Estructuras de Datos y Algoritmos

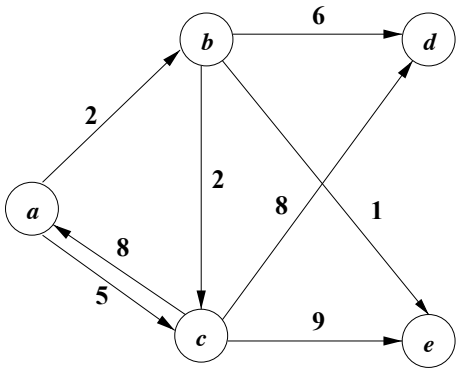
Año 2023

Práctico 7: *Diseño de Algoritmos*

Licenciatura en Ciencias de la Computación - Profesorado en Ciencias de la Computación (plan viejo)

Ejercicio 1:

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:

a) Ejecutar el algoritmo que permita obtener el viaje más barato desde la ciudad c a cada una de las demás ciudades de la región. Muestre paso a paso la ejecución.

b) Explicar cómo modificaría el algoritmo anterior si lo único que quiere obtener es el viaje de menor costo entre un determinado par de ciudades (`viaje-barato(x,y)`).

c) Analizar el esfuerzo máximo, medido en tiempo, expresándolo en notación O, considerando que tiene n vértices y a arcos.

d) ¿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.

e) ¿A qué técnica de diseño, de las conocidas por usted, pertenece el algoritmo que ejecutó en *a)*? Explicar cómo llega a esa conclusión.

Ejercicio 2:

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	-

a) Graficar el grafo que se obtiene a partir de dicha matriz.

b) 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.

c) 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 3:

Considerar el dígrafo del Ejercicio 1 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:

a) Obtener el recorrido mas corto que pasa por todas las ciudades, para ello utilizar el algoritmo de *Prim* partiendo desde c. 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?

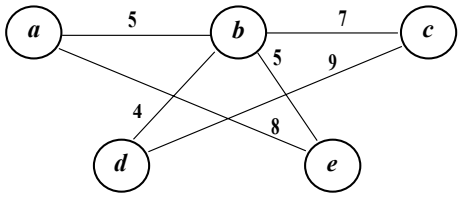
b) Analizar el esfuerzo máximo, medido en tiempo, expresándolo en notación O, considerando que tiene n vértices y a arcos.

c) 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 4:

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?

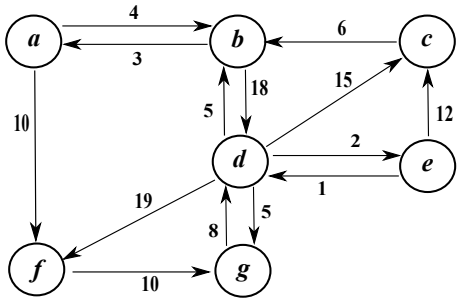
b) Analizar el esfuerzo máximo, medido en tiempo, expresándolo en notación O, considerando que tiene n vértices y a arcos.

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 5:

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 a 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 6:

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.

Ejercicio 7:

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]>...>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 div d[i]
    m := m mod d[i]
  od
end function
```

b)

```
Pre-condiciones: d[1]< d[2]<...< 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,j] := ∞ 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[i] := m[n,k]
  if m[n,k]<>∞ 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?