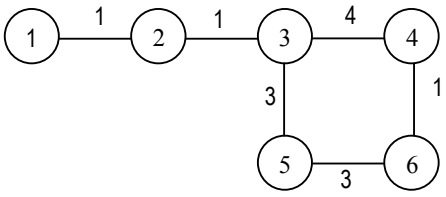
Práctico 8

Algoritmos Voraces (Greedy)

EJERCICIO 1 (I)

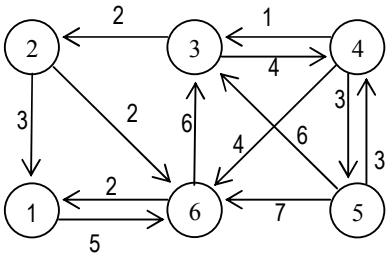
Dado el siguiente Grafo



Indique cuáles son los caminos más cortos con origen en el vértice 1, determinados por el algoritmo de Dijkstra.

EJERCICIO 2 (R)

El siguiente grafo



Tiene 3 caminos de costo mínimo entre los vértices 5 y 1.

a) Determine dichos caminos.
b) ¿Cuál de ellos será encontrado por el algoritmo de Dijkstra? Justifique.
c) Modifique el algoritmo de Dijkstra para determinar el número de caminos más cortos entre un par de vértices, así como su costo.

EJERCICIO 3 (I)

a) Implemente el algoritmo de Dijkstra (para grafos dirigidos con aristas de costo positivo).
b) Calcule el tiempo de ejecución, en el peor caso, para las representaciones de un Grafo mediante matriz de adyacencia y listas de adyacencia.
c) Explique cuales precondiciones deben cumplirse para que el algoritmo funcione correctamente.
d) Demuestre, bajo las condiciones impuestas, porque el algoritmo obtiene el camino de costo mínimo entre dos nodos de un grafo.

EJERCICIO 4 (C)

Utilice el algoritmo del ejercicio anterior para implementar una función que encuentre el camino más corto de todos los vértices a uno fijado.

EJERCICIO 5 (I)

Sea $\{J_1, ..., J_n\}$ un conjunto de trabajos que deben ser ejecutados secuencialmente en un mismo procesador. Cada J_i consume una unidad de tiempo de procesador y produce una ganancia $r_i > 0$ en el caso en que sea concluido antes de su tiempo límite t_i. Si Ji se concluye luego de su t_i, no produce ganancia. El problema consiste en determinar el orden de procesamiento de los trabajos que produzca mayor ganancia. Considere dados los valores $r_1,..., r_n$ y $t_1,..., t_n$.

Formular una estrategia ávida (Greedy) que resuelva el problema, justificando por qué conduce al óptimo.

EJERCICIO 6 (I)

Sean n programas de largo $l_1, ..., l_n$ que serán almacenados en una cinta, y serán recuperados con frecuencias f_i. Si se almacenan en el orden $i_1, ..., i_n$, la probabilidad de que se solicite el j-ésimo es:

$$\frac{f_j}{\sum_{i=1}^{n} f_i}$$

Y el tiempo necesario para recuperar el j-ésimo elemento es: $\sum_{k=1}^{j} l_k$.

Entonces el tiempo de recuperación esperado (*TRE*), es:

$$\frac{\sum_{j=1}^{n}\left(f_j \sum_{k=1}^{j} l_k\right)}{\sum_{i=1}^{n} f_i}$$

a) Mostrar que almacenar los programas en orden creciente de l_i (largo) no necesariamente conduce al *TRE* óptimo.
b) Mostrar que almacenar los programas en orden decreciente de f_i (frecuencia) no necesariamente conduce al *TRE* óptimo.

Buscar una estrategia Greedy que permita encontrar el ordenamiento con *TRE* óptimo. Justifique su estrategia.

EJERCICIO 7 (R)

Se tienen N listas ordenadas de largos l_1,.....,l_N. Éstas deben ser intercaladas ordenadamente de a pares (de a dos listas) hasta alcanzar una lista única. Se dice que el costo de intercalar 2 listas de largos l_i y l_j es $(l_i + l_j)$.

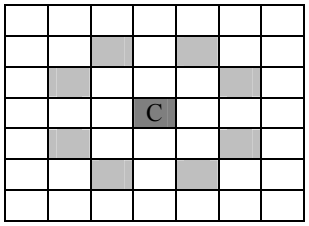
Se desea encontrar un método para elegir el orden en que las listas son intercaladas de manera que se minimice el costo total de intercalar el conjunto de listas.

a) Encontrar una técnica voraz (Greedy) que logre el objetivo.
b) Mostrar por qué funciona.
c) Implementar la técnica presentada.

Nota: El costo de intercalar 3 listas de largos l_1, l_2 y l_3 depende de la manera en que se seleccione el orden de intercalación de éstas. Si se desea intercalar primero l_1 y l_3 y luego intercalar el resultado con l_2 se obtiene un costo total de $((l_1 + l_3) + ((l_1 + l_3) + l_2))$.

EJERCICIO 8 (R)

Sea un tablero de ajedrez de n casillas de lado. Los posibles movimientos del caballo C son los que están marcados en la siguiente figura:



Dadas dos coordenadas (x_1,y_1) (x_2, y_2) en un tablero de ajedrez, hallar la ruta mínima que debería recorrer un caballo de ajedrez, en caso de que sea posible, para ir de (x_1,y_1) a (x_2,y_2).

a) Diseñar un algoritmo utilizando la técnica Greedy que resuelva el problema.
b) Defina y explique las estructuras de datos que utiliza.
c) Demuestre que la estrategia utilizada conduce al óptimo buscado.

EJERCICIO 9 (R)

Sea un Grafo dirigido y acíclico G = (V, E) con N vértices. Cada arista e tiene costo dado por $c(e)$. Dados dos vértices u y v se desea hallar el camino simple de costo **máximo** entre ellos.

Suponga que para resolver el problema se decide hallar el camino de costo **mínimo** entre u y v en otro grafo con costos dados por $c'(e) = k - c(e)$ donde $k = \sum_{e \in E} c(e)$. Analice la correctitud de este método.

EJERCICIO 10 (C)

Dado un Grafo ponderado cuyas aristas pueden tener costos negativos:
a) Explique porqué el algoritmo de Dijkstra no funciona para estos casos.
b) Defina un algoritmo que pueda cumplir con estas condiciones. Diseñe una estrategia para verificar que las precondiciones se cumplan.
c) Determine su tiempo de ejecución.

EJERCICIO 11 (R)

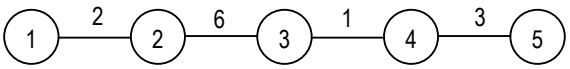
Muestre un Grafo que tenga dos árboles de cubrimiento mínimo diferentes.

EJERCICIO 12 (C)

Construya un grafo con 8 vértices y 11 aristas, de manera que el algoritmo de Kruskal tenga que examinar cada arista antes de producir un árbol de cobertura mínima. Justifique.

EJERCICIO 13 (C)

Sea el siguiente árbol, resultado del algoritmo de Kruskal:



Dibuje el grafo original no dirigido sabiendo que:

- todos los costos son positivos
- la suma de los costos es 27
- el grafo tiene 7 aristas
- cada arista pertenece, al menos, a un ciclo

EJERCICIO 14 (I)

a) Explique las precondiciones que el grafo debe cumplir para poder utilizar el algoritmo de Prim para obtener un árbol de cubrimiento de costo mínimo.
b) Implemente el algoritmo de Prim para la representación de grafos con listas de adyacencia utilizando un Heap para mantener la información sobre las distancias entre los vértices agregados y los no agregados al árbol.
c) Demuestre, bajo las condiciones impuestas, porque el algoritmo funciona correctamente.

EJERCICIO 15 (R)

Dado un Grafo no dirigido G = (V, E) con costos sobre las aristas y un conjunto A contenido en E, deseamos obtener de todos los árboles de cubrimiento (si existe alguno) que contengan todas las aristas de A el que sea de costo mínimo.

Diseñe un algoritmo basado en Kruskal para realizar esta operación y calcule su orden de ejecución en el peor caso.