Instituto Tecnológico de Costa Rica
Escuela de Computación
Ingeniería en Computación
IC-2001 Estructuras de datos
Profesor: M.Sc. Carlos Benavides, Ing.
Estudiantes: Diego Cheng Gómez y Justin Bogantes Rodriguez

Introducción

En este trabajo se abarco el tema de algoritmos de ordenamiento los cuales tenían la función de ordenar la información de distintos tipos de TDA. dichos algoritmos tienen diversas funciones en el campo laboral tanto en el área de la matemática y la ciencia de la computación. Estos algoritmos se pueden dividir en naturales y no naturales

Comparación entre algoritmo.

Bubbler Sort con una lista simple de 100000 elementos tipo int duro aproximadamente 2 minutos y 6 segundos haciendo el ordenamiento

HeapSort con una lista simple de 100000 elementos tipo int duro aproximadamente 4 minutos y 15 segundos

```
20:29:56: Starting C:\Users\Justin\Documents\GitHub\build-Prog
Desktop_Qt_5_11_1_MinGW_32bit-Debug\debug\Progra_1_Datos_Bena.
20:30:17
20:34:32
20:34:39: C:/Users/Justin/Documents/GitHub/build-Progra_1_Dato
Desktop_Qt_5_11_1_MinGW_32bit-Debug/debug/Progra_1_Datos_Bena
```

InsertSort con una lista simple de 100000 elementos tipo int duro aproximadamente 4 minutos y 15 segundos

```
20:40:22: Starting C:\Users\Justin\Documents\GitHub\build-Progra_1_Datos_Bena-
Desktop_Qt_5_11_1_MinGW_32bit-Debug\debug\Progra_1_Datos_Bena...
20:40:44
20:44:59
```

Quicksort con una lista simple de 100000 elementos tipo int duro aproximadamente 4 minutos y 15 segundos

```
20:49:39: Starting C:\Users\Justin\Documents\GitHub\build-Progra_1_Datos_Bena-
Desktop_Qt_5_11_1_MinGW_32bit-Debug\debug\Progra_1_Datos_Bena...
20:50:01
20:51:06
```

Merge sort con una lista simple de 100000 elementos tipo int duro aproximadamente 4 minutos y 15 segundos ya que su complejidad es a la de heapSort

Selection sort con una lista simple de 100000 elementos tipo int duro aproximadamente 2 minutos y 6 segundos haciendo el ordenamiento

Complejidad computacional

Bubblesort	$O(n^2)$
Insertion sort	$O(n^2)$("(en el peor de los casos)")
Bucket sort	O(n)
Merge sort	O(n log n)
Radix sort	O(nk)
Shell sort	O(n1.25)
Selection sort	$O(n^2)$
Heapsort	O(n log n)
Quicksort	Promedio: O(n log n), peor caso: $O(n^2)$

Conclusiones.

Hay ciertos algoritmos que no funcionan con ciertos tipos de datos como el bucket sort el cual se trato de implementar sin embargo no se pudo debido a que la manera en que este mismo funciona es con tipos de datos flotantes donde guarda el valor flotante donde en el bucket correspondiente al ser números enteros hubo ciertos conflictos debido a que dicho algoritmo esta pensado para flotantes el mismo caso con string
notamos que al trabajar con templates es un poco mas complicado debido a que en ocasiones hay error a la hora de implementar dicha estructuras . a la hora de realizar las pruebas de dichos algoritmos se noto que por la capacidad de hardware que tendrá una computadora ahí dependerá el tiempo de ejecución conforme a los ordenamientos que realizara ya que al comprobar 1000000 elementos en cierta maquina el tiempo de duración no iba a ser el mismo en otra maquina