Análisis de Algoritmos 2018/2019

Práctica 1

Nicolás Serrano Salas y Miguel Luque López, 1212.

Código	Gráficas	Memoria	Total

1. Introducción

En esta práctica exploraremos de forma experimental el primero de nuestros algoritmos de ordenación y comprobaremos de forma experimental su eficiencia además de que comprobaremos otras tantas características del mismo, esto se hará con el objetivo de asentar y entender las bases impartidas en las clases teóricas.

Para poder hacer esto necesitaremos crear una función de aleatoriedad, que nos de valores para desordenar los valores dentro de cada tabla y otra que llame a esta para usando estos valores mover los elementos de la tabla y una tercera función que permute estos valores un numero determinado de veces, además de esto se nos pedirá crear el ya mencionado algoritmo SelectSort, por último se implementará un sistema para medir el tiempo que tarda la ejecución del algoritmo para cada caso y se añadirá un segundo algoritmo, SelectSortInv y terminaremos haciendo tablas para guardar los resultados experimentales descubiertos.

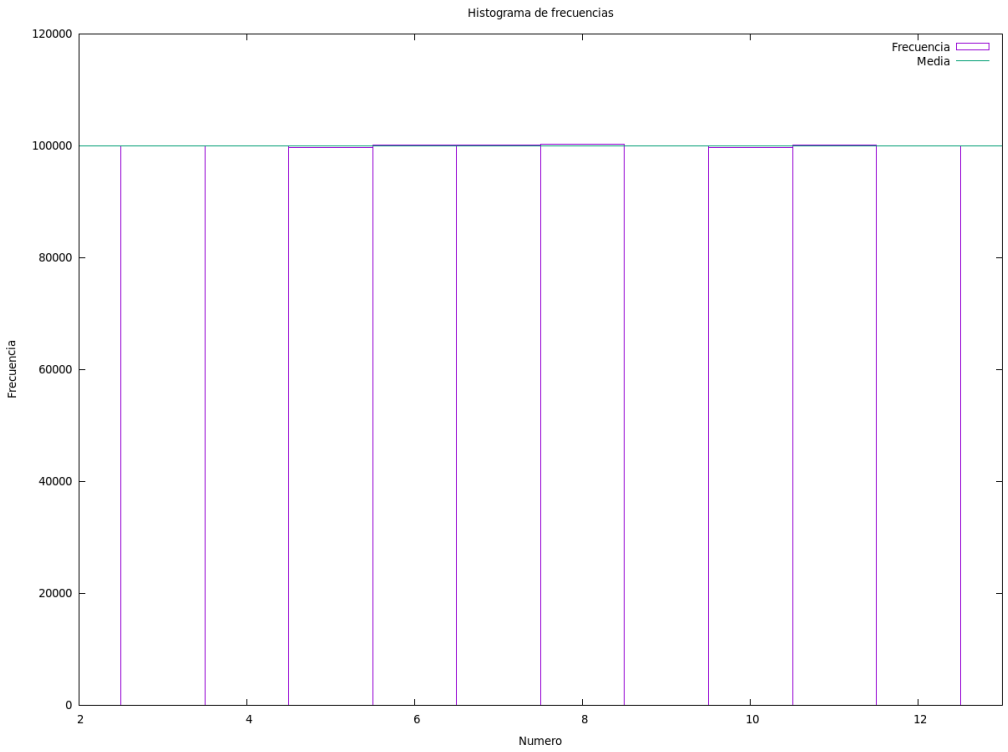
2. Código

3. Resultados, Gráficas

Aquí ponéis los resultados obtenidos en cada apartado, incluyendo las posibles gráficas.

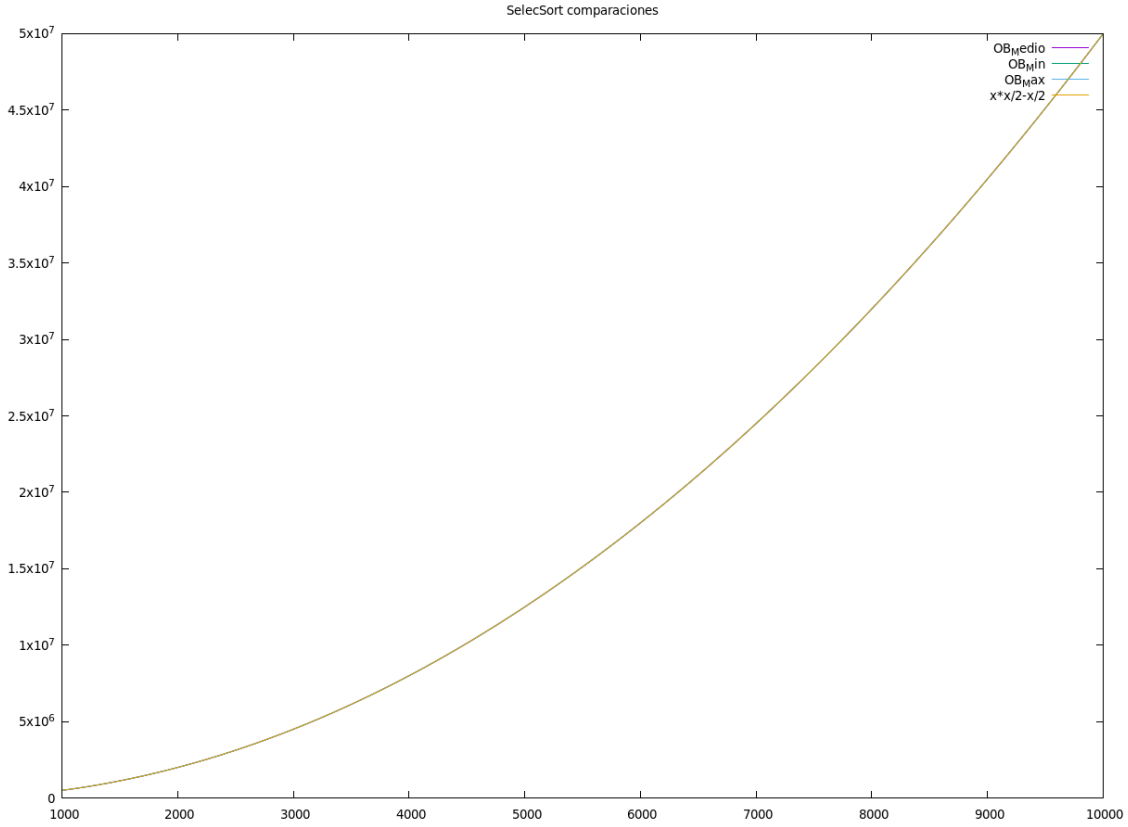
3.1 Apartado 1

> Gráfica del histograma de números aleatorios, en esta gráfica vemos en el eje horizontal los números resultantes y en el eje vertical el número de veces que han aparecido, esto ejecutando nuestro algoritmo de aleatoriedad

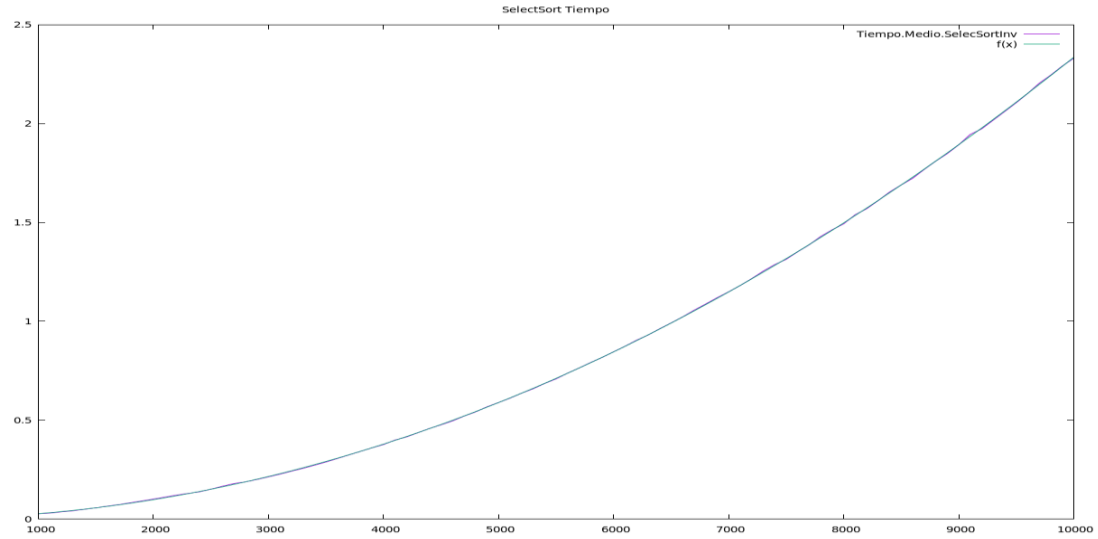


3.2 Apartado 2

Gráfica comparando los tiempos mejor peor y medio en OBs para SelectSort
Podemos comprobar por los resultados adquiridos que los tiempos de ejecución de SelectSort serán iguales para la misma longitud de tabla, por muy ordenada o desordenada que esta esté.

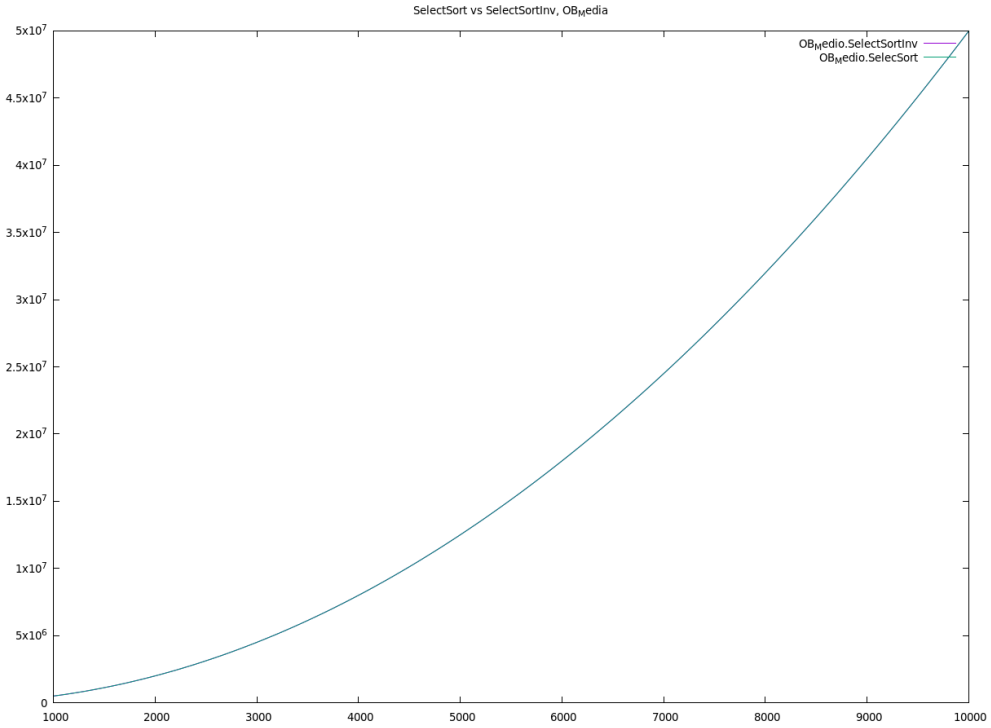


Gráfica con el tiempo medio de reloj para SelectSort, al ser el tiempo de ejecución de este algoritmo siempre N^2/2+N/2 podremos ver como la gráfica crece aplicando esta fórmula, y como explicamos en el apartado anterior esta representa el caso medio que es igual al mejor caso y al peor caso

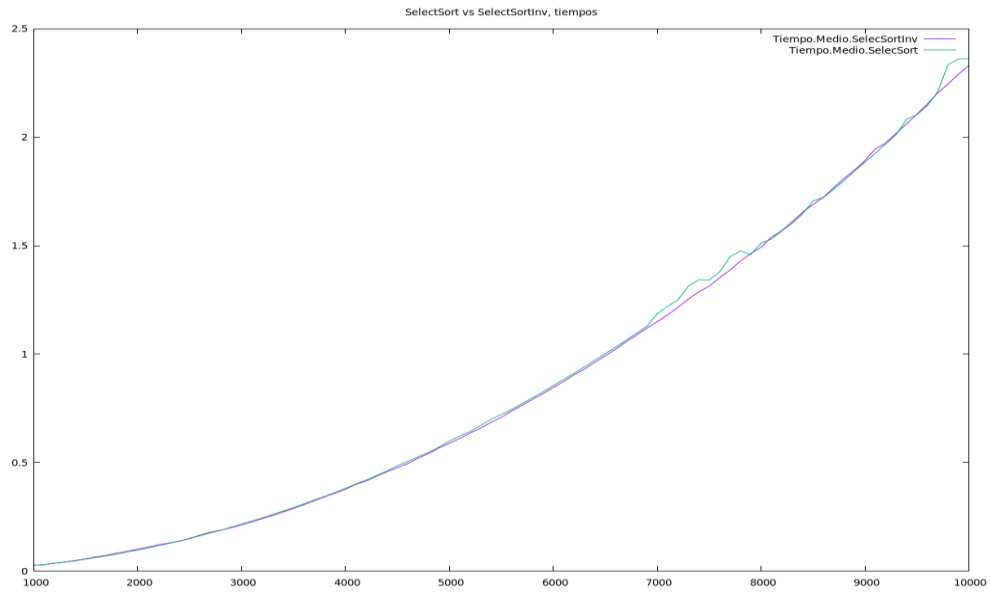


3.3 Apartado 3

Gráfica comparando el tiempo medio de OBs para SelectSort y SelectSortInv, Como se puede observar el tiempo es el mismo para ambas operaciones, esto es esperable pues funcionan de la misma manera y en ninguna de ellas es relevante el orden de los elementos para el tiempo de ordenación



Gráfica comparando el tiempo medio de reloj para SelectSort y SelectSortInv, se observan unos picos producidos por el procesador del ordenador en el que se hizo la simulación, aun así se sigue viendo en la comparación que el tiempo de ambos algoritmos es el mismo



4. Respuesta a las preguntas teóricas.

Aquí respondéis a las preguntas teóricas que se nos han planteado en la práctica.

4.1 Pregunta1: Hemos usado esta función de aleatoriedad porque cada caso es equiprobable, esto se debe a que utiliza la misma dinámica de una regla de tres, que dividiendo al número aleatorio dado por la función rand por el máximo(menos el mínimo) y multiplicando por el nuevo máximo(y finalmente sumándole el mínimo) conseguimos un valor equivalente en proporción al número aleatorio de rand, este método lo aprendimos en las clases del Dr. Iván Cantador en las clases de programación de primero, otra opción habría sido utilizar el módulo del valor dado por rand y el máximo(menos el mínimo)del valor buscado(para luego sumarle su mínimo), sin embargo este método solo será equiprobable en aquellos casos en que[RAND_MAX%(max-min) == 0] por lo que lo descartamos.

4.2 Pregunta 2: El algoritmo SelectSort ordena los elementos de un array de la siguiente manera: recorre el array desde la primera posición de la tabla hasta la última y si encuentra un elemento menor que el primero lo sustituye por este hasta terminar el recorrido, momento en el cual el elemento desde el que empezamos se considerará ordenado, por lo que volverá a recorrer el array pero empezando desde el siguiente, así seguirá ordenando la tabla elemento a elemento hasta que solo quede uno, el cual ya estará ordenado al no tener más posiciones con las que compararlo.

4.3 Pregunta 3: Como se explicó en la pregunta 2, SelectSort no actúa sobre el último elemento de la tabla porque al estar ordenada el resto de la tabla y ser el único elemento restante no le puede corresponder ninguna otra posición de la tabla,porque de ser así alguno de los otros elementos estaría mal colocado.

4.4 Pregunta 4: La operación básica de SelectSort es la comparación del valor de dos posiciones de la tablas (para ver si la posición b es menor que la a(lo que de ser cierto llevaría a que cambien sus posiciones)).

4.5 Pregunta 5: El tiempo de ejecución tanto en el mejor caso como en el peor caso es de: N^2/2+N/2 = N^2/2+O(N)

4.6 Pregunta 6: Los tiempos para ambos algoritmos son los mismos porque en ambos casos tardan el mismo tiempo para cualquier ordenación para una misma longitud de tabla y la diferencia entre ambos es que ordenan de menor a mayor que es un factor que no es afectado por la longitud de la tabla sino precisamente tiene un efecto equivalente a invertir la ordenación (que como hemos visto no afecta al tiempo de ejecución).