Algoritmos y Programación II (75.41, 95.15) – Curso Buchwald

4.to parcialito – 13/12/2021

Resolvé los siguientes problemas en forma clara y legible. Podés incluir tantas funciones auxiliares como creas necesarias.

Debés resolver 3 de los siguientes ejercicios. Los ejercicios a resolver están definidos por tu padrón. Podés revisar en esta planilla.

1. Dado un grafo G no dirigido, se desea implementar un algoritmo usando backtracking que permita numerar los V vértices con los números del 1 a |V|, de forma tal que no haya dos vértices adyacentes numerados con números consecutivos (por ejemplo, el vertice `"a"` no puede numerarse con 2, si existe un vértice adyacente que haya sido numerado previamente con 1 o 3) y que no haya vertices numerados con el mismo número. De ser posible resolver el problema, el algoritmo debe devolver un diccionario que tenga por clave el vértice y por valor el número asignado.

   El diccionario resultante debera tener el siguiente formato:

   ```
   {
       "a": 1,     // El vertice "a" fue numerado con el numero 1
       "c": 3,     // El vertice "c" fue numerado con el numero 3
       "d", 2,     // El vertice "d" fue numerado con el numero 2
       "b": 4      // El vertice "b" fue numerado con el numero 4
   }
   ```

2. Dado un grafo G no dirigido, se desea implementar un algoritmo usando backtracking que permita numerar las E aristas con los números del 1 a |E|, de forma tal que **no haya dos aristas que compartan nodo** numeradas con números consecutivos (por ejemplo, la arista `("a", "b")` no puede numerarse con 2, si existe una arista que salga de `"a"` o de `"b"` que haya sido numerada previamente con 1 o 3) y que no haya aristas numeradas con el mismo número. De ser posible resolver el problema, el algoritmo debe devolver un diccionario que tenga por clave el id de la arista (se posee una primitiva `obtener_id_arista(v1,v2)` que recibe los nodos que forman la arista y devuelve el id de la arista) y por valor el número asignado.

   El diccionario resultante debera tener el siguiente formato:

   ```
   {
       "a": 1,  // La arista con id "a" fue numerado con el numero 1
       "b": 3,  // La arista con id "b" fue numerado con el numero 3
       "c": 2,  // La arista con id "c" fue numerado con el numero 2
       "d": 4   // La arista con id "d" fue numerado con el numero 4
   }
   ```

   Nota: De ser necesario, suponer que el TDA Grafo posee una primitiva `obtener_aristas` que devuelva una lista con todas los ids de las aristas y una primitiva `obtener_nodos_arista(id)` que recibe un id y devuelve los dos nodos que la conforman.

3. Realizar el seguimiento del Algoritmo de Prim (empezando por el vértice `F`) para el siguiente grafo:

	A	B	C	D	E	F
A	0	5	2	3	0	3
B	5	0	1	0	0	0
C	2	1	0	4	0	2
D	3	0	4	0	0	0
E	0	0	0	0	0	9
F	3	0	2	0	9	0

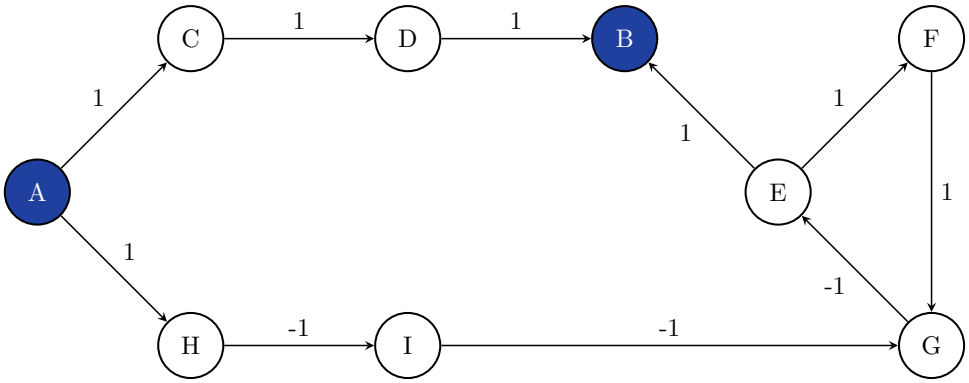
4. Realizar el seguimiento del Algoritmo de Kruskal para el siguiente grafo:

	A	B	C	D	E	F
A	0	5	2	3	0	3
B	5	0	1	0	0	0
C	2	1	0	4	0	2
D	3	0	4	0	0	0
E	0	0	0	0	0	9
F	3	0	2	0	9	0

5. Definir si las siguientes afirmaciones son verdaderas o falsas. Justificar utilizando ejemplos y/o contraejemplos.

   a. Bellman-Ford permite resolver el problema de caminos mínimos para cualquier grafo con aristas negativas.

b. Dado el siguiente grafo, de los algoritmos vistos en clase, el único que permite resolver el problema de caminos mínimos de A hasta B es Dijkstra.

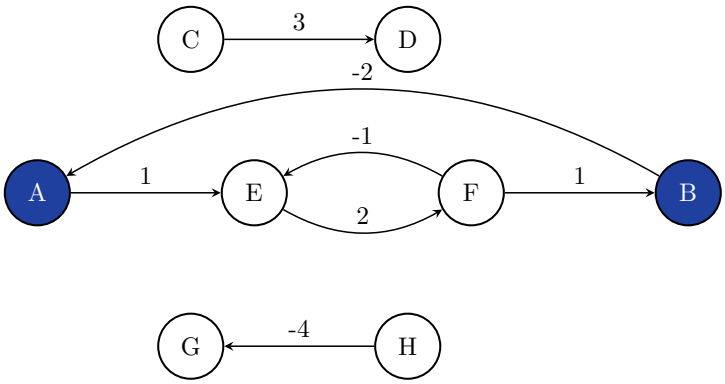


c. El árbol de tendido mínimo resultante al aplicar el algoritmo de Kruskal permite resolver el problema de caminos mínimos

6. Definir si las siguientes afirmaciones son verdaderas o falsas. Justificar utilizando ejemplos y/o contraejemplos.

a. Dijkstra permite resolver el problema de caminos mínimos para cualquier grafo que no contenga ciclos negativos.

b. Dado el siguiente grafo, de los algoritmos vistos en clase, el único que permite resolver el problema de caminos mínimos de A hasta B es Bellman Ford.



c. El árbol de tendido mínimo resultante al aplicar el algoritmo de Prim permite resolver el problema de caminos mínimos