Università degli Studi di Padova

Algoritmi Avanzati

A.A. 2019/2020

31 agosto 2020, 9:30–11:30

Prima Parte: domande di teoria

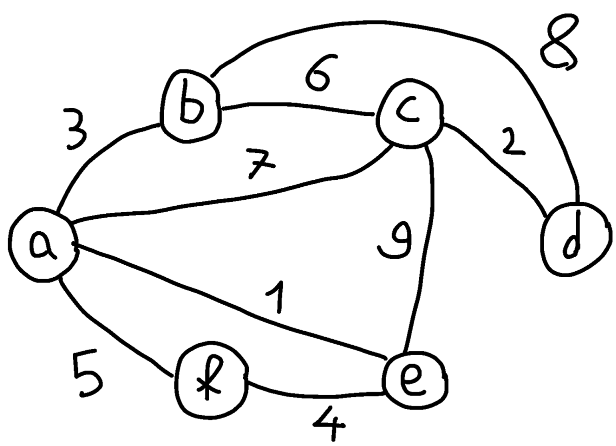
Domanda 1 (6 punti) Si consideri il seguente grafo pesato, rappresentato tramite una matrice di adiacenza dove ogni valore numerico rappresenta il peso del lato corrispondente, e dove il simbolo '$-$' indica l'assenza del lato tra i nodi corrispondenti.

	a	b	c	d	e	f
a	-	3	7	-	1	5
b		-	6	8	-	-
c			-	2	9	-
d				-	-	-
e					-	4
f						-

1. Disegnare il grafo.
2. Elencare i lati del minimum spanning tree nell'ordine in cui sono selezionati dall'algoritmo di Kruskal.
3. Elencare i lati del minimum spanning tree nell'ordine in cui sono selezionati dall'algoritmo di Prim a partire dal nodo c.

Soluzione:

1.



2. $(a, e), (c, d), (a, b), (e, f), (b, c)$.
3. $(c, d), (b, c), (a, b), (a, e), (e, f)$.

Domanda 2 (6 punti) Dimostrare che un grafo con n vertici e con un minimum cut di cardinalità t ha almeno $tn/2$ lati.

Soluzione: Visto a lezione.

Seconda Parte: risoluzione di problemi

Esercizio 1 (10 punti) Il diametro D di un grafo $G = (V, E)$ (non pesato, non diretto) è definito come la massima distanza tra due vertici in G, dove per distanza si intende il numero di lati nel cammino più breve tra i due vertici. Usando in modo opportuno la BFS:

1. Progettare e analizzare un semplice algoritmo per il calcolo del diametro di G.

2. Progettare e analizzare un algoritmo veloce di 2-approssimazione del diametro di G, cioè un algoritmo che ritorni un valore $\bar{D}$ tale che $D/2 \leq \bar{D} \leq D$.

Soluzione:

1. Si ricordi che BFS(G, v) partiziona i vertici della componente connessa contenente v in livelli L_i in base alla loro distanza i dal vertice di partenza v. Quindi, se supponiamo di modificare l'algoritmo BFS in modo che restituisca l'indice dell'ultimo livello generato (cioè l'altezza dell'albero), basta invocare BFS(G, v) n volte, una per ogni vertice $v \in V$, memorizzando il massimo valore di livello restituito. Se il grafo non risulta connesso, l'algoritmo deve restituire ∞. La complessità di questa strategia è $O(n(n + m))$.

2. Se il grafo non risulta connesso, l'algoritmo deve restituire ∞. Altrimenti, è sufficiente invocare BFS(G, v) una singola volta da un qualsiasi vertice v, restituendo l'indice dell'ultimo livello generato (cioè l'altezza dell'albero): la massima distanza tra due vertici in G è il cammino tra due foglie in due rami distinti dell'albero, che quindi può essere al più il doppio dell'altezza dell'albero. La complessità di questa strategia è $O(n + m)$.

Esercizio 2 (10 punti) Dimostrare che il Traveling Salesman Problem (TSP) è NP-hard con una riduzione dal problema del circuito Hamiltoniano.

Soluzione: Data una istanza di input $G = (V, E)$ per il problema del circuito Hamiltoniano, possiamo creare in tempo polinomiale la seguente istanza di input per il TSP: $\langle G' = (V, E'), c, k\rangle$, con G' completo e pesato,

$$c(e \in E') = \begin{cases} 1 & \text{se } e \in E \\ \infty & \text{altrimenti} \end{cases}$$

e $k = n$. Abbiamo quindi che

1. Se esiste un ciclo Hamiltoniano in G, allora l'algoritmo per il TSP eseguito su G' restituisce un ciclo Hamiltoniano di costo n.

2. Se non esiste un ciclo Hamiltoniano in G, un qualsiasi ciclo Hamiltoniano in G' deve avere almeno un lato non in G, quindi di peso ∞. Quindi in questo caso l'algoritmo per il TSP eseguito su G' restituisce un ciclo Hamiltoniano di costo maggiore di n.