Practica Final FAA

Manuel Ángel Pazos Ruiz

Índice

1.Algoritmo del examen. Cálculo de los tiempos teóricos

1.1 . Pseudócódigo/s y análisis de coste

BusquedaKesimoMenorD

```
int función busquedaKesimoMenorD (A:vector; talla: int, k:int)
        OrdenaInserción(A, talla);
        return A[k];
fbusquedaKesimoMenorD
```

Inserción

Fila 1: 1 comparación y 1 op. aritmética

Fila 2: 1 asignación y 1 acceso vector

Fila 3: 1 asignación y 1 operación aritmética

Fila 4: Cond bucle: 2 comparaciones, 1 acceso vector, 1 lógica.

Fila 5: 1 asignación, 2 acceso vector y 1 op. aritmética

Fila 6: 1 asignación y 1 operación aritmética

$$\text{Caso peor}: C(n) = \sum_{i=2}^{n-1}\sum_{j=1}^{i-1} 1 = \sum_{i=2}^{n} i - 1 = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

$$\text{Caso mejor}: C(n) = \sum_{i=2}^{n} 1 = n - 1$$

$$\text{Caso medio}: C(n) = \frac{\frac{n(n-1)}{2} + n - 1}{2} = \frac{(n-1)(n+2)}{4}$$

$$\text{Caso peor, mejor, medio}: I(n) = \sum_{i=1}^{n-1} 1 = n - 1$$

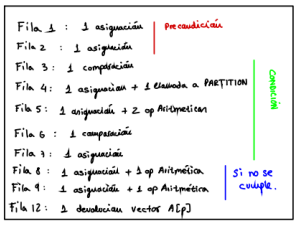
$$\text{Caso medio } T(n) = C(n) + 3 \cdot I(n) = (n-1)\frac{n+14}{4}$$

```
procedimiento inserción(T[1..n])
   para i← 2 hasta n hacer
      x ← T[i]
      j ← i-1
      mientras j>0 AND x < T[j] hacer
         T[j+1] ← T[j]
         j ← j-1
      fmientras
      T[j+1] ← x
   fpara
fprocedimiento
```

Como hace una llamada a el método de ordenación inserción hice el análisis de coste de inserción como ya hicimos en prácticas anteriores O(N^2)

BusquedakesimoMenorIt

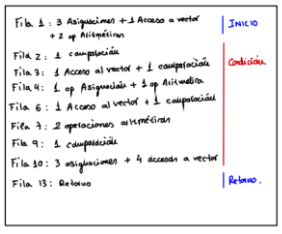
```
int función busquedaKesimoMenorIt (A:vector; talla: int, k:int)
       p = 1;
       r = talla;
       mientras (p<r) hacer
       q = Partition(A, p, r);
             i = q - p + 1;
             si (k <= i)
                   r = q;
             sino
                   p = q + 1;
                   k = k - i;
             fsi
       fmientras
       return A[p];
fbusquedaKesimoMenorIt
```

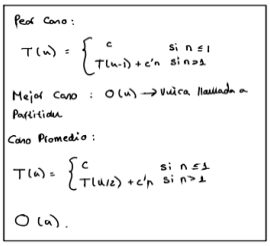


Ya que en este algoritmo se hace una llamada a partition y es de orden (N), este resultaria con este mismo orden.

Partition

```
int función Partition (A:vector;, p,r:int)
   piv= A[p]; i= p-1;j= r+1;
   mientras j ≥ i hacer
         mientras A[j] ≥ piv hacer
            j = j - 1;
         fmientras
         mientras A[i] ≤ piv hacer
            i = i + 1;
         fmientras
         si i < j entonces  /* A[i] ↔ A[j] */
               temp: int; temp= A[j]; A[j]=A[i]; A[i]=temp;
         fsi
   fmientras
   return j; /* retorna el índice para la división (partición) */
ffuncion Partition;
```

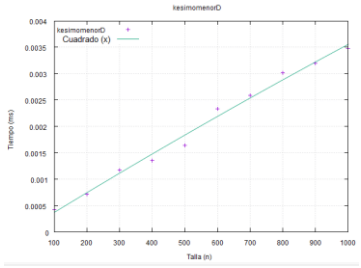




1.2 Tablas(ficheros) y Gráficas de coste

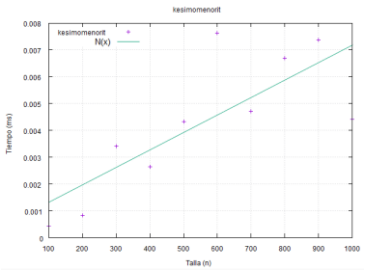
K-esimo menor directo

```
Busqueda kesimomenorD. Tiempos de ejecucion promedio

Talla           Tiempo (mseg)
100             0.000413
200             0.000729
300             0.001047
400             0.001519
500             0.001766
600             0.002251
700             0.002542
800             0.003069
900             0.002976
1000            0.003453
```



k-esimo menor iterativo

```
Busqueda kesimomenorit. Tiempos de ejecucion promedio

Talla           Tiempo (mseg)
100             0.000696
200             0.001197
300             0.001126
400             0.001728
500             0.002803
600             0.002457
700             0.003027
800             0.007573
900             0.008348
1000            0.005361
```



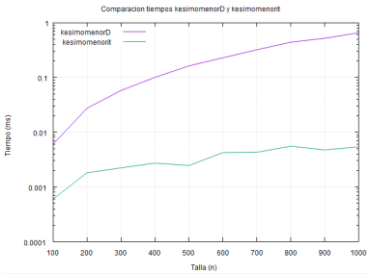
2-Conclusiones

Como vemos en los costes, vemos una clara diferencia entre los costes de un algoritmo y otro, k-esimomenorD es de O(N^2) mientras que k-esimomenorIT es de O(N), esto resulta en una clara diferencia entre ambos algoritmos a favor del algoritmo k-esimomenorIT.

3. Algoritmo/s del examen. Cálculo del tiempo experimental.

3.1. Tablas (ficheros) y Gráficas de coste

```
Busqueda kesimomenorD y kesimomenorit. Tiempos de ejecucion promedios

                kesimomenorD            kesimomenorit
      Talla     Tiempo (mseg)           Tiempo(mseg)
      100       0.005943                        0.000593
      200       0.027185                        0.001803
      300       0.057478                        0.002219
      400       0.099666                        0.002712
      500       0.161826                        0.002448
      600       0.227086                        0.004219
      700       0.319217                        0.004272
      800       0.438968                        0.005535
      900       0.517148                        0.004723
      1000      0.650247                        0.00538
```

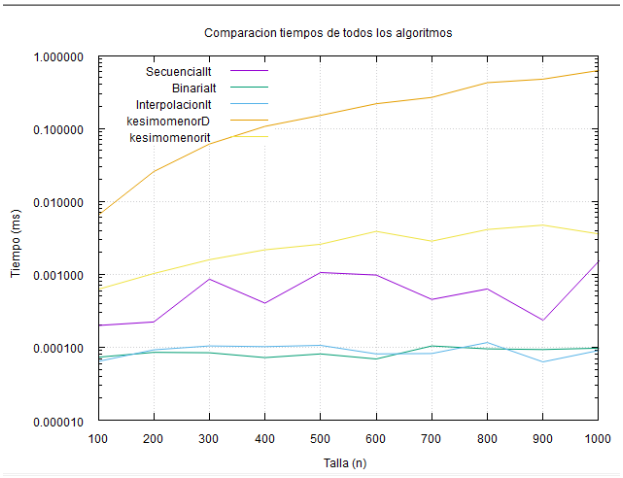


3.2. Conclusiones

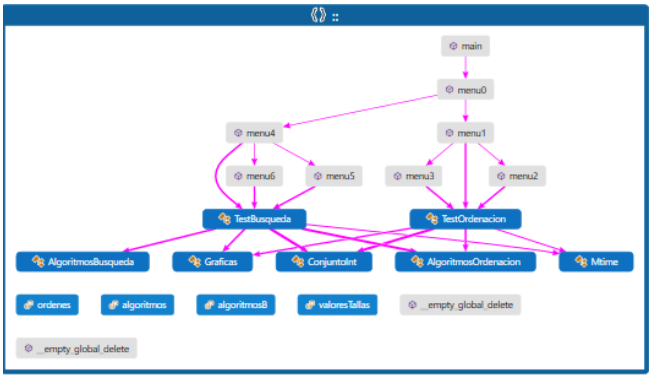
Vemos la comparación entre ambos algoritmos y notamos que hacen efectos sus ordenes N^2 y N respectivamente convirtiendo al algoritmo kesimomenorD

4. Comparación de los resultados teórico y experimental.

Cuando comparamos los resultados teóricos a los prácticos concuerdan y vemos como se adaptan los cálculos hechos en teoría con la implementación a continuación, vamos a ver la comparación de todos los algoritmos donde la veremos encabezada por el k-esimo menor directo y después el k-esimo menor iterativo.



5. Diseño de la aplicación.



AlgoritmosBusqueda->aquí podremos encontrar los distintos algoritmos de búsqueda implementados en la modificación además de los anteriores implementados en otras prácticas, además del Partition.

AlgortimosOrdenación-> Aquí encontraremos los algoritmos de ordenación implementados anteriormente en otras prácticas, este modulo no se utilizo en la modificación.

ConjuntoInt->Aquí encontraremos distintas herramientas para tratar vectores (escribir, vaciar, coger datos, generar una key).

Graficas->En graficas podemos encontrar los distintos archivos para que se ejecuten las graficas en el GNUPLOT, hay una para cada los casos medios, otra para comparar y otra para comparar todos.

Mtime-> aquí tenemos distintas funciones para cálculos de tiempo

Principal->En el menú principal, he dividido los menús en funciones void al ser tan numerosos me parecía bastante acertado.

TestBusqueda->Aquí hacemos distintas funciones para que el menú de búsqueda funcione correctamente (comprobar algoritmos, caso medio, comparar todos).

TestOrdenación-> Aquí hacemos distintas funciones para que el menú de ordenación funcione correctamente (comprobar algoritmos, caso medio, comparar todos).

6. Conclusiones y valoraciones personales de la Práctica Final.

Como conclusión diríamos que de los dos nuevos algoritmos implementados tenemos que el menos eficiente seria k-esimo menor directo que tiene orden de N^2, mientras que el k-esimo menor iterativo tiene orden N, por lo tanto es mas eficiente.

Como valoración personal de la practica me ha parecido una actividad enriquecedora ya que esta me ha dado la oportunidad de darme cuenta de cuales fueron mis errores después de haber hecho el examen y haber tenido la oportunidad de corregirlo y así aprender de mis errores, por lo tanto, me ha servido bastante.

7. Valoración personal de las prácticas de FAA.

A nivel global mirando el conjunto de practicas que hemos hecho durante el curso han servido un poco para poner en pie los conceptos adquiridos en teoría y verlos reflejados en las practicas, es bastante menos abstracto verlo en la práctica que verlo en la teoría, por eso me han resultado de mucha utilidad.