UNIVERSIDAD ADOLFO IBAÑEZ
FACULTAD DE INGENIERÍA Y CIENCIAS

Tarea 2
TICS311: Estructura de Datos y Algoritmos

Fecha de Entrega: 26 de Mayo 2023, 23:59 hrs.

El objetivo de esta tarea es profundizar en el lenguaje C y en algoritmos de ordenamiento.

Instrucciones

1. La tarea se puede hacer en grupos de a lo más 3 personas. Debe respetar los grupos inscritos en la planilla de Webcursos.
2. Debe subir a Webcursos un archivo `experimentos.c` para la parte 1 y un archivo `main.c` para la parte 2. No suba ningún otro archivo `.c`.
3. Junto con los archivos `.c`, **debe** subir un informe indicando el compilador que utilizó y explicando brevemente la lógica detrás de sus códigos en ambas partes. El informe debe indicar también los resultados y análisis de la parte 1.
4. Se aceptan atrasos. Se descontará 1.0 pts por cada día de atraso.
5. Notar que hay un punto extra, luego la nota máxima es un 8.0.

Parte 1 (3.0 pts)

En esta parte, su misión es analizar empíricamente la complejidad de los 5 algoritmos de ordenamiento vistos en clases (BubbleSort, SelectionSort, InsertionSort, MergeSort, QuickSort). Para esto deberá hacer lo siguiente:

- Genere 6 arreglos aleatorios de largo 100, 1000, 10000, 100000, 200000 y 500000. Para la generación de números aleatorios puede usar la función `rand()` de la librería `<stdlib.h>`. Para más información puede ir a este link.
- Usando los códigos dados en clases, pruebe cada uno de los 5 algoritmos sobre los 6 arreglos generados y mida el tiempo de ejecución de cada uno de los algoritmos. Para medir el tiempo de manera precisa utilice la función `clock()` de la librería `<time.h>`. Para un ejemplo de uso, puede ver este link.

Reporte y analice sus resultados. ¿Qué puede observar de los tiempos de ejecución?

Observaciones:

- En instancias grandes, algunos de los algoritmos pueden tomar algo de tiempo (entre 2 a 15 minutos approx).
- Debe implementar sus experimentos en la función main del archivo `experimentos.c`.

Parte 2 (4.0 pts)

Suponga que queremos manipular datos de usuarios, los cuales guardaremos en el siguiente struct:

```
typedef struct u {
   char nombre[100];
   int edad;
}
```

Cada usuario tiene un nombre (string de largo a lo más 100) y edad. Implemente un programa en C que haga lo siguiente:

- Reciba por línea de comando el nombre de un archivo (el argumento `argv[1]`). Este archivo tendrá en cada línea un usuario escrito en formato `nombre,edad`. A modo de ejemplo, puede mirar el archivo `usuarios.txt`.
- Su programa debe almacenar la información del archivo en un arreglo de structs de tipo usuario.
- Debe imprimir en pantalla los usuarios ordenados según el siguiente criterio: primero se ordena por edad (de menor a mayor), y en caso de empate, se ordena por el nombre (de menor a mayor lexicográficamente).
- Para ordenar su arreglo, **deben** adaptar el algoritmo MergSort o QuickSort (cualquiera de los dos, ustedes escogen) para recibir un arreglo de usuarios. La firma de estas funciones se verá así:

  ```
  void merge_sort(usuario *arreglo, int N);
  void quick_sort(usuario *arreglo, int N);
  ```

Observaciones:

- Para comparar strings puede usar la función `strcmp` de `<string.h>`.
- La función `strtok` de `<string.h>` puede ser útil para procesar cada línea del archivo.
- Cada línea del archivo debe pasarse a un struct usuario. A la hora de copiar el nombre al campo `nombre` del struct usuario, le puede servir usar la función `strcpy` de `<string.h>`.
- Debe implementar su programa en la función main del archivo `main.c`.