Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Estructuras de Datos y Algoritmos I

Trabajo Práctico 1

Integrantes:

Cipullo, Inés

Palumbo, Matías

Universidad Nacional de Rosario

2020

A continuación, se detallan las elecciones en general y particularidades de la resolución del Trabajo Práctico 1 en cuanto a las estructuras de datos elegidas, compilación de los programas y resultados obtenidos.

1 Elección de estructura de datos

La estructuras de datos utilizadas a lo largo del programa son listas generales, y decidimos implementarlas como listas doblemente enlazadas circulares. Esta elección se basó principalmente en la capacidad de poder recorrer la lista en ambos sentidos (al tener punteros al elemento anterior y al siguiente en cada nodo) y en la facilidad y rapidez con la que se puede acceder al último elemento de la lista sin necesidad de una estructura auxiliar (ya que al ser circulares el nodo anterior al comienzo de la lista es el último elemento).

Por ejemplo, estas características de las listas elegidas se aprovecharon en la implementación del algoritmo Insertion Sort y Merge Sort, aumentando significativamente la rapidez.

2 Compilación de los programas

La estructuración de los archivos y sus dependencias se encuentra detallada en el archivo *makefile*. Para compilar el programa se utiliza el comando `make all` o `make`. A su vez, luego de la compilación, los ejecutables del primer y segundo programa se corren mediante los siguientes comandos:

`./programa1 ARCH_NOMBRES ARCH_LUGARES ARCH_SALIDA CANT_PERSONAS`

`./programa2 ARCH_SALIDA`,

donde `ARCH_NOMBRES` y `ARCH_LUGARES` son los archivos con todos los nombres y lugares de nacimiento, respectivamente, y `ARCH_SALIDA` es el archivo donde se escribirán las `CANT_PERSONAS` generadas; `programa1` crea un archivo con la lista de personas y `programa2` crea seis archivos (uno por cada algoritmo por cada función comparadora) con los tiempos de ejecución y las listas ordenadas.

2.1 Caracteres especiales

Los caracteres especiales presentes en los archivos de entrada de los programas son las vocales acentuadas y la letra Ñ. Al almacenar los nombres y lugares como strings convencionales, el espacio requerido por estos caracteres especiales es mayor al espacio de un char, por lo que ocupan el espacio de dos chars en el string. Sin embargo, este desborde en la memoria de un char hace que el número devuelto al castear el char como entero sea negativo en ambos chars ocupados por el caracter especial, y, si son analizados por separado, no representan nigún valor significativo. Este comportamiento no trae problemas en la lectura y escritura de archivos.

En cuanto a las funciones comparadoras elegidas, como se realizó la comparación de las edades y los nombres de las personas únicamente, no es necesario manipular caracter por caracter a los strings con lugares (en esta manipulación surge lo comentado en el párrafo anterior). Por otro lado, teniendo en cuenta que de los caracteres especiales, la lista de nombres posee solo letras Ñ mayúscula, y por tanto el desborde en la memoria del char surge sólo en la presencia de estas letras, se hace uso de esto en la comparación de nombres. Concretamente, comparando caracteres, si el número entero asociado al caracter es negativo, se asume que se trata de una letra Ñ y como la misma ocupa el espacio de dos chars, se ignora el caracter siguiente.

3 Comparación de resultados

Para comparar diferentes atributos de las personas se hizo uso de las funciones comparadoras `comp_edades` y `comp_nombres`, las cuales comparan las edades (en forma ascendente) y nombres (alfabéticamente) de cada persona, respectivamente. A continuación, se presenta una tabla bivariada de los tiempos promedio de diez ejecuciones según los tres algoritmos y dos atributos a comparar, con un archivo de veinte mil personas, así como también dos tablas con tiempos promedio de ejecución de los algoritmos con respecto a cada una de las funciones comparadoras.

Tabla bivariada con tiempos promedio de ejecución de algoritmos según atributo comparado (20000 personas)

Atributo comparado	Selection Sort	Insertion Sort	Merge Sort
Edad	1.9270779s	1.6108322s	0.007852s
Nombre	7.2298428s	4.0083017s	0.0327217s

Tiempos promedio de ejecución de los algoritmos comparando edades

Cantidad de Personas	Selection Sort comparando Edades	Insertion Sort comparando Edades	Merge Sort comparando Edades
5000	0.0975377s	0.0805441s	0.00168s
10000	0.4078023s	0.3472982s	0.00443s
20000	1.9270779s	1.6108322s	0.007852s
30000	5.3687142s	4.724104s	0.013881s

Tiempos promedio de ejecución de los algoritmos comparando nombres

Cantidad de Personas	Selection Sort comparando Nombres	Insertion Sort comparando Nombres	Merge Sort comparando Nombres
5000	0.4176615s	0.2181018s	0.0064452s
10000	1.6911357s	0.8867611s	0.0152668s
20000	7.2298428s	4.0083017s	0.0327217s
30000	18.1892948s	11.9598137s	0.0516609s

Analizando los datos, se tiene que Selection Sort es el algoritmo con el mayor tiempo de ejecución, seguido de Insertion Sort y por último, con un tiempo considerablemente menor, Merge Sort. Estos tiempos reflejan el hecho de que Selection Sort recorre los elementos de la lista la misma cantidad de veces independientemente del orden inicial de la lista (es decir, el mejor y peor caso son iguales), y, en contraposición, la cantidad de operaciones que realiza Insertion Sort puede ser menor según el orden inicial de la lista. Por otro lado, como Merge Sort es una función recursiva y por tanto su ejecución requiere de una mayor cantidad de memoria que los otros dos algoritmos, la cantidad de operaciones que realiza (y consecuentemente el tiempo de ejecución) es mucho menor.

Como acotación, es relevante destacar que los tiempos en general de los algoritmos cuando comparan nombres son mayores a cuando comparan edades debido a que la función `comp_nombres` realiza una mayor cantidad de operaciones que `comp_edades`, lo que se refleja en mayor tiempo de ejecución con cantidades grandes.

4 Bibliografía

- `https://en.wikipedia.org/wiki/Insertion_sort`
- `https://en.wikipedia.org/wiki/Selection_sort`
- `https://en.wikipedia.org/wiki/Merge_sort`
- `https://www.tutorialspoint.com/c_standard_library/c_function_clock.htm`
- `https://theasciicode.com.ar`