1 Tracce esercizi

Es.1 Es. 22 p. 200 Kleinberg Tardos
Let us say that a graph $G = (V, E)$ is a *near-tree* if it is connected and has at most $n + 8$ edges, where $n = |V|$. Give an algorithm with running time $O(n)$ that takes a near-tree G with costs on its edges, and returns a minimum spanning tree of G. You may assume that all the edge costs are distinct.
Domanda aggiuntiva: se un grafo *near-tree* fosse stato definito come un grafo connesso con un numero di archi pari al massimo a $n + costante$, che cosa sarebbe cambiato nella soluzione?

Es.2 Es. 22 p. 200 Kleinberg Tardos
Consider the Minimum Spanning Tree Problem on an undirected graph $G = (V, E)$, with a cost $c_e \geq 0$ on each edge, where the costs may not all be different. If the costs are not all distinct, there can in general be many distinct minimum-cost solutions. Suppose we are given a spanning tree $T \subseteq E$ with the guarantee that for every $e \in T$, e belongs to some minimum-cost spanning tree in G. Can we conclude that T itself must be a minimum-cost spanning tree in G? Give a proof or a counterexample with explanation.

Es.3 Es. 12 p. 193 capitolo 4 Kleinberg - Tardos
"Suppose you have n video streams that need to be sent,one after another, over a communication link. Stream i consists of a total of b_i bits that need to be sent, at a constant rate, over a period of t_i seconds. You cannot send two streams at the same time, so you need to determine a schedule for the streams: an order in which to send them. Whichever order you choose, there cannot be any delays between the end of one stream and the start of the next. Suppose your schedule starts at time 0 (and therefore ends at time $\sum_{i=1}^{n} t_i$, whichever order you choose). We assume that all the values b_i and t_i are positive integers.
Now, because you're just one user, the link does not want you taking up too much bandwidth, so it imposes the following constraint, using a fixed parameter r:
$(*)$ For each natural number $t > 0$, the total number of bits you send over the time interval from 0 to t cannot exceed rt. Note that this constraint is only imposed for time intervals that start at 0, not for time intervals that start at any other value.
We say that a schedule is valid if it satisfies the constraint $(*)$ imposed by the link.
The Problem. Given a set of n streams, each specified by its number of bits b_i and its time duration t_i, as well as the link parameter r, determine whether there exists a valid schedule.
Example. Suppose we have n = 3 streams, with (b1, t1) = (2000, 1), (b2, t2) = (6000, 2), (b3, t3) = (2000, 1), and suppose the link's parameter is r = 5000. Then the schedule that runs the streams in the order 1, 2, 3, is valid, since the constraint $(*)$ is satisfied: t = 1: the whole first stream has been sent, and $2000 < 5000 \cdot 1$
$t = 2$: half of the second stream has also been sent, and $2000 + 3000 < 5000 \cdot 2$
Similar calculations hold for $t = 3$ and $t = 4$.

(a) Consider the following claim:
There exists a valid schedule if and only if each stream i satisfies $b_i \leq rt_i$
Decide whether you think the claim is true or false, and give a proof of either the claim or its negation.

(b) Give an algorithm that takes a set of n streams, each specified by its number of bits b_i and its time duration t_i, as well as the link parameter r, and determines whether there exists a valid schedule. The running time of your algorithm should be polynomial in n."

2 Soluzioni

Es.1 L'algoritmo consiste in una BFS modificata. Ogni volta che si incontra un nodo già visitato si incontra un ciclo, dato dal cammino sull'albero della BFS tra il nodo che sto visitando e quello già visitato e l'arco tra loro due. Si prende l'arco di peso massimo nel ciclo e, se presente nell'albero della BFS, lo si elimina inserendo l'arco del ciclo che prima non era presente. Si restituisce l'albero T della BFS così modificata.
Il grafo restituito è uno spanning tree in quanto è un albero e nessun nodo viene lasciato scoperto (copre infatti tutti i nodi dell'albero della BFS che a sua volta è uno spanning tree). Inoltre è minimo; per dimostrarlo supponiamo per assurdo non lo sia, ossia $\exists T' \neq T$ t.c. T' è uno spanning tree con un costo minore di T. Essendo diversi differiscono di almeno un arco $e' \in E(T') \wedge e' \notin E(T)$. Osserviamo però che gli archi esclusi dal nostro algoritmo sono solo i più pesanti dei cicli incontrati e quindi, per la *Cycle Property*, non possono far parte di un minimum spanning tree, assurdo. Il nostro algoritmo restituisce l'mst.
Il costo è dato dalla BFS, che è $O(m+n)$ e dalla verifica dei cicli, verificare un ciclo costa al più $O(n)$ in quanto un ciclo si può pensare come una permutazione di un sottoinsieme di nodi, il numero di cicli è costante essendo il grafo *near-tree*. Dato che m è $O(n)$, l'algoritmo richiede tempo di esecuzione $O(n)$.

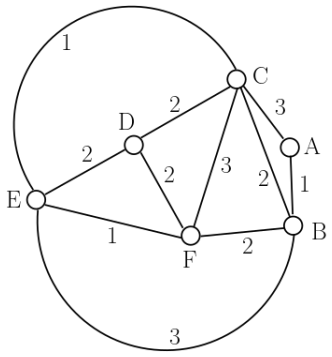

Figure 1:

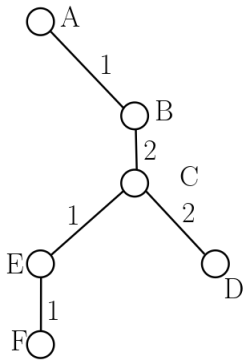

Figure 2:

Es.2 Controesempio.
Il grafo in figura 1 ha (almeno) gli mst delle figure 2, 3 e 4.

L'albero della figura 5 gode della proprietà che ogni suo arco appartiene a qualche mst. Osserviamo però che non ha peso 7 e quindi non è un mst.

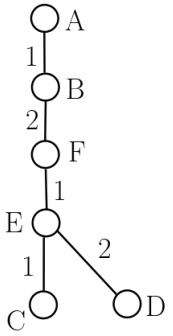

Figure 3:

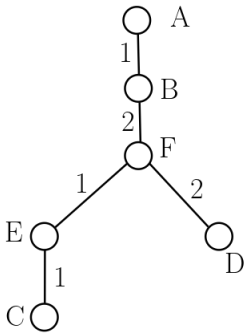

Figure 4:

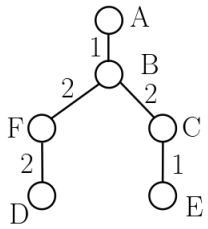

Figure 5:

Es.3

Oss. (Banale).
Dire che uno schedule è valido se e solo se $\forall t \in \mathbb{N} - \{0\}$ il numero totale di bit trasmessi nell'intervallo che va da 0 a t è $\leq rt$ equivale a dire che uno schedule è valido se e solo se $\forall t \in \mathbb{N} - \{0\}$ t.c. $t \leq \sum_{i=1}^{n} t_i$ il numero totale di bit trasmessi nell'intervallo che va da 0 a t è $\leq rt$, in quanto una volta trasmessi tutti i video non ci saranno ulteriori bit spediti.

a.
Proviamo a dimostrare il claim per vedere se è giusto o sbagliato. Dato che si tratta di un se e solo se servono due dimostrazioni, una per ogni verso della freccia.

Claim. $\exists$ *uno schedule valido* $\leftrightarrow \forall$ *stream* i $b_i \leq rt_i$

Proof.
$\leftarrow$
Una qualsiasi permutazione dei video è uno schedule valido, quindi qualsiasi algoritmo è corretto.
Dimostriamolo per induzione sul tempo:

- $t = 1$ Il numero di bit spediti al tempo 1, che chiamo B_1, deve essere minore uguale di $rt = r$. Quanto vale B_1? I bit del primo video da mandare diviso il tempo che serve per spedire il video e *per ogni* video i si ha che $b_i \leq rt_i$, ossia $\frac{b_i}{t_i} \leq r$
- t generico. (Quindi per ipotesi induttiva stiamo assumendo che la proprietà valga per ogni $t' \in \{1, ..., t-1\}$). Il numero totale di bit spediti, che chiamo B, è uguale alla somma dei bit mandati fino al tempo $t-1$, che chiamo B_{t-1}, e quelli spediti solo al tempo t, che chiamo B_t. Per ipotesi induttiva sappiamo che $B_{t-1} \leq r(t-1)$. Se dimostriamo che $B_t \leq r$ abbiamo fatto perché, se così fosse, $B = B_{t-1} + B_t \leq r(t-1) + r = rt$. Quanto vale B_t? Quanto il numero di bit del video che si sta mandando diviso il tempo che serve per mandare quel video. Osserviamo che *per ogni* video i si ha che $b_i \leq rt_i$, ossia $\frac{b_i}{t_i} \leq r$

Osserviamo che nell'induzione non abbiamo usato video specifici ma abbiamo lavorato con *qualsiasi* video, quindi qualsiasi permutazione dei video è uno schedule valido e quindi qualsiasi algoritmo è corretto. Ovviamente dato che ogni schedule è valido, ne esiste almeno uno valido.
$\rightarrow$
Abbiamo un A $\rightarrow$ B e questo equivale a un $\neg B \rightarrow \neg A$, ossia $\exists$ *stream* i $b_i \leq rt_i \rightarrow$ *ogni schedule non è valido.* Proviamo a dimostrare per assurdo quest'ultimo statement. Per farlo dobbiamo dimostrare che $\neg B \rightarrow A$ è falso. $\neg B \rightarrow A$ corrisponde a dire
(#): $\exists$ *stream* i $b_i \leq rt_i \rightarrow \exists$ *schedule valido.*

Def. i è un video *pesante* se $b_i > r \cdot t_i$, altrimenti ($b_i \leq r \cdot t_i$) è *leggero*

Assumendo ci sia un video pesante dobbiamo provare a far vedere che non può esserci nessuno schedule valido.
Sia i il video pesante che è spedito per primo tra tutti i video pesanti.
Supponiamo sia il primo video spedito in assoluto e consideriamo l'intervallo $[0, t_i]$: il numero di bit spediti sarebbe proprio b_i, che sono $b_i > rt_i$, e quindi questo non può accadere perché lo schedule è valido.
Supponiamo non sia spedito per primo, ma anzi a un tempo t^*, dato che lo schedule è valido allora $b^* \leq rt^*$, dove b^* è il numero di bit spediti fino al tempo t^*. Come possiamo far vedere che lo schedule non è valido? Pensandoci su ci accorgiamo che lo statement (#) non è sbagliato, si consideri infatti il seguente controesempio:
$r = 3000$
$v_1 =$ (9000 bit,tempo 3)
$v_2 =$ (9000 bit,tempo 3)
$v_3 =$ (2000 bit,tempo 1)
$v_4 =$ (10000 bit,tempo 3)
v_4 sarebbe il video pesante, se mandiamo prima v_1, v_2, v_3 e poi v_4 otteniamo uno schedule valido. Infatti, dato che i primi tre video sono leggeri e per la dimostrazione della freccia da destra a sinistra, possiamo concludere che fino a $t_1+t_2+t_3 = 7$ non vengono spediti troppi bit. In particolare al tempo 7 saranno spediti 20000 bit $\leq r \cdot 7 = 3000 \cdot 7 = 21000$. Gli intervalli $[0, 8]$, $[0, 9]$, $[0, 10]$ sono tutti validi (rispettivamente: $3333, \overline{3} + 20000 \leq 8 \cdot 3000, 6666, \overline{6} + 20000 \leq 9 \cdot 3000, 10000 + 20000 \leq 10 \cdot 3000$) Quindi concludiamo che la parte di claim corretta è solamente

Claim. $\exists$ *uno schedule valido* $\leftarrow \forall$ *stream* i $b_i \leq rt_i$

□

Oss. dalla dimostrazione

- Se $\forall$ stream i $b_i \leq rt_i$ allora *qualsiasi* algoritmo restituisce uno schedule corretto
- $\exists$ *uno schedule valido* $\rightarrow \exists$ *stream* i $b_i \leq rt_i$
  Supponiamo per assurdo che esista uno schedule valido e tutti i video siano pesanti ($\forall$ stream i $b_i > rt_i$). Allora il primo video spedito sarà uno pesante e questo non è possibile per quanto detto nella dimostrazione

b.
Cosa ha impedito la dimostrazione del se e solo se nel claim di prima? Se si pensa bene al controesempio è stato possibile spedire il video pesante grazie al fatto che mandando i video leggeri prima abbiamo guadagnato un certo vantaggio. Per spiegare meglio: se per esempio

bisogna spedire
$v_1 = (1000\text{bit}, \text{tempo } 1)$
$v_2 = (7000\text{bit}, \text{tempo } 1)$
$r = 4000$
Mandando prima v_1 ho un guadagno di $4000 - 1000 = 3000$ bit che mi fa da cuscinetto per il video successivo. Una volta spedito il secondo infatti, guardando l'intervallo $[0, 2]$, ho che tot bit $= 7000 + 1000 = 8000 = 4000 + \textit{3000} + 1000 \leq r \cdot t = 2 \cdot 4000$. Se avessi spedito prima il secondo non avrei ottenuto uno schedule valido.
Questa osservazione ci suggerisce il seguente algoritmo greedy:

1. Ordina i video in modo crescente rispetto $\frac{b_i}{t_i}$
2. Verifica se spedendoli dal primo all'ultimo si ottiene uno schedule valido, se sì restituisci quello schedule altrimenti dire che non esiste nessuno schedule valido

Il costo di esecuzione è quello dell'ordinamento, ossia $O(n \log n)$, più la verifica dello schedule. Quanto costa la verifica?

Oss. Sia i un video generico con (b_i, t_i) e j il tempo generico in cui si inizia a spedirlo. Sia B_{j-1} il numero di bit spediti fino al tempo $j-1$. Supponiamo fino a quel momento lo schedule è valido ossia $B_{j-1} \leq r \cdot (j-1)$), quindi $\Delta = B_{j-1} - \ r \cdot (j-1)) \leq 0$ (*).
Vogliamo dimostrare che se $B_{j-1} + b_i \leq r \cdot (j - 1 + t_i)$ allora non c'è nessun tempo tra j e $t_i + j$ in cui vengono spediti troppi bit.
Osserviamo che $B_{j-1} + b_i \leq r \cdot (j-1+t_i) \ \leftrightarrow \ B_{j-1} - r \cdot (j-1) + b_i \leq r \cdot t_i \ \leftrightarrow \ \Delta \leq r \cdot t_i - b_i$, mi riferisco all'ultima disequazione con (**).
Se per assurdo esistesse un $k \in \{1, ..., t_i\}$ t.c. $B_{j-1} + \frac{b_i}{t_i} \cdot k > r(j-1+k)$, allora $B_{j-1} - r \cdot (j-1) > r \cdot k - \frac{b_i}{t_i} \cdot k$, quindi $\Delta > r \cdot k - \frac{b_i}{t_i} \cdot k$. Moltiplicando a sinistra e destra per $\frac{t_i}{k}$ si ottiene: $\Delta \cdot \frac{t_i}{k} > r \cdot t_i - b_i$; dato che $k \in \{1, ..., t_i\}$ sappiamo che $1 \leq \frac{t_i}{k} \leq t_i$ e, da (*), sappiamo che $\Delta \leq 0$, quindi $\Delta \geq \Delta \cdot \frac{t_i}{k} > r \cdot t_i - b_i$,questo contraddice (**) e quindi ecco l'assurdo.

Quindi per la verifica basterà vedere se per ogni video h accade che $\sum_{i=1}^{h} b_i \leq r \cdot \sum_{i=1}^{h} t_i$ e questo costa $O(n)$.
Ora bisogna dimostrare la correttezza dell'algoritmo, si userà sia un po' della tecnica *Greedy Stays Ahead*, sia di *Exchange Argument*.

Claim. $\exists$ *schedule valido* $\leftrightarrow$ *lo schedule ottenuto dall'algoritmo greedy è valido*

Proof.
$\leftarrow$
Non c'è bisogno di dimostrarlo.
$\rightarrow$
Supponiamo per assurdo che il nostro algoritmo greedy dica che nessuno schedule è valido ma esista un algoritmo F che restituisce uno schedule corretto. Con F indicheremo sia

l'algoritmo sia lo schedule valido che restituisce l'algoritmo, sarà chiaro dal contesto.
Parte Exchange Argument
Dalle osservazioni fatte nella prima parte dell'esercizio ()punto a) deduciamo che se esiste uno schedule valido F allora ci deve essere almeno un video i t.c. $b_i \leq rt_i$.
Sia v il nome dell'istanza, senza perdita di generalità sia $\{1, ..., k-1\}$ l'insieme dei video leggeri. Sia v^* l'istanza formata prendendo v e togliendo i video leggeri. Se applichiamo l'algoritmo F a v^*, verrà restituito una permutazione dei video formata dalla permutazione restituita da F su v tranne quelli leggeri.
Es. siano 1, 2, 3 i video leggeri e 4, 5, 6 quelli pesanti. Supponiamo F su tutta l'istanza restituisca 3, 4, 2, 1, 6, 5; allora F sull'istanza v^* restituisce 4,6,5.

Claim. *Se l'algoritmo F restituisce uno schedule valido sull'istanza v (per abbreviare F(v)), allora F(v*) è t.c.*
$\forall t \in \mathbb{N} - \{0\}$ *t.c.* $t \leq \sum_{i \in v^*} t_i$ *il numero di bit spediti in [0,t] è minore uguale di* $r \cdot t + r \cdot \sum_{i \in \{1,...,k-1\}} t_i - \sum_{i \in \{1,...,k-1\}} b_i$

Proof. Indicheremo con $B_t i$ il numero di bit spediti nell'intervallo $[0, ti]$ secondo lo schedule di $F(v^*)$.
Supponiamo per assurdo non sia vero il claim, ossia $F(V)$ valido e, per quanto riguarda $F(V^*)$, $\exists\, t$ t.c. $B_t > r \cdot t + r \cdot \sum_{i \in \{1,...,k-1\}} t_i - \sum_{i \in \{1,...,k-1\}} b_i$; sia y il video che si sta spedendo a quel tempo.
Sia t' il tempo in cui si sta spedendo y nello schedule $F(v)$, quindi il tempo t' in $F(V)$ corrisponde al tempo t dello schedule $F(v^*)$.
Se $B_t > rt + r \sum_{i \in \{1,...,k-1\}} t_i - \sum_{i \in \{1,...,k-1\}} b_i$ allora $B_t + \sum_{i \in \{1,...,k-1\}} b_i > rt + r \sum_{i \in \{1,...,k-1\}} t_i$.
Sia $H \subseteq \{1, ..., k-1\}$ il sottoinsieme dei video leggeri che sono stati inviati dallo schedule $F(V)$ al tempo t'. Allora $B_t + \sum_{i \in \{1,...,k-1\}} b_i > rt + r \sum_{i \in \{1,...,k-1\}} t_i$ equivale a $B_t + \sum_{i \in H} b_i + \sum_{i \in H - \{1,...,k-1\}} b_i > rt + r \sum_{i \in H - \{1,...,k-1\}} t_i + r \sum_{i \in H} t_i$, dato che i video sono leggeri possiamo dedurre che $B_t + \sum_{i \in H} b_i > rt + r \sum_{i \in H} t_i$. Ma $B_t + \sum_{i \in H} b_i$ è proprio il numero di bit spediti dallo schedule $F(v)$ al tempo t' e $rt + r \sum_{i \in H} t_i = rt'$, quindi lo schedule non sarebbe valido, assurdo. □

Per abbreviare useremo questa definizione

Def. Se uno schedule F' sull'istanza v^* è tale che $\forall t \in \mathbb{N} - \{0\}$ t.c. $t \leq \sum_{i \in v^*} t_i$ il numero di bit spediti in [0,t] è minore uguale di $r \cdot t + r \cdot \sum_{i \in \{1,...,k-1\}} t_i - \sum_{i \in \{1,...,k-1\}} b_i$, allora diciamo che F' è *valido in modo rilassato.*

Usiamo anche quest'altra definizione

Def. Dato un algoritmo A che restituisce uno schedule e un tempo i, sia B_i il numero di bit spediti nell'intervallo di tempo $[0, i]$. Definiamo $X_A(i) = r \cdot i - B_i$

Quando parleremo di un algoritmo generico scriveremo $X(i)$ anziché $X_A(i)$.

Parte Greedy Stays Ahead.

Claim. $\forall$ *algoritmo A che restituisce uno schedule,* $\forall$ *tempo i si ha che* $X_A(i) \leq X_G(i)$*, dove G è il nostro algoritmo greedy.*

Proof. Dimostriamolo per induzione sul tempo.

- Caso base: t = 1. L'algoritmo greedy spedisce il video che ha $\frac{b_i}{t_i}$ minimo, quindi massimizza $r - \frac{b_i}{t_i} = r \cdot 1 - B_1 = X(i)$
- Passo induttivo: t generico. Abbiamo $X(t) = r \cdot (t-1+1) - B_{t-1} - \frac{b_j}{t_j} \cdot k$. Dove B_{t-1} è il numero di bit spediti alla fine del tempo $t-1$, j è un generico video che si sta spedendo al tempo t e k è da quanto tempo si sta spedendo il video j. Abbiamo che $r \cdot (t-1+1) - B_{t-1} - \frac{b_j}{t_j} \cdot k = \underbrace{r \cdot (t-1) - B_{t-1}}_{\alpha} + \underbrace{r - \frac{b_j}{t_j} \cdot k}_{\beta}$.

  La parte α l'algoritmo greedy la massimizza per ipotesi induttiva, per quanto riguarda la parte β: r è fissato quindi è massimizzata quando $\frac{b_j}{t_j}$ è minimizzata e, proprio per la nostra scelta greedy, G la minimizza in qualsiasi tempo.

□

Unione delle due parti

Claim. *Sia* $F(v^*)$ *uno schedule valido in senso rilassato allora* $G(v)$ *restituisce uno schedule valido.*

Proof. Dimostriamolo per assurdo.
Se il greedy afferma che nessuno schedule è valido deve esistere un video j t.c. $b_j + \sum_{i=1}^{j-1} b_i > r \cdot \sum_{i=1}^{j} t_i$.
Dato che quando si mandano i video leggeri qualsiasi ordine è uno schedule valido, il video j che trae in inganno l'algoritmo greedy deve essere un video pesante. Facciamo girare l'algoritmo greedy solo sull'istanza v^*. Sia h un tempo in cui si sta inviando j e $B_{PG}(h)$ il numero di bit dei video spediti dall'algoritmo greedy girato su v^* nell'intervallo $[0, h]$. Con $B_F(h)$ indichiamo i bit mandati dallo schedule F fino al tempo h compreso.
Dal claim del Greedy Stays Ahead sappiamo che $X_G(h) \geq X_F(h)$, ossia $r \cdot h - B_{PG}(h) \geq r \cdot h - B_F(h)$, quindi $B_{PG}(h) \leq B_F(h)$. Essendo F valido in senso rilassato: $B_F(h) \leq r \cdot h + r \cdot \sum_{i \in \{1,\dots,k-1\}} t_i - \sum_{i \in \{1,\dots,k-1\}} b_i$. D'ora in poi, per brevità, chiamiamo $\Gamma = r \cdot h + r \cdot \sum_{i \in \{1,\dots,k-1\}} t_i - \sum_{i \in \{1,\dots,k-1\}} b_i$. Abbiamo appena dimostrato che $B_{PG}(h) \leq B_F(h) \leq \Gamma$
Se l'algoritmo greedy gira sull'istanza completa v, sappiamo che non è valido quindi $B_{PG}(h) + \sum_{i \in \{1,\dots,k-1\}} b_i > r \cdot h + r \cdot \sum_{i \in \{1,\dots,k-1\}} t_i$, ossia $B_{PG}(h) > r \cdot h + r \cdot \sum_{i \in \{1,\dots,k-1\}} t_i - \sum_{i \in \{1,\dots,k-1\}} b_i = \Gamma$.
Si ha quindi che $\Gamma < B_{PG}(h) \leq \Gamma$, assurdo. (**Domanda**: se fosse stato $\Gamma \leq B_{PG}(h)$ che sarebbe successo?) □

Facciamo il punto della situazione: volevamo dimostrare per assurdo che se $\exists$ schedule valido $\rightarrow$ lo schedule ottenuto dall'algoritmo greedy è valido. Quindi l'obiettivo è di dimostrare che lo statement *" $\exists$ uno schedule valido F e l'algoritmo greedy dice che nessuno schedule è valido"* è falso.
Per il claim sulla parte stile Exchange Argument, però, se $F(v)$ allora $F(v^*)$ restituisce uno schedule valido in senso rilassato; ma questo implica (per il claim qua sopra) che l'algoritmo greedy è valido. Assurdo. $\square$