Algoritmi e Strutture Dati

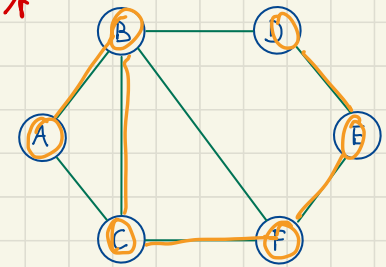
Lezione 22

18 novembre 2022

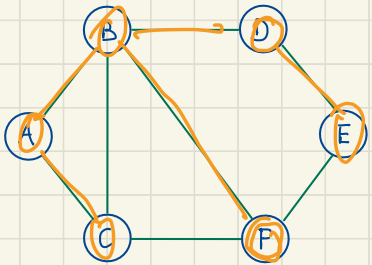
VISITA IN PROFONDITA' (Depth first search)

- Si inizia visitando un vertice s
- Partendo da s si percorre un cammino di vertici non ancora raggiunti visitandoli tutti, terminando su un vertice privo di vicini non ancora raggiunti
- Si ripete dal passo precedente partendo dall'ultimo vertice sul cammino precedente che ha un vertice non ancora raggiunto

da A

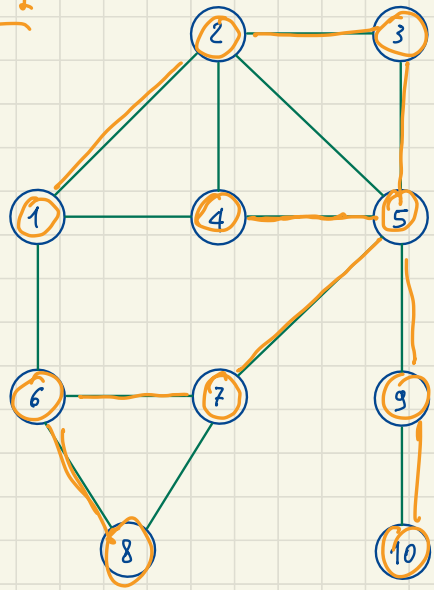


da F



ESEMPIO

da 1



ALGORITMO visitaInProfondità (grafo $G=(V,E)$, vertice s) $\rightarrow$ Albero

$T \leftarrow$ albero formato solo da s

visitaRicorsiva (G, s, T)

RETURN T

PROCEDURA visitaRicorsiva (grafo $G=(V,E)$, vertice u, albero T)

marca u come visitato

FOR EACH $(u,v) \in E$ DO

IF v non è marcato come visitato THEN

aggiungi v e (u,v) a T

visitaRicorsiva (G, v, T)

Problemi di ottimizzazione e algoritmi *greedy*

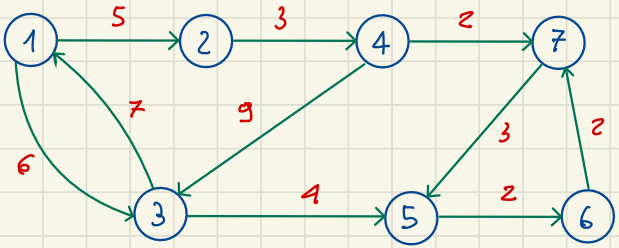
GRAFI PESATI

Spesso ad archi o nodi di un grafo si associano delle informazioni

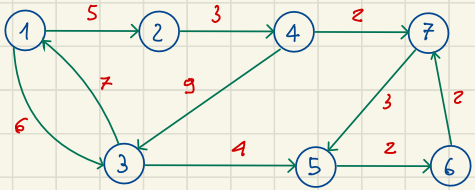
GRAFO con pesi sugli archi

$G = (V, E)$ grafo

$w : E \to \mathbb{R}$ funzione peso



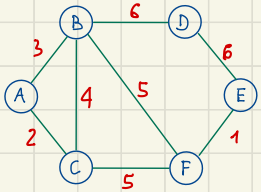
GRAFI PESATI: alcuni problemi



CAMMINI MINIMI

COMMESSO VIAGGIATORE

ALBERO RICOPRENTE MINIMO (grafi non orientati)

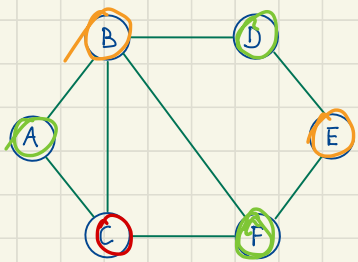


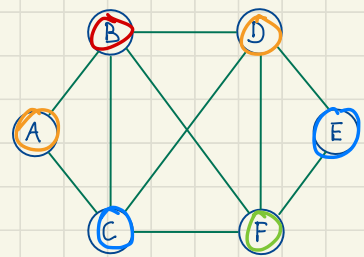
PROBLEMI DI OTTIMIZZAZIONE: esempi

COLORAZIONE DI GRAFI

Istanza Grafo non orientato $G=(V,E)$

Problema Trovare il minimo numero di colori da attribuire ai vertici di G in modo che vertici adiacenti abbiano colori differenti





PROBLEMI DI OTTIMIZZAZIONE: esempi

COLORAZIONE DI GRAFI

Istanza Grafo non orientato $G=(V,E)$

Problema Trovare il minimo numero di colori da attribuire ai vertici di G in modo che vertici adiacenti abbiano colori differenti

coloramento: $c: V \to Colori$ t.c. $\underbrace{(x,y) \in E \Rightarrow c(x) \neq c(y)}_{\text{vincolo}}$

obiettivo: minimizzare $\# Im(c)$

Soluzione ammissibile vertici adiacenti hanno colori differenti

Soluzione ottima soluzione ammissibile con minimo numero di colori

PROBLEMI DI OTTIMIZZAZIONE:

Tra tutte le soluzioni AMMISSIBILI → soddisfano i vincoli posti dal problema

vogliamo determinarne una OTTIMA

rispetto a un dato criterio → di solito massimizzazione o minimizzazione

PROBLEMI DI OTTIMIZZAZIONE: esempio

ZAINO MONODIMENSIONALE

Istanza Zaino di altezza h, K contenitori $c_1, c_2, \dots, c_K$ di altezze $h_1, h_2, \dots, h_K$

Problema Scegliere quali contenitori collocare nello zaino, in modo da riempirlo il più possibile

Soluzione ammissibile sottoinsieme $S \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$$\underbrace{f(S) \leq h}_{\text{vincolo}} \qquad \text{con} \qquad f(S) = \sum_{c_i \in S} h_i$$

Soluzione ottima sottoinsieme ammissibile $S^* \subseteq \{c_1, \dots, c_K\}$ t.c.

$$\underbrace{f(S^*) \geq f(S) \text{ per ogni } S \text{ ammissibile}}_{\text{ottimalità}}$$

PROBLEMI DI OTTIMIZZAZIONE: esempio

ZAINO MONODIMENSIONALE

Istanza Zaino di altezza h, K contenitori $c_1, c_2, \dots, c_K$ di altezze $h_1, h_2, \dots, h_K$

Problema Scegliere quali contenitori collocare nello zaino, in modo da riempirlo il più possibile

<u>Soluzione ammissibile</u> sottoinsieme $S \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$f(S) \leq h$ con $f(S) = \sum_{c_i \in S} h_i$

<u>Soluzione ottima</u> sottoinsieme ammissibile $S^* \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$f(S^*) \geq f(S)$ per ogni S ammissibile

es.

$h = 6$ $K = 4$

$h_1 = 1.5$ $h_2 = 2.5$ $h_3 = 3$ $h_4 = 5$

$S_1 = \{c_1, c_4\}$ NON AMM

$S_2 = \{c_4\}$ AMM $f(S_2) = 5$

$S_3 = \{c_1, c_3\}$ AMM $f(S_3) = 4.5$

$S_4 = \{c_2, c_3\}$ AMM $f(S_4) = 5.5$

TECNICA GREEDY

PROBLEMA DI OTTIMIZZAZIONE

- C INSIEME DI CANDIDATI
- TROVARE $S^* \subseteq C$ OTTIMA

- A partire dall'insieme vuoto, in una sequenza di passi si costruisce una soluzione ammissibile $S \subseteq C$
- Ad ogni passo si espande una soluzione parziale già ottenuta
- L'algoritmo termina quando non è più possibile espandere la soluzione parziale

TECNICA GREEDY

PROBLEMA DI OTTIMIZZAZIONE

- C INSIEME DI CANDIDATI
- TROVARE $S^* \subseteq C$ OTTIMA

- A partire dall'insieme vuoto, in una sequenza di passi si costruisce una soluzione ammissibile $S \subseteq C$
- Ad ogni passo si espande una soluzione parziale già ottenuta
- L'algoritmo termina quando non è più possibile espandere la soluzione parziale

ESPANSIONE

- SOLUZIONE AMMISSIBILE

  La soluzione parziale soddisfa i vincoli del problema

- SCELTA DELL'OTTIMO LOCALE

  Tra i candidati disponibili si sceglie quello che, al momento, appare migliore

- SCELTA IRREVOCABILE

  Le scelte effettuate non vengono più messe in discussione

TECNICA GREEDY: schema

ALGORITMO greedy (insieme C) $\rightarrow$ soluzione

$S \leftarrow \emptyset$

WHILE $C \neq \emptyset$ DO

$x \leftarrow$ seleziona (C) $\rightarrow$ sceglie candidato "migliore" al momento

$C \leftarrow C \setminus \{x\}$

IF $S \cup \{x\}$ è ammissibile

THEN $S \leftarrow S \cup \{x\}$

RETURN S

TECNICA GREEDY: esempio

ZAINO MONODIMENSIONALE

Istanza Zaino di altezza h, K contenitori $c_1, c_2, \dots, c_K$ di altezze $h_1, h_2, \dots, h_K$

Problema Scegliere quali contenitori collocare nello zaino, in modo da riempirlo il più possibile

$C = \{c_1, c_2, \dots, c_K\}$

CANDIDATI = CONTENITORI

<u>Soluzione ammissibile</u> sottoinsieme $S \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$f(S) \leq h$ con $f(S) = \sum_{c_i \in S} h_i$

<u>Soluzione ottima</u> sottoinsieme ammissibile $S^* \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$f(S^*) \geq f(S)$ per ogni S ammissibile

STRATEGIA GREEDY:

ispezioniamo i contenitori in ordine di altezza, dal più alto al più basso

"seleziona": scegli il contenitore più alto

ZAINO MONODIMENSIONALE

Istanza Zaino di altezza h, K contenitori $c_1, c_2, \dots, c_K$ di altezze $h_1, h_2, \dots, h_K$

Problema Scegliere quali contenitori collocare nello zaino in modo da riempirlo il più possibile

$C = \{c_1, c_2, \dots, c_K\}$

CANDIDATI = CONTENITORI

Soluzione ammissibile sottoinsieme $S \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$f(S) \leq h$ con $f(S) = \sum_{c_i \in S} h_i$

Soluzione ottima sottoinsieme ammissibile $S^* \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$f(S^*) \geq f(S)$ per ogni S ammissibile

STRATEGIA GREEDY:

ispezioniamo i contenitori in ordine di altezza, dal più alto al più basso

FS 1 $h = 20$

- c_1 $h_1 = 8$
- c_2 $h_2 = 7$
- c_3 $h_3 = 6$
- c_4 $h_4 = 3$
- c_5 $h_5 = 2$
- c_6 $h_6 = 1$

TOT 20

OTTIMA

FS 2 $h = 20$

- c_1 $h_1 = 8$
- c_2 $h_2 = 7$
- c_3 $h_3 = 6$
- c_4 $h_4 = 3$
- c_5 $h_5 = 3$
- c_6 $h_6 = 1$

TOT 19

$\{c_1, c_3, c_4, c_5\} \rightsquigarrow 20$ ottima

FS 3 $h = 20$

- c_1 $h_1 = 10$
- c_2 $h_2 = 8$
- c_3 $h_3 = 8$
- c_4 $h_4 = 3$
- c_5 $h_5 = 3$

TOT 18

$\{c_2, c_3, c_4\}$

$\downarrow$

19

ottima

Esempio: ZAINO CON VALORI (o Sacco del ladro)

Istanza Sacco che può portare al massimo peso P

K oggetti: $c_1, c_2, \dots, c_K$ di peso $p_1, p_2, \dots, p_K$ e valore $v_1, v_2, \dots, v_K$, rispettivamente

Problema Scegliere quali oggetti collocare nel sacco, in modo da massimizzare il valore totale, evitando che il sacco si rompa

<u>Obiettivo</u> guadagnare il più possibile

<u>Vincolo</u> non superare la capacità del sacco

Esempio: ZAINO CON VALORI (o Sacco del ladro)

Istanza Sacco che può portare al massimo peso P

K oggetti $c_1, c_2, \dots, c_K$ di peso $p_1, p_2, \dots, p_K$ e valore $v_1, v_2, \dots, v_K$, rispettivamente

$C = \{c_1, \dots, c_K\}$ insieme candidato

Problema Scegliere quali oggetti collocare nel sacco, in modo da massimizzare il valore totale, evitando che il sacco si rompa

$S \subseteq C$ soluzione $\quad val(S) = \sum_{c_i \in S} v_i$ valore di S

Soluzione ammissibile $S \subseteq C$ è ammissibile se $\sum_{c_i \in S} p_i \leq P$

Soluzione ottima $S^* \subseteq C$ è ottima se (1) S^* è ammissibile

(2) $\forall S$ ammissibile $val(S^*) \geq val(S)$

ZAINO CON VALORI: strategia greedy

ispeziona gli oggetti in ordine di valore

"seleziona": scegli l'oggetto di maggior valore

Es $p = 20$

	p	v
c_1	12	100
c_2	3	80
c_3	6	75
c_4	6	70
c_5	3	30

Valore totale
210

c_2, c_3, c_4, c_5 255

ottimo

Esempio: DISTRIBUTORE AUTOMATICO DI RESTO

Istanza Quantità di monete disponibili per ciascun taglio

Resto da erogare

K_5 = # monete da 5 centesimi

K_{10} = # monete da 10 centesimi

K_{20} = # monete da 20 centesimi

K_{50} = # monete da 50 centesimi

R = resto

$K_5, K_{10}, K_{20}, K_{50} \geq 0 \quad R > 0$

Problema E' possibile dare il resto esatto?

$\exists\, j_5, j_{10}, j_{20}, j_{50} \geq 0$ t.c.

$$j_5 \leq K_5, \quad j_{10} \leq K_{10}, \quad j_{20} \leq K_{20}, \quad j_{50} \leq K_{50}$$

e $5 j_5 + 10 j_{10} + 20 j_{20} + 50 j_{50} = R$?

Esempio: DISTRIBUTORE AUTOMATICO DI RESTO

Istanza Quantità di monete disponibili per ciascun taglio

Resto da erogare

Problema È possibile dare il resto esatto?

K_5 = # monete da 5 centesimi

K_{10} = # monete da 10 centesimi

K_{20} = # monete da 20 centesimi

K_{50} = # monete da 50 centesimi

R = resto

$K_5, K_{10}, K_{20}, K_{50} \geq 0 \quad R > 0$

$\exists\, j_5, j_{10}, j_{20}, j_{50} \geq 0$ t.c.

$j_5 \leq K_5, \; j_{10} \leq K_{10}, \; j_{20} \leq K_{20}, \; j_{50} \leq K_{50}$

e $5 j_5 + 10 j_{10} + 20 j_{20} + 50 j_{50} = R$?

Esempio

MONETA	QUANTITÀ
5	10
10	10
20	10
50	10

Resto = 85

$K_5 = K_{10} = K_{20} = K_{50} = 10$

$R = 85$

1 1 1 1

3 3 2 0

Soluzione parziale ammissibile

vettore di interi $(j_5, j_{10}, j_{20}, j_{50})$ t.c.

$j_5, j_{10}, j_{20}, j_{50} \geq 0,$

$j_5 \leq K_5, \; j_{10} \leq K_{10}, \; j_{20} \leq K_{20}, \; j_{50} \leq K_{50}$ e

$5 j_5 + 10 j_{10} + 20 j_{20} + 50 j_{50} \leq R$

DISTRIBUTORE AUTOMATICO DI RESTO: strategia greedy

"selezione": scegli la moneta disponibile di maggior valore

Esempio 1

MONETA	QUANTITÀ	
50	10	1
20	10	1
10	0	0
5	5	3

resto 85

Resto = 85 SI

Esempio 2

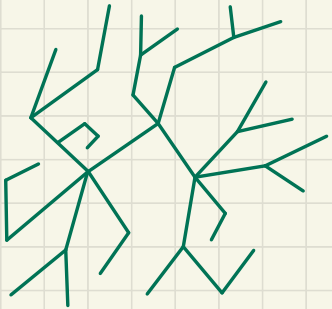
MONETA	QUANTITÀ		
50	10	1	0
20	10	1	4
10	0	0	0
5	2	2	1

resto 80

Resto = 85 NO

Albero ricoprente minimo

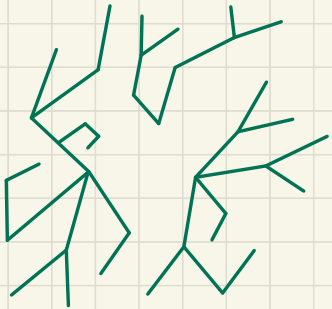
ALBERI



ALBERO. grafo NON ORIENTATO, CONNESSO e PRIVO DI CICLI

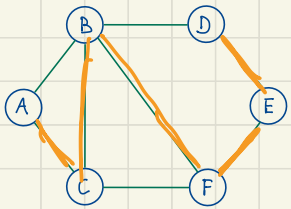
<u>Proprietà</u> $G=(V,E)$ non orientato e connesso è un albero sse $\#E = \#V - 1$

FORESTA: insieme di alberi



ALBERO di SUPPORTO o RICOPRENTE (Spanning tree)

Dato $G=(V,E)$ grafo non orientato connesso, un albero ricoprente di G è un albero $G'=(V',E')$ con $V'=V$ e $E' \subseteq E$



ALBERO RICOPRENTE MINIMO

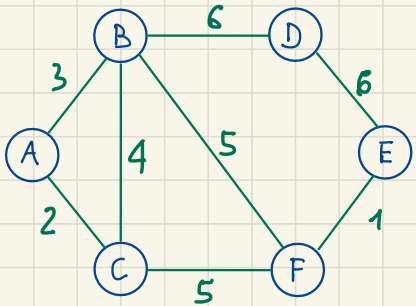
<u>Problema</u> Dato $G=(V,E)$ non orientato connesso con una funzione peso $\omega: E \to \mathbb{R}$ trovare un albero ricoprente $T=(V,E_T)$ di peso minimo

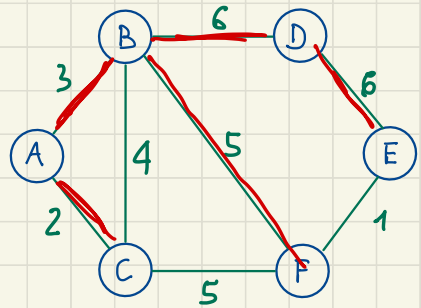
- Dato $G'=(V,E')$ $E' \subseteq E$: $\omega(G') = \sum_{e \in E'} \omega(e)$

- $T=(V,E_T)$ è una soluzione del problema se:
  - T è un albero con $E_T \subseteq E$ ← T ammissibile
  - $\forall T'$ albero con $T'=(V,E_{T'})$ $E_{T'} \subseteq E$: $\omega(T) \leq \omega(T')$

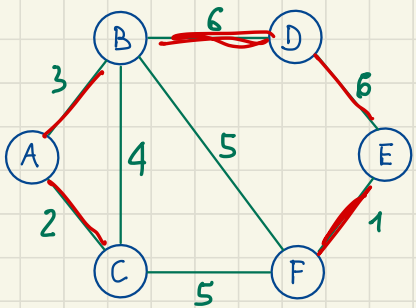
T è ottima

ESEMPIO

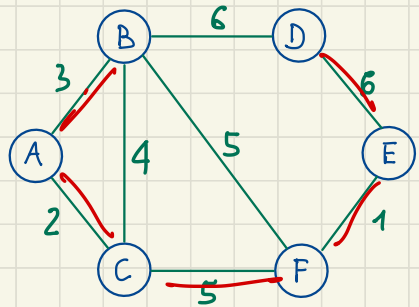




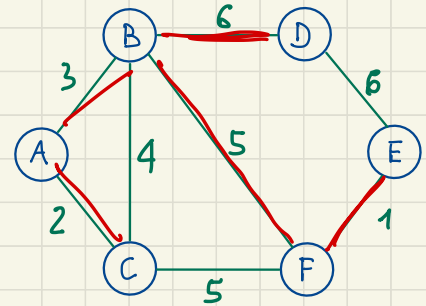
$\omega(S) = 22$



$\omega(T) = 18$



17



17

$\binom{m}{n-1}$

ALBERO RICOPRENTE MINIMO: una strategia greedy

Inizialmente:

$T = (V, \emptyset)$

$C = E$ insieme dei candidati

Ad ogni passo:

prelevo da C l'arco di peso minimo

lo aggiungo a T se non forma cicli

con gli archi già scelti

ALGORITMO greedy (insieme C) $\rightarrow$ soluzione

$S \leftarrow \emptyset$

WHILE $C \neq \emptyset$ DO

$x \leftarrow$ seleziona (C)

$C \leftarrow C - \{x\}$

IF $S \cup \{x\}$ è ammissibile THEN

$S \leftarrow S \cup \{x\}$

RETURN S

ALGORITMO Kruskal (Grafo $G=(V,E,w)$) $\rightarrow$ albero

ordina E in maniera non decrescente in base ai pesi

$T \leftarrow (V, \emptyset)$

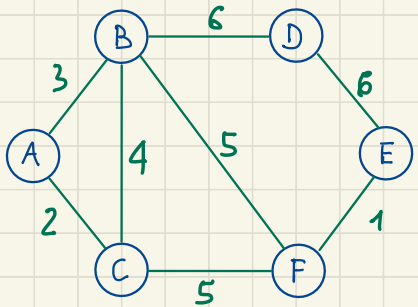
FOR EACH $(x,y) \in E$ secondo l'ordine DO

IF x e y non sono connessi in T THEN

aggiungi a T l'arco (x,y)

RETURN T

ESEMPIO



(E, F) —

(A, C) —

(A, B) —

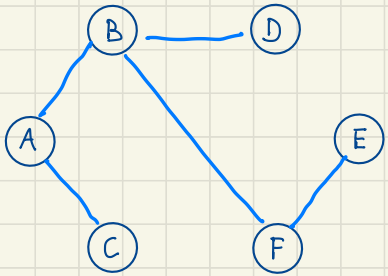
(B, C)

(B, F) —

(C, F)

(B, D) —

(D, E)



17

ALGORITMO Kruskal (Grafo $G = (V, E, w)$) → albero

ordina E in maniera non decrescente in base ai pesi

$T \leftarrow (V, \emptyset)$

FOR EACH $(x, y) \in E$ secondo l'ordine DO

  IF x e y non sono connessi in T THEN

    aggiungi a T l'arco (x, y)

RETURN T