

Práctico 5: Diseño de Algoritmos:

Año 2026

Ejercicio 1:

Suponga que tiene una cantidad ilimitada de monedas con valores de \$1, \$4 y \$6 y se debe dar cambio de \$8 con esas monedas. Los algoritmos en **a)** y **b)** dan solución al problema del cambio, donde el parámetro d tiene las denominaciones de las monedas y m o k es el monto sobre el que se pide cambio.

- a)** **Pre-condiciones:** $d[1] > d[2] > \dots > d[n]$
function cambio1(d :array[1.. n] de **entero**, m :**entero**) **ret** S :array[1.. n] de **entero**
 for i := 1 **to** n **do**
 $S[i]$:= $m \text{ div } d[i]$
 m := $m \text{ mod } d[i]$
 od
end function
- b)** **Pre-condiciones:** $d[1] < d[2] < \dots < d[n]$
function cambio2(d :array[1.. n] de **entero**, k :**entero**) **ret** nr :array[0.. n] de **entero**
 var m :array[0.. n ,0.. k] de **entero**
 r, s :**entero**
 for i := 0 **to** n **do** $m[i,0]$:= 0 **od**
 for i := 1 **to** k **do** $m[0,i]$:= ∞ **od**
 for i := 1 **to** n **do**
 for j := 1 **to** k **do**
 if $d[i] > j$ **then** $m[i,j]$:= $m[i-1,j]$
 else $m[i,j]$:= $\min(m[i-1,j], 1+m[i,j-d[i]])$
 fi
 od
 od
 for i := 0 **to** n **do** $nr[i]$:= 0 **od**
 $nr[n]$:= $m[n,k]$
 if $m[n,k] < \infty$ **then**
 r := n
 s := k
 while $m[r,s] > 0$ **do**
 if $m[r,s] == m[r-1,s]$ **then** r := $r-1$
 else
 $nr[r]$:= $nr[r]+1$
 s := $s-d[r]$
 fi
 do
 fi
 end function

- 1) ¿A qué técnica de diseño, de las conocidas por usted, pertenece cada algoritmo? Justifique su respuesta.
- 2) ¿Cuál obtiene una solución óptima para el problema planteado, considerando dar el cambio con la menor cantidad de monedas posibles?

Ejercicio 2:

Dadas las siguientes frecuencias para los caracteres contenidos en un archivo, obtener los códigos para cada uno de ellos al aplicar el algoritmo de Huffman:

a:1, b:1, c:2, d:3, e:5, f:8, g:13, h:21, i:34, j:55

Ejercicio 3:

Dados los siguientes algoritmos que resuelven el mismo problema:

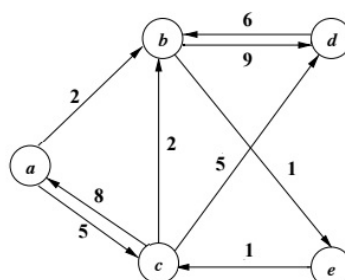
a) **function** num1(T:array [1..n] de entero, n:entero) **ret** entero
 var i:entero
 if n<=2 **then**
 return 1
 else
 T[1]:=1
 T[2]:=1
 for i:=3 **to** n **do**
 T[i]:=T[i-1]+T[i-2]
 od
 return T[n]
 fi
end function

b) **function** num2(n: entero) **ret** entero
 var i,suma,x,y:entero
 if n<=2 **then**
 return 1
 else
 x:=1
 y:=1
 for i:=3 **to** n **do**
 suma:=x+y
 y:=x
 x:=suma
 od
 return suma
 fi
end function

- 1) ¿A qué técnica de diseño, de las conocidas por usted, pertenece cada algoritmo? Justifique su respuesta.
- 2) ¿Qué problema resuelven?
- 3) Considerando tiempo y espacio ¿Cuál obtiene una solución óptima para el problema planteado? Exprese en notación O .

Ejercicio 4:

Dado el siguiente 1-dígrafo que representa los recorridos de una empresa de transporte, cada nodo simboliza una ciudad, cada arco el camino que cubre el transporte y el número sobre el arco es el costo de realizar ese viaje.



Se pide:

- Ejecutar el algoritmo que permita obtener el viaje más barato desde la ciudad a a cada una de las demás ciudades de la región. Muestre paso a paso la ejecución.
- Explicar cómo modificaría el algoritmo anterior si lo único que quiere obtener es el viaje de menor costo entre un determinado par de ciudades ($\text{viaje-barato}(x, y)$).
- Analizar el esfuerzo máximo, medido en tiempo, expresándolo en notación O , considerando que tiene n vértices y m arcos.
- ¿Cómo modificaría el algoritmo ejecutado en $a)$ para que permita recuperar cuál es el recorrido del viaje más barato? Realice una rutina en pseudo-código que le permita mostrar el recorrido de costo mínimo entre el vértice origen y un vértice dado.
- ¿A qué técnica de diseño, de las conocidas por usted, pertenece el algoritmo que ejecutó en $a)$? Explicar cómo llega a esa conclusión.

Ejercicio 5:

En la siguiente matriz de incidencia de un grafo se muestran las distancias entre los aeropuertos de las ciudades de Londres, México City, Nueva York, París, Pekín y Tokio:

	L	MC	NY	Pa	Pe	To
L	-	56	35	2	51	60
MC	56	-	21	57	78	70
NY	35	21	-	36	68	68
Pa	2	57	36	-	51	61
Pe	51	78	68	51	-	13
To	60	70	68	61	13	-

- Graficar el grafo que se obtiene a partir de dicha matriz.
- Dar el nombre de un algoritmo que permita obtener un grafo parcial (que incluya a todas las ciudades) conexo, de manera tal que la suma de distancias entre cada ciudad sea la mínima. Ejecutar el mismo sobre el grafo obtenido en el punto $a)$, agregando las hipótesis que considere necesarias.
- Suponiendo que las filas de la matriz indican el origen del arco y las columnas su destino, ejecutar el algoritmo que permita obtener los caminos más cortos desde Pekín a las demás ciudades.

Ejercicio 6:

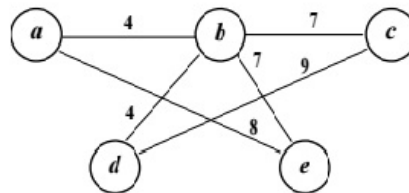
Considerar el dígrafo del Ejercicio 4 como un grafo que representa un grupo de ciudades de una región, y los valores que aparecen sobre las aristas indican la distancia de una ciudad a la otra. Se pide:

- Obtener el recorrido más corto que pasa por todas las ciudades, para ello utilizar el algoritmo de *Prim* partiendo desde a . Mostrar cómo quedan las estructuras después de cada paso de la ejecución. Desde el punto de vista de la teoría de grafos ¿Qué le permite obtener este algoritmo?
- Analizar el esfuerzo máximo, medido en tiempo, expresándolo en notación O , considerando que tiene n vértices y m arcos.
- Si ejecuta el algoritmo de *Prim* varias veces con el mismo grafo de entrada G ¿Se podrán obtener diferentes resultados cada vez?

- d) ¿A qué técnica de diseño, de las conocidas por usted, pertenece el algoritmo de *Prim*? Justifique su respuesta.

Ejercicio 7:

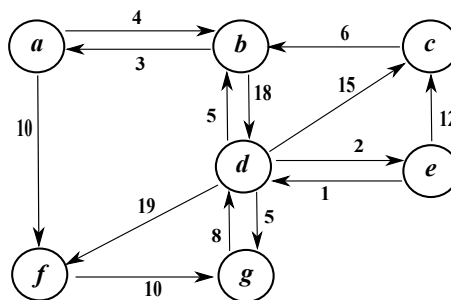
Dado el grafo:



- a) Ejecutar paso a paso el algoritmo de *Kruskal*. Muestre cómo quedan las estructuras después de cada paso de la ejecución. ¿Qué le permite obtener este algoritmo?
- b) Analizar el esfuerzo máximo, medido en tiempo, expresándolo en notación O , considerando que tiene n vértices y m arcos.
- c) ¿El algoritmo de *Kruskal* puede obtener diferentes resultados si se lo ejecuta varias veces sobre el mismo grafo G de entrada?
- d) ¿A qué técnica de diseño, de las conocidas por usted, pertenece el algoritmo de *Kruskal*? Justifique su respuesta.

Ejercicio 8:

Sea el dígrafo:



- a) Ejecutar el algoritmo de *Floyd*, ¿Qué le permite obtener este algoritmo?
- b) Realice en pseudo-código una rutina que resuelva el mismo problema que el algoritmo de *Floyd*, pero usando el algoritmo de *Dijkstra*. (suponga la rutina *Dijkstra* ya programada).
- c) Analizar el esfuerzo máximo, medido en tiempo, expresándolo en notación O , del algoritmo desarrollado en b), considerando que tiene n vértices y m arcos.
- d) ¿Cuándo conviene usar el algoritmo obtenido en b) en lugar del usado en a)? ¿Cuándo no conviene?
- e) ¿A qué técnica de diseño, de las conocidas por usted, pertenece el algoritmo de *Floyd*? Justifique su respuesta.

Ejercicio 9:

Utilizando el dígrafo del ejercicio anterior como entrada:

- a) Ejecutar el algoritmo de *Warshall*, ¿Qué le permite obtener este algoritmo?
- b) A partir de la salida del algoritmo de *Floyd* ¿puede resolver el problema que resuelve el algoritmo de *Warshall*?
- c) ¿Cómo modificaría el algoritmo ejecutado en a) para que permita recuperar un camino? Realice una rutina en pseudo-código que le permita mostrar un camino entre un par de vértices dados.
Si entre un par de vértices existen varios caminos, ¿cuál de los caminos recuperaría su algoritmo?
- d) ¿A qué técnica de diseño, de las conocidas por usted, pertenece el algoritmo de *Warshall*? Justifique su respuesta.