Progettazione di Algoritmi **Anno Accademico 2023–2024**

Esercizi

Ugo Vaccaro

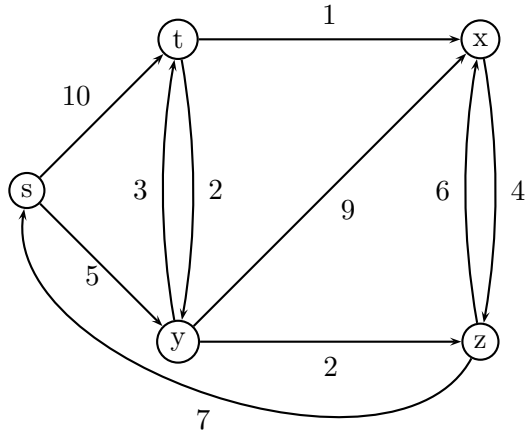
Esercizi su Grafi: Parte Seconda

N.B. Si ricorda che ogni algoritmo và accompagnato da una argomentazione sul perchè calcola correttamente l'output e da un'analisi della sua complessità di tempo. Inoltre, si possono usare algoritmi visti a lezione (come BFS, DFS, etc.) senza necessariamente riportare il relativo pseudocodice, purchè lo si menzioni esplicitamente. In generale, di ogni algoritmo è preferibile presentare il relativo pseudocodice. Tuttavia, anche una sola descrizione a parole dell'idea dell'algoritmo (purchè precisa e corretta) verrà accettata all'esame.

1. *Esercizio*: Si presenti l'algoritmo di Dijkstra per il calcolo dei cammini minimi in un grafo, si argomenti la sua correttezza e se ne valuti la sua complessità di tempo.

◇

2. *Esercizio*: Si esegua l'algoritmo di Dijkstra per il calcolo dei cammini minimi dal nodo s sul seguente grafo. Si esplicitino in dettaglio i passi che l'algoritmo esegue (ovvero, NON è sufficiente riportare semplicemente l'albero dei cammini minimi risultanti dall'esecuzione dell'algoritmo).



◇

3. *Esercizio*: Sia $G = (V, E)$ un grafo diretto, con lunghezze $\ell(e) > 0$ per ogni arco $e \in E$. Si progetti e si analizzi (e si argomenti la correttezza) un algoritmo che presi in input un vertice $s \in V$ ed un arco $(u, v) \in E$, determini: **1.** se tra tutti i cammini di lunghezza minima da s a v ne esiste almeno uno che usi l'arco (u, v); **2.** se tra tutti i cammini di lunghezza minima da s a v ne esiste almeno uno che non usi l'arco (u, v).

◇

4. *Esercizio*: Dato un grafo orientato $G = (V, E)$, con lunghezze $\ell(e) > 0$ per per ogni arco $e \in E$, e due nodi $v, t \in V$.

   (a) Si proponga un algoritmo che determini, oltre ad un cammino minimo p_1 da v a t che supponiamo essere $p_1 = (v, s_1, ...s_n, t)$, un secondo cammino p_2 che sia il cammino di lunghezza minima tra quelli che congiungono v con t e non contengono l'arco (s_n, t).

   (b) Si provi la correttezza e si valuti la complessità dell'algoritmo proposto.

   (c) Si proponga un algoritmo che determini, oltre ad un cammino minimo p_1 da v a t, un secondo cammino p_2 che sia il cammino di lunghezza minima tra tutti i cammini che congiungono v con t e sono distinti da p_1.

◇

5. *Esercizio*: Sia $G = (V, E)$ un grafo diretto in cui ad ogni nodo $u \in V$ è associato un peso $w(u) > 0$. Descrivere un algoritmo per calcolare i cammini di peso minimi da un nodo sorgente $s \in V$ a tutti i nodi $v \in V$. In questa situazione, il peso di un cammino è definito come la somma dei pesi dei nodi che appaiono nel cammino. (Sugg.: si riduca il problema a quello di calcolare i cammini di peso minimi in un opportuno grafo con pesi su archi).

◇

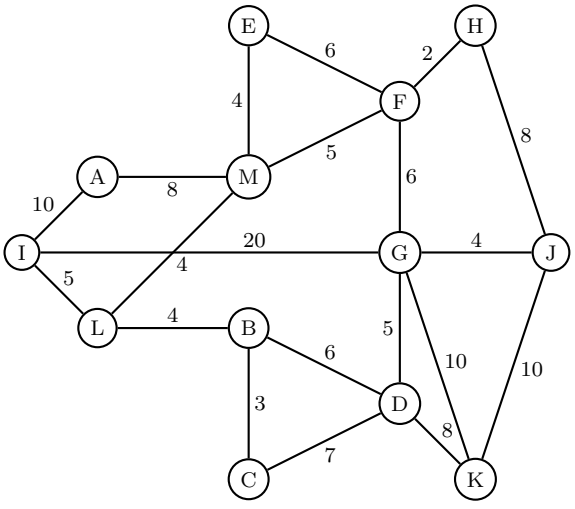
6. *Esercizio*: Descrivere l'algoritmo di Prim per la costruzione di un MST e provarne con precisione la correttezza.

◇

7. *Esercizio*: Descrivere l'algoritmo di Kruskal per la costruzione di un MST e provarne con precisione la correttezza.

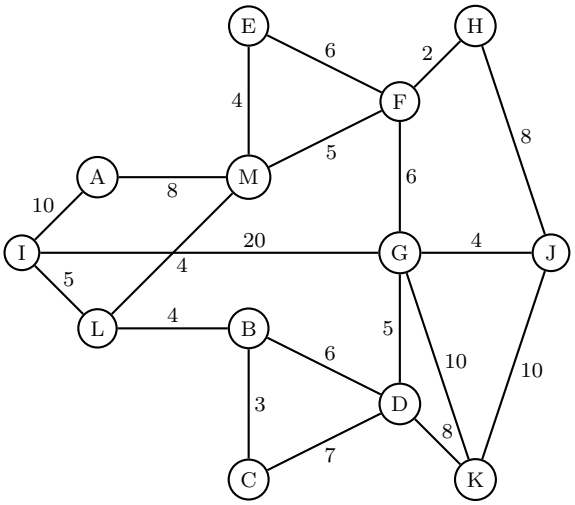
◇

8. *Esercizio*: Dato il grafo rappresentato in figura, calcolare un MST di esso applicando l'algoritmo di Prim a partire dal nodo I. Si descrivano con precisione le computazioni effettuate dall'algoritmo passo dopo passo.



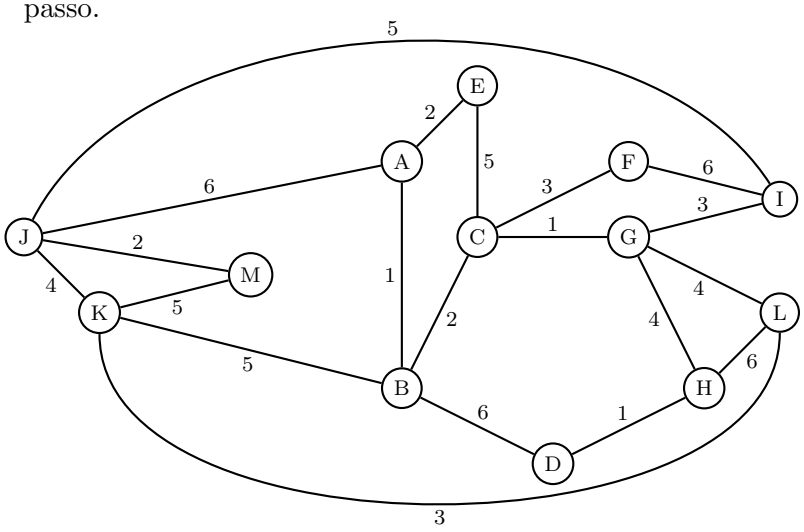
◇

9. *Esercizio*: Dato il grafo rappresentato in figura, calcolare un MST di esso applicando l'algoritmo di Kruskal. Si descrivano con precisione le computazioni effettuate dall'algoritmo passo dopo passo.



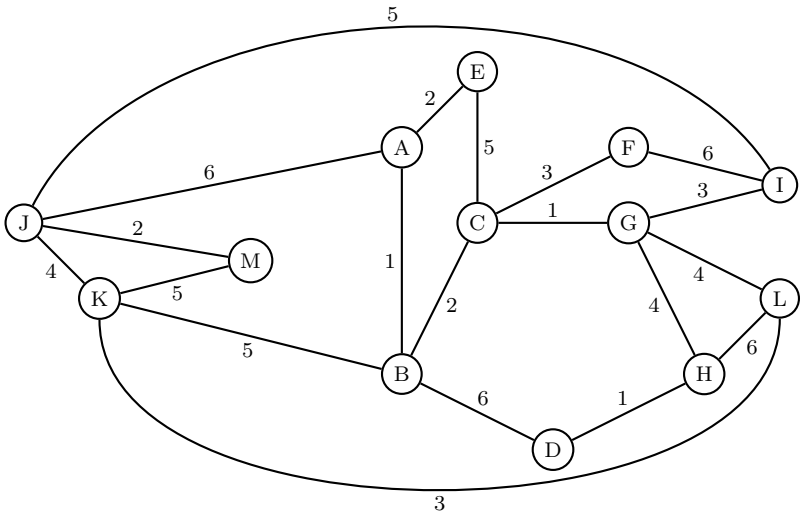
◇

10. *Esercizio*: Dato il grafo rappresentato in figura, calcolare un MST di esso applicando l'algoritmo di Kruskal. Si descrivano con precisione le computazioni effettuate dall'algoritmo passo dopo passo.



◇

11. *Esercizio*: Dato il grafo rappresentato in figura, calcolare un MST di esso applicando l'algoritmo di Prim a partire dal vertice I. Si descrivano con precisione le computazioni effettuate dall'algoritmo passo dopo passo.



◇

12. *Esercizio*: Si proponga un algoritmo che, dato in input un grafo non non orientato $G = (V, E)$, pesi $w(e)$ per ogni arco $e \in E$, ed un arco $(u, v) \in E$, determini l'esistenza o meno di un MST per G non contenente l'arco (u, v). Si motivi la correttezza e si calcoli la complessit'a dell' algoritmo proposto. Si riconsiderino i punti precedenti nel caso in cui si voglia determinare l'esistenza di un MST per il grafo G contenente l'arco (u, v).

◇

13. *Esercizio*: Sia $G = (V, E)$ un grafo non orientato e pesato, con archi di costo $c(e)$, per ogni $e \in E$.

    (a) Si descriva brevemente un algoritmo efficiente (per esempio tra quelli visti a lezione) che determini un Minimum Spanning Tree per G e se ne valuti la complessità.

    (b) Si provi o si refuti la seguente proprietà: dato un Minimum Spanning Tree per G questo contiene almeno un arco tra quelli aventi peso minimo in G.

    (c) Si provi o si refuti la seguente proprietà: dato un Minimum Spanning Tree per G questo contiene un cammino minimo per ogni coppia di nodi in G.

◇

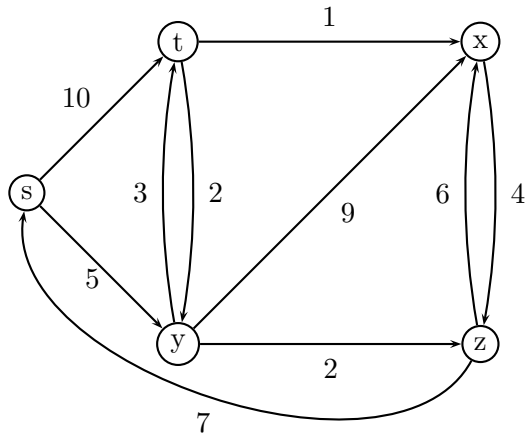
14. *Esercizio*: Sia $G = (V, E)$ un grafo non orientato e pesato, con archi di costo $c(e)$, per ogni $e \in E$. Si provi la seguente affermazione:

    sia $\emptyset \neq S \subset V$ *un sottoinsieme dei nodi, e sia* $e = (u, v)$ *l'arco di* costo minimo *con un estremo in* S *e l'altro in* $V - S$. *Allora* ogni *MST contiene l'arco* e.

◇

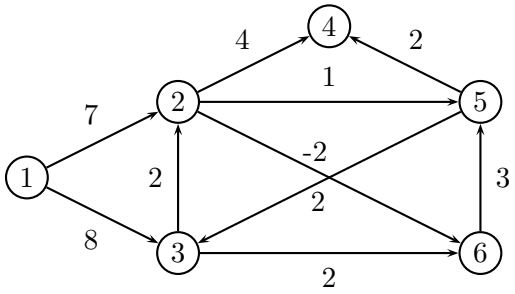
◇

15. *Esercizio*: Si esegua l'algoritmo di Bellman-Ford per il calcolo del cammino minimo da s a t nel grafo di sotto riportato. Si descrivano con precisione le computazioni effettuate dall'algoritmo passo dopo passo.



◇

16. *Esercizio*: Si esegua l'algoritmo di Bellman-Ford per il calcolo dei cammini minimi da 1 al nodo 6. Si descrivano con precisione le computazioni effettuate dall'algoritmo passo dopo passo.



(L'arco da 5 a 3 ha peso -2 e l'arco da 2 a 6 ha peso 2.)

17. *Esercizio*: Derivare l'equazione di ricorrenza per il costo di cammini minimi in grafi, argomentando i relativi passi. Descrivere un algoritmo per il calcolo dell'equazione prima ottenuta.

◇

18. *Esercizio*: Derivare l'algoritmo per scoprire se un grafo diretto con lunghezze sugli archi contiene cicli di costo totale negativo e provarne la correttezza.

◇

19. *Esercizio*: Derivare l'algoritmo per trovare in un grafo diretto un ciclo di costo totale negativo (se esso esiste) e provarne la correttezza.