Università degli Studi di Padova

Algoritmi Avanzati

A.A. 2019/2020

15 luglio 2020, 10:00–12:00

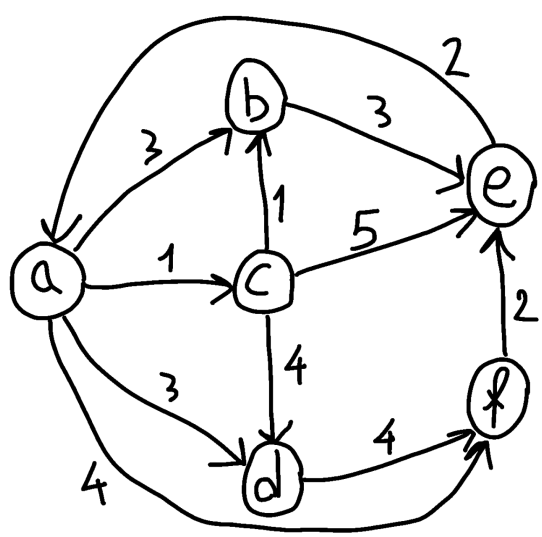
Prima Parte: domande di teoria

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

	a	b	c	d	e	f
a	-	3	1	3	-	4
b	-	-	-	-	3	-
c	-	1	-	4	5	-
d	-	-	-	-	-	4
e	2	-	-	-	-	-
f	-	-	-	-	2	-

1. Disegnare il grafo.
2. Elencare le lunghezze dei cammini più brevi tra il nodo a e tutti gli altri nodi del grafo, nell'ordine in cui queste sono determinate dall'algoritmo di Dijkstra.
3. Qual'è la complessità dell'algoritmo di Dijkstra quando implementato con heap?

Soluzione:

1. 

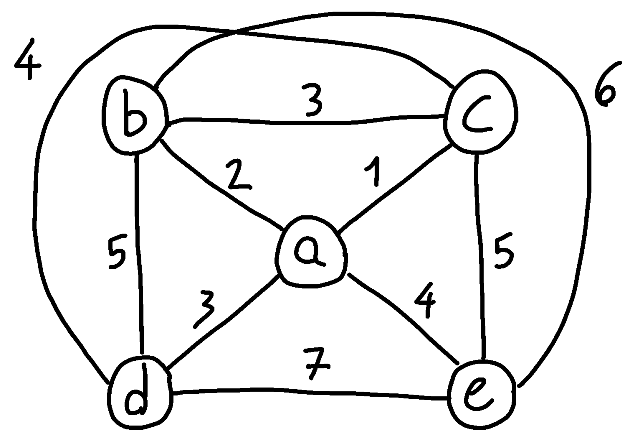
2. $a-c:1$, $a-b:2$, $a-d:3$, $a-f:4$, $a-e:5$.
3. $O((m+n)\log n)$.

Domanda 2 (6 punti) Si consideri il seguente grafo completo pesato, rappresentato tramite una matrice di adiacenza dove ogni valore numerico rappresenta il peso del lato corrispondente.

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

1. Disegnare il grafo.
2. Elencare i lati del minimum spanning tree nell'ordine in cui sono selezionati dall'algoritmo di Prim a partire dal nodo a.
3. Siccome i pesi dei lati soddisfano la disuguaglianza triangolare, si può applicare l'algoritmo di 2-approssimazione APPROX_T_TSP visto in classe per il TSP. Considerando i nodi ordinati secondo l'ordine alfabetico, fornire l'output dell'algoritmo APPROX_T_TSP quando eseguito su questo grafo.

Soluzione:

1. 

2. $(a, c), (a, b), (a, d), (a, e)$.
3. Considerando i nodi ordinati secondo l'ordine alfabetico, l'algoritmo APPROX_T_TSP esegue l'algoritmo di Prim a partire dal nodo a. Un possibile output è quindi a, c, b, d, e, a.

Seconda Parte: risoluzione di problemi

Esercizio 1 (10 punti) Un grafo $G = (V, E)$ si dice *bipartito* se l'insieme di vertici V può essere partizionato in due sottoinsiemi X e Y tali che ogni lato di E incide su un vertice di X e uno di Y. Progettare e analizzare un algoritmo efficiente che determini se un grafo G è bipartito.

Soluzione: Si osservi che G è bipartito se e solo se ciascuna sua componente connessa è bipartita. Basta quindi controllare che ogni componente connessa sia bipartita.

Si consideri quindi un'arbitraria componente connessa. Un possibile algoritmo si basa sull'uso della BFS:

1. Eseguire una BFS da un vertice qualsiasi della componente connessa

2. Colorare i vertici del BFS tree blu se sono in un livello pari, rosso se sono in un livello dispari (quindi, X corrisponde a blu e Y a rosso, o viceversa)

3. Controllare ogni lato per vedere se ne esiste uno che incide su due vertici dello stesso colore: se sì, allora return FALSE, altrimenti return TRUE

Correttezza dell'algoritmo: se non esiste nessun lato che incide su due vertici dello stesso colore, allora si ha una corretta bipartizione e quindi la componente è bipartita. Altrimenti, la componente non può essere bipartita: se lo fosse, non avrebbe lati che incidono su vertici dello stesso colore.

Per quanto riguarda la complessità, l'algoritmo equivale a una visita di tutto il grafo che, come visto a lezione, ha complessità $O(m+n)$.

Esercizio 2 (10 punti) $m = 6n \ln n$ jobs vengono assegnati in modo casuale a n processori. (Nota: si ricordi che $\ln x = \log_e x$.) Si consideri un certo processore p, e si dimostri che, con alta probabilità nel parametro n, il processore p non riceve più di $12 \ln n$ jobs. (Suggerimento: definire una opportuna variabile indicatore per ogni job, e applicare il seguente bound di Chernoff.)

Theorem 1. *Siano $X_1, X_2, \ldots, X_n$ variabili indicatore indipendenti con $E[X_i] = p_i, 0 < p_i < 1$. Sia $X = \sum_{i=1}^{n} X_i$ e $\mu = E[X]$. Allora, per ogni $0 < \delta \leq 1$,*

$$Pr(X > (1+\delta)\mu) \leq e^{-\frac{\delta^2 \mu}{3}}.$$

Soluzione: Sia X_i, con $i = 1, 2, \ldots, 6n \ln n$, una variabile indicatore che vale 1 se l'i-esimo job viene assegnato al processore p. Ovviamente, $Pr(X_i = 1) = 1/n$, e le X_i sono indipendenti. Il numero di job ricevuti dal processore p è quindi $X = \sum_{i=1}^{6n \ln n} X_i$. Per applicare il bound di Chernoff poniamo $12 \ln n$ uguale a $(1+\delta)\mu$; siccome $\mu = E[X] = \sum_{i=1}^{6n \ln n} E[X_i] = (6n \ln n)/n = 6 \ln n$, otteniamo $\delta = 1$. Quindi

$$Pr(X > 12 \ln n) \leq e^{-\frac{6 \ln n}{3}} = 1/n^2,$$

cioè

$$Pr(X \leq 12 \ln n) > 1 - 1/n^2.$$