Febrero2019-Resuelto.pdf

alberto_fm_

Algorítmica y Modelos de Computación

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería
Universidad de Huelva

Febrero 2019

lunes, 3 de enero de 2022 18:24

Ejercicio_1. (2 puntos)

- Analizar el algoritmo de la **Búsqueda del *k*-ésimo menor elemento:** dado un vector de *n* elementos, el problema de la selección consiste en buscar el *k*-ésimo menor elemento.
- Supongamos que disponemos de la siguiente definición de tipo:
  CONST n = ...;
  TYPE vector = ARRAY [1..n] OF INTEGER;
  Y supongamos que `primero` y `último` indican los límites del array (inicialmente `primero=1` y `ultimo=n`)
- Para la solución del problema utilizamos la idea del algoritmo **Partition** (utilizado en Quicksort): El vector *A*[*p*..*r*] se particiona(reorganiza) en dos subvectores *A*[*p*..*q*] y *A*[*q*+1..*r*], de forma que los elementos de *A*[*p*..*q*] son menores o iguales que el pivote(por ej. primer elemento) y los de *A*[*q*+1..*r*] mayores o iguales.

```
int función Partition (A:vector;, primero,ultimo:int)
    piv= A[primero]; i= primero;j= ultimo;
    mientras j ≥ i hacer        Fallo en el examen, se dijo cambiar a j>i
          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;
```

- El algoritmo de la **Búsqueda del *k*-ésimo menor elemento** puede ser implementado:

1. Versión iterativa de la **Búsqueda del *k*-ésimo menor elemento**.

```
int función SelectIterativa (A:vector;, primero,ultimo, k:int)
    mientras primero < ultimo hacer
       q = Partition(A,primero,ultimo);
       si K ≤ q entonces
          ultimo = q          /*buscamos en A[primero..q]*/
       sino
          primero = q + 1     /*buscamos en A[q + 1.. ultimo]*/
       fsi
    fmientras
    return A[primero];
ffuncion SelectIterativa;
```

2. Versión recursiva de la **Búsqueda del *k*-ésimo menor elemento**.

```
int función SelectRecursiva (A:vector;, primero,ultimo, k:int)
    si (primero == ultimo)
          return A[primero];
    fsi
    q = Partition(A,primero,ultimo);
    i = q-primero+1;            /*i es el número de elementos en el primer subvector*/
    si (k ≤ i)
          return SelectRecursiva (A,primero,q,k);    /*buscamos en A[primero..q]*/
    sino
          return SelectRecursiva(A,q+1,ultimo,k-i); /*buscamos en A[q + 1.. ultimo]*/
    fsi
ffuncion SelectRecursiva
```

a. (1 puntos). Realizar una traza de `SelectRecursiva` y `SelectIterativa` con
   A = {31, 23, 90, 0, 77, 52, 49, 87, 60, 15} y *k*=7
b. (0,5 puntos).Calcular la complejidad del algoritmo iterativo propuesto mediante el conteo del número de operaciones elementales.
c. (0,5 puntos). Calcular la complejidad del algoritmo recursivo propuesto por el método de la ecuación característica.

a) SelectRecursiva(A, 1, 10, 7) $\quad \underline{K=7}$

$q = 4 \leftarrow$ Partition(A, 1, 10)

1) $A[j]=15$; $piv=31$; $A[i]=31$

{15, 23, 90, 0, 77, 52, 49, 87, 60, 31}
{15, 23, 31, 0, 77, 52, 49, 87, 60, 90}
{15, 23, 0, 31, 77, 52, 49, 87, 60, 90}

$i = 4$

$7 \leq 4 \rightarrow$ Falso

SelectRecursiva(A, 5, 10, 3)

$q=8 \leftarrow$ Partition(A, 5, 10)

{77, 52, 49, 87, 60, 90} $j=10$
{60, 52, 49, 87, 77, 90} $j=9$
{60, 52, 49, 77, 87, 90} $j=8$

$i=4$

$3 \leq 4 \rightarrow$ True

SelectRecursiva(A, 5, 8, 3)

$q = 7 \leftarrow$ Partition(A, 5, 8)

{60, 52, 49, 77} $j = \cancel{8}\ 8$

{49, 52, 60, 77} $j = \cancel{3}\ 7$

$i = 3$

$3 \leq 3 \rightarrow$ True.

SelectRecursiva(A, 5, 7, 3)

$q = 5. \leftarrow$ Partition(A, 5, 7)

{49, 52, 60} $j = 7$

{49, 52, 60} $j = 6$

{49, 52, 60} $j = 5$

$i = 1$

$3 \leq 1 \rightarrow$ False

SelectRecursiva(A, 6, 7, 2)

$q = 6 \leftarrow$ Partition(A, 6, 7)

{52, 60} $j = \cancel{8}\ 7$

$j = \cancel{7}\ 6$

$i = 1$

$2 \leq 1 \rightarrow$ Falso

SelectRec(A, 7, 7, 1)

$7 == 7 \rightarrow$ True

Return $\boxed{A[7] = 60}$

SelectIterativa(A, 1, 10, 7)

$q = 4 \leftarrow$ Partition(A, 1, 10)

$7 \leq 4 \rightarrow$ Falso

primero = 5

$q = 8 \leftarrow$ Partition(A, 5, 10)

$7 \leq 8 \rightarrow$ Verdadero

ultimo = 8

$q = 7 \leftarrow$ Partition(A, 5, 8)

$7 \leq 7 \rightarrow$ Verdadero

ultimo = 7

$q = 5 \leftarrow$ Partition(A, 5, 7)

$7 \leq 5 \rightarrow$ Falso

primero = 6

$q = 6 \leftarrow$ Partition(A, 6, 7)

$q = 6 \leftarrow$ Partition (A, 6, 7)

$7 \leq 6 \rightarrow$ Falso

primero = 7

primero < ultimo → Falso

return $A[7] = \underline{60}$

---

b)

```
int función SelectIterativa (A:vector;, primero,ultimo, k:int)
    mientras primero < ultimo hacer
        q = Partition(A,primero,ultimo);
        si K ≤ q entonces
            ultimo = q
        sino
            primero = q + 1
        fsi
    fmientras
    return A[primero];
ffuncion SelectIterativa;
```

```
int función Partition (A:vector;, primero,ultimo:int)
    piv= A[primero]; i= primero;j= ultimo;
    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;
```

Fallo en el examen, se dijo cambiar a j>i

$$T(n) = 1 + \sum^{?}_{?} (1 + 1 + T_{partition} + 1 + 2 + 1) + 2$$

$$T(n) = 3 + \sum^{?} (2 + T_{partition} + 4) =$$

$$= 3 + \sum (6 + T_{partition})$$

<u>Caso PEOR</u> → Partition divide el vector en 2 subvectores de tamaño 1 y (n-1).

$$T_{partition}(n) = 5 + \underset{\text{Comp. / Find}}{1} + 1 + \sum_{i=1}^{n} (2 + 2 + 1 + 1 + 8 + 1) + 2 + 1)$$

Salto Final

$$= 7 + \sum_{i=1}^{1} (18) = 18n + 7 \;.$$

$$T(n) = 1 + \sum_{i=1}^{n} (6 + 18i + 7) = 1 + 13n + \sum_{i=1}^{n} 18i = \qquad n^2)$$

$$= 13n + 1 + 18 \sum_{i=1}^{n} i = 13n + 1 + 18 \cdot \left(\frac{n(n+1)}{2} \right)$$

$$= 13n + 1 + 9n(n+1) = 13n + 1 + 9n^2 + 9n =$$

$$= 9n^2 + 22n + 1 \rightarrow T(n) \in O(n^2)$$

<u>Caso MEJOR</u> → El elemento se encuentra en el subvector de tamaño 1
El if no se cumple nunca.

$$T_{partition}(n) = 7 + \sum_{i=1}^{1} \left(1 + 2 + 2 + 1 + 1 + \sum_{j=1}^{n} (2 + 2 + 1)\right)$$

$$= 7 + 7 + 5n = 14 + 5n \in O(n)$$

$$T(n) = 1 + 1 + \sum_{i=1}^{1} 1 + T_{partition}(i) + 1 + 1 + 1 + 1) =$$

$$= 2 + 1 + 14 + 5n + 4 = \boxed{5n + 21 \in O(n)}$$

<u>Caso Medio</u>

$$\underline{T_{partition}(n)} = 7 + \sum_{i=1}^{n} \left(\sum_{j=1}^{p/n} (2 + 5) + \sum_{k=1}^{\frac{n-p}{n}} (2 + 5) + \frac{1}{2} \cdot 8 + 1 \rightarrow \right.$$

$$= 7 + n \left(\frac{p}{n} (7) + \frac{n-p}{n} (7) + 4 + 1 \rightarrow \right.$$

$$= 7 + 7p + 7(n-p) + 5n \rightarrow$$

$$= 7(1 + \cancel{p} + n - \cancel{p}) + 5n \rightarrow$$

$$= 7 + 7n + 5n = 7 + 12n \rightarrow T(n) \in O(n)$$

→ la media

$$T(n) = 2 + \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{i}\left(5 + T_{partition}(j)\right)\right) + 2 \rightarrow$$

$$= 2 + 1/n \sum_{i=1}^{n}\left(\sum_{j=1}^{i}(5 + 12j + 7)\right) + 2 \rightarrow$$

$$= 2 + \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{i}(12 + 12j)) + 2 = 4 + \frac{1}{n}\left(\sum_{i=1}^{n} 12i + 12\sum_{j=1}^{i} j\right) \rightarrow$$

$$= 4 + \frac{1}{n}\left(\sum_{i=1}^{n}\left(12i + 12\left(\frac{i(i+1)}{2}\right)\right)\right) = 4 + 1/n\left(12\left(\frac{n(n+1)}{2}\right) + 12\sum_{i=1}^{n}\left(\frac{i(i+1)}{2}\right)\right) \rightarrow$$

$$= 4 + \frac{1}{n}\left(6n(n+1) + \frac{12}{2}\sum_{i=1}^{n} i^2 + i\right) = 4 + \frac{1}{n}\left(6n(n+1) + 6\left(\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right)\right)$$

$$= 4 + \frac{1}{\cancel{n}}\left(6\cancel{n}(n+1) + \cancel{n}(n+1)(2n+1) + 3\cancel{n}(n+1)\right) \rightarrow$$

$$= 4 + 6n + 6 + 2n^2 + n + 2n + 1 + 3n + 3 \rightarrow$$

$$= 14 + 12n + 2n^2 = \boxed{2n^2 + 12n + 14 \in O(n^2)}$$

c)

```
int función SelectRecursiva (A:vector;, primero,ultimo, k:int)
   si (primero == ultimo)
         return A[primero];        n=1
   fsi
   q = Partition(A,primero,ultimo);
   i = q-primero+1;              /*i es el número de element
   si (k ≤ i)
         return SelectRecursiva (A,primero,q,k);    /*busc
   sino
         return SelectRecursiva(A,q+1,ultimo,k-i); /*busc
   fsi
ffuncion SelectRecursiva
```

```
int función Partition (A:vector;, primero,ultimo:int)
   piv= A[primero]; i= primero;j= ultimo;
   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;
```

Fallo en el examen, se dijo cambiar a j>i

$$T(n) = \begin{cases} 3 & n = 1 \\ T(n-1) + 18n + 7 + 11 & n > 1 \end{cases}$$

$$T(n) = T(n-1) + 18n + 18 \rightarrow T(n) - T(n-1) = 18n + 18 \xrightarrow{T(n) = x^k}$$

$$(x-1)(x-1)^2 = 0 \rightarrow \boxed{x = 1 \text{ (triple)}}$$

$$T(n) = c_0 \cdot 1^n \cdot n^0 + c_1 \cdot 1^n \cdot n^1 + c_2 \cdot 1^n \cdot n^2$$

$$T(n) = c_0 + c_1 n + c_2 n^2$$

$T(1) = \underline{3}$

$T(2) = T(1) + 36 + 19 = 3 + 55 = \underline{58}$

$T(3) = 58 + 54 + 19 = \underline{131}$

$T(4) = 131 + 72 + 19 = \underline{222}$

1) $C_0 + 2C_1 + 4C_2 = 58$

2) $C_0 + 3C_1 + 9C_2 = 131$

3) $C_0 + 4C_1 + 16C_2 = 222$

$$A = \begin{vmatrix} 1 & 2 & 4 & 58 \\ 1 & 3 & 9 & 131 \\ 1 & 4 & 16 & 222 \end{vmatrix} = F_2 - F_1 \to F_2$$

$$= \begin{vmatrix} 1 & 2 & 4 & 58 \\ 0 & 1 & 5 & 73 \\ 1 & 4 & 16 & 222 \end{vmatrix} = F_3 - F_1 \to F_3$$

$$= \begin{vmatrix} 1 & 2 & 4 & 58 \\ 0 & 1 & 5 & 73 \\ 0 & 2 & 12 & 164 \end{vmatrix} = F_3 - 2F_2 \to F_3$$

$$= \begin{vmatrix} 1 & 2 & 4 & 58 \\ 0 & 1 & 5 & 73 \\ 0 & 0 & 2 & 18 \end{vmatrix} = \frac{1}{2}F_3 \to F_3$$

$$= \begin{vmatrix} 1 & 2 & 4 & 58 \\ 0 & 1 & 5 & 73 \\ 0 & 0 & 1 & 9 \end{vmatrix} = F_2 - 5F_3 \to F_2$$

$$= \begin{vmatrix} 1 & 2 & 4 & 58 \\ 0 & 1 & 0 & 28 \\ 0 & 0 & 1 & 9 \end{vmatrix} = F_1 - 2F_2 - 4F_3 \to F_1$$

$$= \begin{vmatrix} 1 & 0 & 0 & -34 \\ 0 & 1 & 0 & 28 \\ 0 & 0 & 1 & 9 \end{vmatrix} \quad \begin{aligned} c_0 &= -34 \\ c_1 &= 28 \\ c_2 &= 9 \end{aligned}$$

$$T(n) = \underline{-34 + 28n + 9n^2}$$

CASO MEJOR

$$T(n) \begin{cases} 3 & n = 1 \\ T(1) + 5n + 14 + 9 & n > 1 \end{cases}$$

$T(n) = 3 + 5n + 14 + 9 = 5n + 26 \in O(n)$

CASO MEDIO

$$T(n) \begin{cases} 3 & n = 1 \\ \frac{1}{n-1}\sum_{i=1}^{n-1}(T(i)) + 12n + 7 + 11 \end{cases}$$

$T(n) = \frac{1}{n-1}\sum_{i=2}^{n-1}(T(i)) + \frac{T(1)}{n-1} + 12n + 18 \Rightarrow$

$T(2) = \frac{1}{2}\left(\sum_{2}^{1} T(i) + \frac{3}{\cancel{2}1}\right) 24 + 18 = 3 + 24 + 18 = \underline{\underline{45}}$

$K = 45 \to n \geq 2 \quad K n \geq T(n)$

$n = 2 \quad 45 \cdot 2 \geq 45$

$90 \geq 45$

$T(3) = \frac{1}{2}\sum_{2}^{3-1} T(i) + \frac{3}{2} + 36 + 18$

$= \frac{1}{2}(45) + \frac{3}{2} + 54 = 22.5 + 1.5 + 54 = \underline{\underline{78}}$

$\underline{45 \cdot 3 \geq 78}$

$\boxed{T(n) = 45n \in O(n)}$

Ejercicio_2. (1,5 puntos)

- La sucesión de Fibonacci se define como
  fib(0) = fib (1) = 1;
  fib(n) =fib (n – 1) + fib (n – 2) si n ≥ 2.
- **Se pide**:

a. (0,75 puntos) Escribir tres posibles implementaciones, **simples y cortas**, para el cálculo del n-ésimo número de fib con las siguientes estrategias:
   1. procedimiento directo
   2. divide y vencerás
   3. programación dinámica

b. (0,75 puntos). Realizar una estimación del orden de complejidad de los tres algoritmos del apartado anterior. Comparar los órdenes de complejidad obtenidos, estableciendo una relación de orden entre los mismos.

a)

1)
```
fibonacciDirecto(n)
    a = 0
    b = 1
      para i hasta n:
           a = b
           b = a+b
      fpara
    return a;
ffibonacciDirecto.
```

2)
```
fibonacciRecursivo( n)
    si (n==0  OR  n==1)
         return 1;
    sino
         return fibonacciRecursivo(n-1) + fibonacciRec (n-2)
    fsi
```

3)
```
fibonacciPD(n)
    F array of integer [0.. N]
    F(0) = 1
    F(1) = 1
    i=2
     mientras i≤n
          F[i] = F[i-1] + F[i-2]
          i++;
    fmientras
          return F[n]
```

b) Directo:

1) $T(n) = \sum_{i=0}^{n}(6) + 3 = 6n+3$

$\boxed{T(n) \in O(n)}$

2) DyV:

$$T(n)\begin{cases} 4 & n=0,\ n=1 \\ 9+T(n-1)+T(n-2) & n\geq 2 \end{cases}$$

$$T(n) = 9+T(n-1)+T(n-2) \rightarrow T(n)-T(n-1)-T(n-2) = 9$$

$$\rightarrow (x^2-(x-1)-2)(x-1) = 0 \rightarrow (x^2-x-1)(x-1) = 0$$

$$\begin{cases} r_1 = 1 \\ r_2 = \frac{1\pm\sqrt{1+4}}{2} = \frac{1+\sqrt{5}}{2} \\ r_3 = \frac{1-\sqrt{5}}{2} \end{cases}$$

$$T(n) = c_0 \cdot 1^n \cdot n^0 + c_1 \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n \cdot n^0 + c_2 \cdot \left(\frac{1-\sqrt{5}}{2}\right)^n \cdot n^0 \rightarrow$$

$$T(n) = c_0 + \left(\frac{1+\sqrt{5}}{2}\right)^n c_1 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^n$$

$$\hookrightarrow T(n) \in O\left(\frac{1+\sqrt{5}}{2}\right)^n$$

PD:

$$T(n) = 3+\sum^{n} 11 + 2 = 5 + 11\cdot(n-2+1) = 5+11n-11 = 11n-6 \in O(n)$$

PD:

$$T(n) = 3 + \sum_{i=2}^{n} 11 + 2 = 5 + 11 \cdot (n-2+1) = 5 + 11n - 11 = 11n - 6 \in O(n)$$

(Directo < PD < DyV)

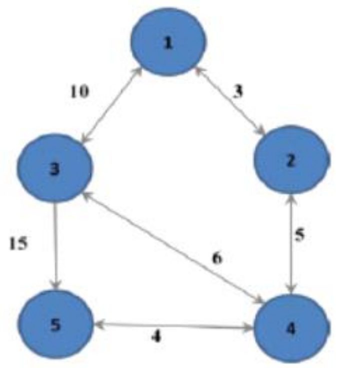
Ejercicio_3. (3 puntos)

El algoritmo de Floyd determina la ruta más corta entre dos nodos cualesquiera de la red. Una posible implementación consiste en:

- Representar la red de *n* nodos como una matriz cuadrada de orden *n*, la llamaremos **matriz** *C*. De esta forma, **el valor C[i , j] representa el coste de ir desde el nodo *i* al nodo *j***, inicialmente en caso de no existir un arco entre ambos, el valor **C[i , j]** será infinito.
- Definir otra **matriz** *D*, también cuadrada de orden *n*, cuyos elementos van a ser los **nodos predecesores en el camino hacia el nodo origen**, es decir, el valor **D[i , j] representará el nodo predecesor a *j* en el camino mínimo desde *i* hasta *j***. Inicialmente son caminos de **longitud** *1*, por lo que **D[i , j]= i**
- Los pasos a dar en la aplicación del algoritmo de Floyd son los siguientes:
  1. Formar las matrices iniciales *C* y *D*.
  2. Se toma *k=1*.
  3. Se selecciona la fila y la columna *k* de la matriz *C* y entonces, para *i* y *j*, con *i≠k*, *j≠k* e *i≠j*, hacemos:
     - **Si (C[i , k] + C[k , j]) < C[i , j] ⇒ D[i , j] = D[k, j] y C[i , j] = C[i , k] + C[k , j]**
     - En caso contrario, dejamos las matrices como están.
  4. Si *k ≤ n*, aumentamos *k* en una unidad y repetimos el paso anterior(3.), en caso contrario paramos las iteraciones.

➢ Se pide:

a. (1.5 puntos). Aplicar el algoritmo de Floyd sobre el siguiente grafo para obtener las rutas más cortas entre cada dos nodos.



b. (1.5 puntos).Escribir posibles implementaciones de algoritmos para calcular:

1. **Distancia más corta**, Aplicar por ej, a la distancia más corta del nodo 1 al nodo 5.
2. **La ruta asociada del camino mínimo**. Aplicar por ej, entre el nodo 1 y el nodo 5

C	1	2	3	4	5
1	∞	3	10	~~∞~~ 8	~~∞~~ ~~25~~ 12
2	3	∞	~~∞~~ 13	5	~~∞~~ 28
3	10	~~∞~~ 13	∞	6	15
4	~~∞~~ 8	5	6	∞	4
5	~~∞~~ 25	∞	15	4	∞

D	1	2	3	4	5
1	–	1	1	2	3
2	2	–	1	2	3
3	3	1	–	3	3
4	2	4	4	–	4
5	3	3	5	5	–

K=1

K=1 , i=2 , j=3

$(3+10) < \infty \rightarrow C[i,j]=13 \; ; \; D[i,j] = D[K,j]$

$K=1$, $i=2$, $j=4$

$(3+\infty) < 5 \to$ NO

$K=1$, $i=2$ $j=5$

$(3+\infty) < \infty \to$ NO

$K=1$, $i=3$, $j=2$

$(10+3) < 13 \to$ NO

$K=1$ $i=3$, $j=4$

$(10+\infty) < 6 \to$ NO

$K=1$, $i=3$, $j=5$

$10+\infty < 15 \to$ NO

$K=1$ $i=4$ $j=2$

$\infty < C[4,2] \to$ NO

$C[1,4]=\infty \to$ Salta a la siguiente i

$K=1$ $i=5$ $j=2$

$C[1,5]=\infty \to$ Salta a la siguiente K

$K=2$, $i=1$, $j=3$

$(3+13) < 10 \to$ NO

$K=2$, $i=1$ $j=4$

$(3+5) < \infty \to$ Sí $\to$ $C[1,4]=8$; $D[1,4]=2$

$K=2$ $i=1$ $j=5$

$(3+\infty) < ?$ NO

$K=2$ $i=3$ $j=1$

$(13+3) < 10 \to$ NO

$K=2$ $i=3$ $j=4$

$(13+5) < 6 \to$ NO

$K=2$ $i=3$ $j=5$

$13+\infty \to$ NO

$K=2$, $i=4$, $j=1$

$(5+3) < 8 \to$ NO

$K=2$ $i=4$ $j=3$

$(5+13) < 6 \to$ NO

$K=2$ $i=4$ $j=5$

$5+\infty \to$ NO

$K=2$ $i=5$ $j=1$

$C[2,5]=\infty \to K{+}{+}$

$K=3$ $i=1$ $j=2$

$10+13 < 3 \to$ NO

$K=3$ $i=1$ $j=4$

$10+6 < 8 \to$ NO

$K=3$ $i=1$ $j=5$

$10+15 < \infty \to$ Sí

$D[1,5]=3$

$C[1,5]=25$

$K=3$ $i=2$ $j=1$

$13+10 < 3 \to$ NO

$K=3$, $i=2$, $j=4$

$13+6 < 5 \to$ NO

$K=3$, $i=2$, $j=5$

$13+15 < \infty \to$ Sí

$K=3$, $i=4$, $j=1$,

$6+10 < 8 \to$ NO

$K=3$, $i=4$, $j=2$

$6+13 < 5 \to$ NO

$K=3$, $i=4$, $j=5$

$6+15 < 4 \to$ NO

... NO hay ninguna $K=3$ que satisfaga la condición ...

$K=4$, $i=1$, $j=2 \to$ NO

$K=4$, $i=1$, $j=3 \to$ NO

$K=4$, $i=1$, $j=5 \to$ Sí $\to$ $D[1,5]=4$ $C[1,5]=12$

...

C	1	2	3	4	5	D	1	2	3	4	5
1	-	3	10	8	12	1	-	1	1	2	4
2	3	-	11	5	9	2	2	-	4	2	4
3	10	11	-	6	10	3	3	4	-	3	4
4	8	5	6	-	4	4	2	4	4	-	4
5	12	9	10	4	-	5	4	4	4	5	-

b) Para calcular la distancia mínima solo habría que acceder directamente a la tabla C

C[1][5] = 12

c) Para recomponer la ruta asociada habría que realizar el siguiente algoritmo

```
algoritmo RecomponerRuta (origen, destino: Integer; D: array)
    si (origen == destino || destino == D[origen][destino])
        devolver 0;
    sino
        Escribir (destino);
        RecomponerRuta (origen, D[origen][destino], D);
    fsi
falgoritmo
```

Ejercicio_4. (1,5 puntos)

Consideremos el problema de la mochila modificado en el que tenemos:

- n objetos, cada uno con un peso (p_i) y un valor o beneficio (b_i)
- Una mochila en la que podemos meter objetos, con una capacidad de peso máximo **M**.
- Cada objeto puede **meterse** dentro de la mochila, **no meterse**, o meterse la **mitad** del objeto obteniendo la mitad del beneficio.
- **Objetivo:** llenar la mochila con esos objetos, maximizando la suma de los beneficios (valores) transportados, y respetando la limitación de capacidad máxima **M**.
- Se supondrá que los objetos se **pueden partir en la mitad** ($x_i = 0$, $x_i = \frac{1}{2}$, $x_i = 1$).

➢ **Se pide:**

a) (1 punto). Diseñar un algoritmo voraz para resolver el problema aunque no se garantice la solución óptima. Es necesario marcar en el código propuesto a que corresponde cada parte en el esquema general de un algoritmo voraz (criterio, candidatos, función……). Si hay más de un criterio posible elegir uno razonadamente y discutir los otros. Comprobar si el algoritmo garantiza la solución óptima en este caso (la demostración se puede hacer con un contraejemplo). Calcular su tiempo de ejecución.

b) (0,5 puntos). Aplicar el algoritmo para n = 2; M = 5; p = (8, 5); b = (10, 6)

```
funcion MochilaModificadaVoraz ( M: Integer, p,b: array [1..N] of Integer)
    X: array [1..N] of Float;
    peso := 0;
    ordenarPorCriterio();
    i ← 0
        mientras ( i < n ) hacer
            si ( peso + p[i] ≤ M) hacer
                X[i] = 1;
                peso ← peso + p[i];
            sino
                si (peso + p[i]/2 ≤ M) hacer
```

sino
si (peso + p[i]/2 ≤ M) hacer
X[i] = 1/2;
peso ← peso + p[i]/2;
fsi
fsi
fmientras
ffuncion

b) Aplicamos el algoritmo para:

$n=2$, $M=5$, $p=(8,5)$, $b=(10,6)$

ordenarPorCriterio() → De menor a mayor cociente beneficio/peso

$b = (6, 10)$
$p = (5, 8)$
$b/p = (1.2, 1{,}25)$

1) peso = 5
$X = \{1, 0\}$

2) peso = 5
$x = \{1, 0\}$;

Ejercicio_5. (2 puntos)

- Dado el AFND = ({a,b} , {p,q,r,s}, f, p, {s}) donde *f* viene dada por la siguiente tabla de transiciones:

	a	b	λ
→ p	{q,s}	{p}	{q,r}
q		{q,r}	{r}
r		{p,s}	{q}
* s	{s}	{q,r,s}	

➢ **Se pide:**
a. (0,5 puntos). El AFD equivalente
b. (0,5 puntos). El AFD mínimo
c. (0,5 puntos).La gramática regular equivalente al AFD obtenido en el apartado b
d. (0,5 puntos). Obtener una expresión regular equivalente al AFD obtenido en el apartado b

a)

	a	b
→ Q_0	Q_1	Q_2
* Q_1	Q_3	Q_1
* Q_2	Q_1	Q_2
* Q_3	Q_3	Q_1
—	—	—

$Q_0 = CL(p) = \{p, q, r\}$
$f'(Q_0, a) = \{q, s, r\} = Q_1$
$f'(Q_0, b) = \{p, q, r, s\} = Q_2$
$f'(Q_1, a) = \{s\} = Q_3$
$f'(Q_1, b) = \{q, r, p, s\} = Q_2$
$f'(Q_2, a) = \{q, s, r\} = Q_1$
$f'(Q_2, b) = \{p, q, r, s\} = Q_2$
$f'(Q_3, a) = \{s\} = Q_3$
$f'(Q_3, b) = \{q, r, s\} = Q_1$

b) Para calcular el AFD-mínimo, usaremos el algoritmo de conjunto-cociente:

$E/Q_0 = (c_0 = \{Q_0\}, c_1 = \{Q_1, Q_2, Q_3\})$

$f'(Q_1, a) = c_1$; $f'(Q_1, b) = c_1$
$f'(Q_2, a) = c_1$; $f'(Q_2, b) = c_1$
$f'(Q_3, a) = c_1$; $f'(Q_3, b) = c_1$

Como son indistinguibles, no creamos un nuevo conjunto.

Por tanto, el AFD mínimo será:

	a	b
$\rightarrow C_0$	C_1	C_1
$*C_1$	C_1	C_1

c) La gramática regular equivalente viene dada por la cuádrupla:

$G = \{Q, \Sigma, P, F\}$ donde:

$Q = \{c_0, c_1\}$

$\Sigma = \{a, b\}$

$F = \{c_1\}$

P viene definido por las producciones:

$$P = \begin{cases} C_0 ::= aC_1 \mid bC_1 \mid a \mid b \\ C_1 ::= aC_1 \mid bC_1 \mid a \mid b \end{cases}$$

d) La expresión regular equivalente para el AFD anterior es:

$$\begin{cases} X_0 = aX_1 + bX_1 + a + b \\ X_1 = aX_1 + bX_1 + a + b \end{cases} \rightarrow \boxed{X = aX + \beta \leftrightarrow X = a^*\beta}$$

$X_1 = (a+b)X_1 + a + b \rightarrow X_1 = (a+b)^*a + (a+b)^*b$

$X_0 = a((a+b)^*a + (a+b)^*b) + b((a+b)^*a + (a+b)^*b) + a + b$

$X_0 = a(a+b)^*a + a(a+b)^*b + b(a+b)^*a + b(a+b)^*b + a + b$