Lezione del 7/12/04 [2 ore, in AULA C, dalle 11:00 alle 13:00]

di: *Algoritmi & Laboratorio (Modulo 1)*

"Riassunto" della lezione:

Sono stati svolti i seguenti esercizi (alla lavagna).

1. Dato il grafo in fig.(a):

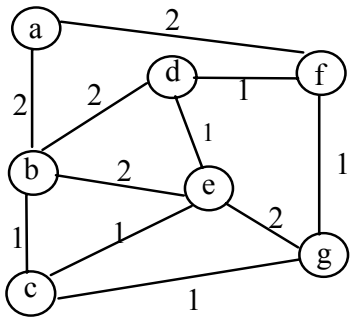


fig.(a)

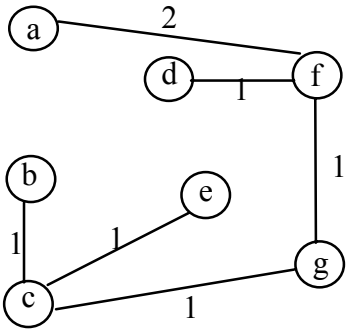
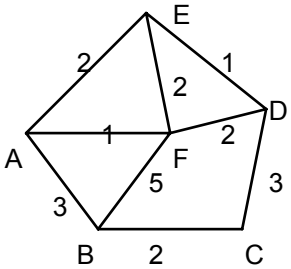


fig.(b)

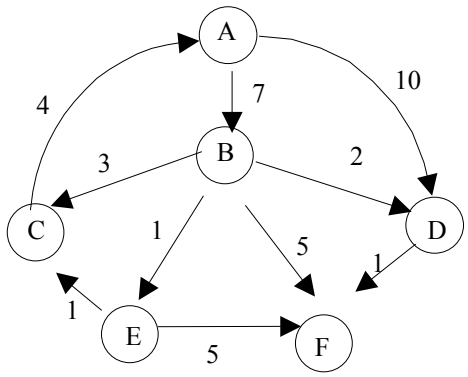
ordinare gli archi in modo che l'algoritmo di Kruskal fornisca l'albero di copertura minimo in fig.(b).

2. Costruire per il seguente grafo:



un albero di copertura minimo usando l'algoritmo di Kruskal e un albero di copertura minimo (non necessariamente lo stesso) usando l'algoritmo di Prim. Per ognuno dei due algoritmi mostrare i passi della costruzione.

3. Sia dato il seguente grafo orientato e pesato:



applicando l'algoritmo di Dijkstra si determinino i pesi dei cammini minimi che collegano il vertice A con tutti gli altri vertici.
Si compilino le seguenti tabelle indicando ad ogni iterazione del ciclo esterno, nella prima tabella, come si modifica la stima della distanza di ogni vertice dal vertice A e, nella seconda, l'insieme dei vertici per cui d è la distanza effettiva.

d	A	B	C	D	E	F
0						
1						
2						
3						
4						
.						

	Insieme vertici t per cui d[t] = δ(A,t)
0	
1	
2	
3	
4	
.	

Svolgimento dell'esercizio 1.

Per svolgere l'esercizio occorre ricordare che il primo passo dell'algoritmo di Kruskal consiste nell'ordinare gli archi del grafo in ordine non decrescente rispetto al peso.

Per far si che l'algoritmo restituisca l'albero indicato nel disegno occorre quindi, a parita' di peso, mettere prima gli archi che fanno parte dell'albero.

Un possibile ordinamento che soddisfa tali requisiti e' il seguente (dove gli archi dell'albero sono indicati in grassetto):

Archi di peso 1:

(c,e)
(c,g)
(d,f)
(f,g)

(d,e)

Archi di peso 2:

(a,f)
(b,d)
(b,e)
(e,g)

Svolgimento dell'esercizio 2.

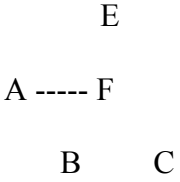
Albero di copertura minimo usando l'algoritmo di Kruskal.

Per prima cosa occorre produrre una lista degli archi in ordine non descrescente di peso:

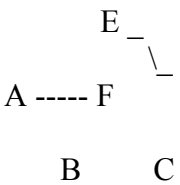
1. (A,F)
2. (D,E)
3. (A,E)
4. (B,C)
5. (D,F)
6. (E,F)
7. (A,B)
8. (C,D)
9. (B,F)

Poi occorre esaminare gli archi uno ad uno (nell'ordine precedente) e:
- includere l'arco nella soluzione, se non introduce un ciclo,
- non includerlo nella soluzione, altrimenti.

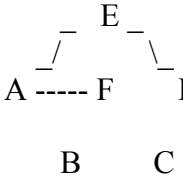
1. Arco (A,F) incluso



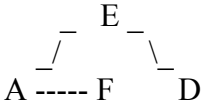
2. Arco (D,E) incluso



3. Arco (A,E) incluso



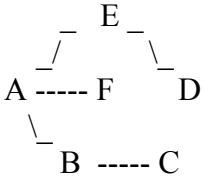
4. Arco (B,C) incluso



5. Arco (D,F) escluso

6. Arco (E,F) escluso

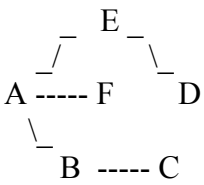
7. Arco (A,B) incluso



8. Arco (C,D) escluso

9. Arco (B,F) escluso

L'albero e' quindi il seguente:



Albero di copertura minimo usando l'algoritmo di Prim.

Costruiamo l'albero a partire dal vertica A.

La prima tabella indica, per ogni iterazione del ciclo esterno, per ogni vertice v che non appartiene alla soluzione (l'albero definitivo), il peso di un arco di peso minimo che collega tale vertice ad un vertice della soluzione (∞ **se tale arco non e' ancora stato trovato)**. Il valore di k non e' piu' riportato (si mette il segno -) quando il vertice e' ormai parte della soluzione.

La seconda indica, per ogni iterazione del ciclo esterno, l'insieme dei vertici v che sono entrati a far parte della soluzione (per i quali k[v] è il peso dell'arco incluso).

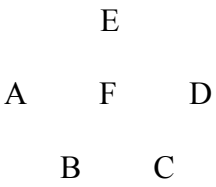
quello che viene prima secondo l'ordine alfabetico.

k	A	B	C	D	E	F
0	0	∞	∞	∞	∞	∞
1	-	3	∞	∞	2	1
2	-	3	∞	2	2	-
3	-	3	3	-	1	-
4	-	3	3	-	-	-
5	-	-	2	-	-	-
6	-	-	-	-	-	-

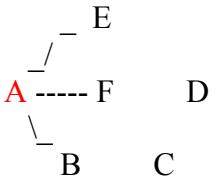
	Insieme vertici t inclusi nella soluzione
0	
1	A
2	A, F
3	A, F, D
4	A, F, D, E
5	A, F, D, E, B
6	A, F, D, E, B, C

Per completezza (ANCHE SE NON E' RICHIESTO DALL'ESERCIZIO) riportiamo nel seguito l'albero mantenuto dall'algoritmo (nel quale indichiamo SIA gli archi che fanno parte della soluzione CHE gli archi candidati) al termine di ogni iterazione (cioe' DOPO che sono state aggiornate le appetibilita' dei vertici in coda). Il punto 0 corrisponde al temine dell'inizializzazione (prima di entrare nel ciclo).

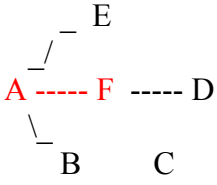
0.



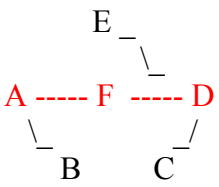
1. Vertice A incluso



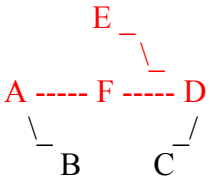
2. Vertice F e arco (A,F) inclusi



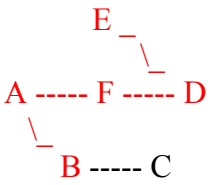
3. Vertice D e arco (D,F) inclusi



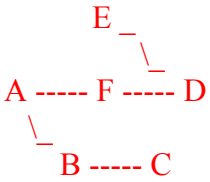
4. Vertice E e arco (D,E) inclusi



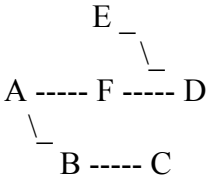
5. Vertice B e arco (A,B) inclusi



6. Vertice C e arco (B,C) inclusi



L'albero e' quindi il seguente:



Svolgimento dell'esercizio 3.

La prima tabella indica, per ogni iterazione del ciclo esterno, la stima della distanza di ogni vertice dal vertice A. I valore di d non e' piu' riportato (si mette il segno -) quando il vertice e' ormai parte della soluzione (l'albero definitivo).

La seconda tabella indica, per ogni iterazione del ciclo esterno, l'insieme dei vertici v che sono entrati a far parte della soluzione (per i quali d[v] è la distanza).

Quando nella coda con priorita' ci sono vertici con lo stesso valore minimo di d si sceglie quello che viene prima secondo l'ordine alfabetico.

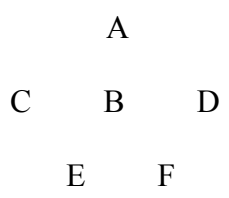
d	A	B	C	D	E	F
0	0	∞	∞	∞	∞	∞
1	-	7	∞	10	∞	∞
2	-	-	10	9	8	12
3	-	-	9	9	-	12
4	-	-	-	9	-	12
5	-	-	-	-	-	10
6	-	-	-	-	-	-

	Insieme vertici t per cui d[t] = δ(A,t)
0	
1	A
2	A, B
3	A, B, E
4	A, B, E, C
5	A, B, E, C, D
6	A, B, E, C, D, F

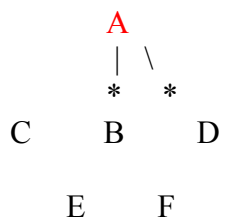
Per completezza (ANCHE SE NON E' RICHIESTO DALL'ESERCIZIO) riportiamo nel seguito l'albero mantenuto dall'algoritmo (nel quale indichiamo SIA gli archi che fanno parte della soluzione CHE gli archi candidati) al termine di ogni iterazione (cioe' DOPO che sono state aggiornate le appetibilita' dei vertici in coda). Il punto 0 corrisponde al temine dell'inizializzazione (prima di entrare nel ciclo).

Le "punte" degli archi sono indicate con un asterisco.

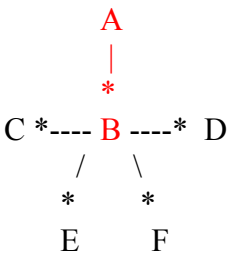
0.



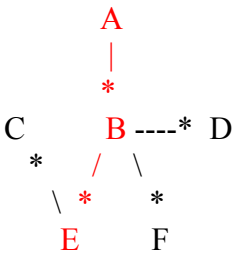
1. Vertice A incluso



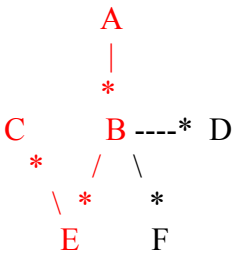
2. Vertice B e arco (A,B) inclusi



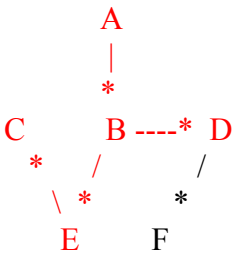
3. Vertice E e arco (B,E) inclusi



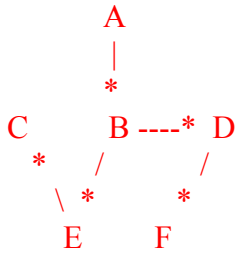
4. Vertice C e arco (E,C) inclusi



5. Vertice D e arco (B,D) inclusi



6. Vertice F e arco (D,F) inclusi



L'albero e' quindi il seguente: