreglo queda así:
A A A B H M R



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
```

Respuesta:

Insertion Sort

No.	A	B	R	A	H	A	M
1	A	B	R	A	H	A	M
2	A	A	B	R	H	A	M
3	A	A	B	H	R	A	M
4	A	A	A	B	H	R	M
5	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 n^2? De un ejemplo que cause dicha condición.

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

Respuesta: QuickSort generalmente tiene un rendimiento muy eficiente en términos de tiempo, con una complejidad promedio de O(n log n). Sin embargo, puede degradarse a O(n^2) en el peor de los casos. Esto sucede cuando QuickSort elige un pivote desfavorable en cada iteración y se divide en particiones de tamaño muy desigual.

Un ejemplo que causa esta condición sería si los datos de entrada ya están ordenados o inversamente ordenados, y el pivote elegido en cada iteración es el elemento más pequeño o más grande en la partición actual. Esto hará que el algoritmo divida la secuencia en dos partes de tamaño desigual, y en cada iteración, una de las partes tendrá n-1 elementos y la otra tendrá solo un elemento.

Por ejemplo, supongamos que tenemos los siguientes datos:

5, 4, 3, 2, 1

Si QuickSort elige el primer elemento (5) como pivote en cada iteración (Ya que se puede elegir si quiere escoger al primero, el de en medio o el ultimo como pivote), y la partición resultante coloca todos los demás elementos en un lado, obtendríamos el siguiente proceso:

Paso 1: 1, 4, 3, 2, 5
Paso 2: 1, 2, 3, 4, 5

En cada paso, solo se ordena un elemento, y el algoritmo requerirá O(n^2) operaciones para ordenar la lista en este caso particular.

En el caso de identificar si todos los algoritmos han sido implementados se puede identificar y suponer que Quick Sort e insertion Sort son algoritmos que funcionan bien mientras que Bubble Sort al ser un algoritmo de ordenamiento simple pero no muy eficiente por lo que el tiempo de 3 milisegundos que se proporciono es razonable, pero BubbleSort puede requerir más tiempo en comparación con QuickSort o InsertionSort.