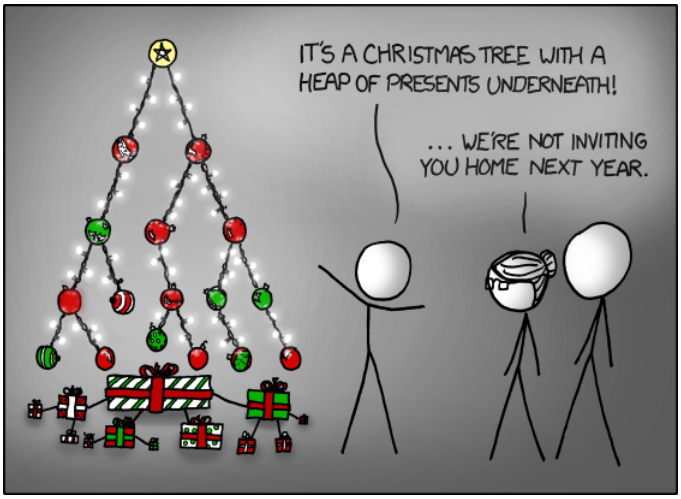
Problem Set 3

(quello di natale, ovvero:
algoritmi sotto l'albero)
docente: Luciano Gualà

Esercizio 0 *(esercizio di preparazione allo spirito natalizio)*
Per accogliere lo spirito natalizio di questo Problem Set si consiglia prima di preparare un albero di natale adeguato. Un suggerimento è fornito in Figura 1.

Figura 1: Un esempio di albero di natale adeguato.



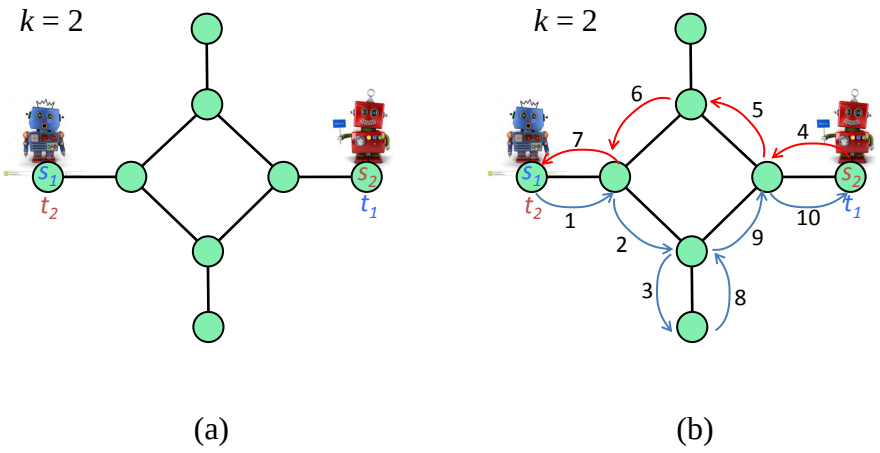
Esercizio 1 *(Babbo Natale e le nuove tecnologie)*
Quest'anno Babbo Natale per consegnare i regali, non potendo usare la slitta perché delle associazioni animaliste gli hanno fatto causa per sfruttamento delle renne, si è dotato di robot che posso essere telecomandati a distanza. Non è stato facile cambiare le sue abitudini, ma ce l'ha quasi fatta. Gli restano infatti da consegnare solo gli ultimi due regali. Ce la farà?

La mappa del mondo è rappresentata come un grafo non orientato e non pesato $G = (V, E)$ dove i due robot si possono muovere. All'inizio i due robot, ognuno con il proprio regalo da consegnare, sono posizionati su due nodi del grafo, diciamo s_1 ed s_2, mentre le due case in cui vanno consegnati i regali si trovano su t_1 e t_2. In ogni istante di tempo Babbo Natale può effettuare la seguente mossa: ordinare ad uno dei due robot di spostarsi dal nodo su cui è a un nodo adiacente (percorrendo un arco del grafo). L'obiettivo è quindi portare il robot che si trova su s_1 nel nodo t_1 ed il robot che si trova su s_2 nel nodo t_2. Le antenne dei robot, però, soffrono di problemi di interferenze: se i robot finiscono troppo vicini l'uno con l'altro non riescono più a ricevere il segnale e quindi a muoversi. Per questo motivo si vuole che i robot in ogni istante di tempo siano sempre a distanza reciproca (nel

grafo) di almeno k, dove k è un parametro del problema. Un esempio di istanza e soluzione è fornito in Figura 2.

Aiutate Babbo Natale ad andare in vacanza progettando un algoritmo che trovi il numero minimo di mosse che porta i robot nelle posizioni desiderate.

Figura 2: (a) Un'istanza del problema: i due robot devono scambiarsi di posto, in ogni istante di tempo si può muovere solo uno dei due lungo un arco adiacente al nodo in cui si trova, i due robot devono sempre trovarsi a distanza almeno k. (b) una possibile soluzione da 10 mosse.



(a) (b)

Soluzione Esercizio 1 L'idea è quella di usare la tecnica della riduzione e ricondurre il problema in esame al problema del calcolo di un cammino minimo in un grafo ausiliario. Abbiamo usato questa tecnica già diverse volte. Il grafo ausiliario in questione, in particolare, ricorda –in senso qualitativo– quello che abbiamo definito per trovare il numero minimo di mosse per risolvere il cubo di Rubik. Più formalmente, il grafo ausiliario di cui abbiamo bisogno è un grafo che riesca a modellare nella sua topologia le posizioni ammissibili (a distanza almeno k) che i robot possono avere sul grafo originale, e le possibili mosse effettuabili dai robot ancora una volta non violando il vincolo sulla distanza.

Chiameremo tale grafo come *grafo delle configurazioni* e sarà indicato con $G' = (V', E')$. Nel grafo delle configurazioni i nodi sono una coppia *ordinata* $\langle u, v \rangle$ dove u e v sono nodi del grafo G, un nodo $\langle u, v \rangle$ appartiene a V' se e soltanto se $d_G(u, v) \geq k$, dove $d_G(u, v)$ denota la distanza in G fra u e v.

Formalmente G' è definito come segue :

- $V' = \{\langle u, v\rangle : u, v \in V \text{ e } d_G(u, v) \geq k\}$
- $E' = \{(\langle u, v\rangle, \langle x, y\rangle) : (u = x \text{ e } (v, y) \in E) \text{ o } (v = y \text{ e } (u, x) \in E)$

Intuitivamente i nodi in V' rappresentano tutti i possibili modi in cui i robot possono trovarsi senza violare il vincolo relativo all'interferenza delle antenne: uno nodo $\langle x, y\rangle$ indica che il robot 1 si trova su x e il robot 2 su y. Gli archi invece, modellano le possibili mosse: c'è un arco fra due configurazioni se è possibile passare da una all'altra spostando uno dei due robot lungo un arco.

Una volta comprese le proprietà di G' è facile convincerci che per trovare la sequenza di spostamenti minimi che risolva il problema basta calcolare un cammino minimo tra $\langle s_1, s_2\rangle$ e $\langle t_1, t_2\rangle$ in G'. Naturalmente si sta supponendo che le posizioni iniziali e target rispettino il vincolo sulla distanza, ciò fa si che i nodi in questione si trovino in V'. Nel caso in cui in G' non esista alcun cammino tra il nodo sorgente e il nodo destinazione, non è possibile trovare una sequenza di mosse dei robot che risolvano il problema.

Per quanto riguarda la complessità dell'algoritmo, invece, dobbiamo rispondere ad alcune domane. Quanto costa costruire il grafo G'? Quale sarà la sua dimensione in relazione alla dimensione di G? Quanto costa calcolare il cammino minimo richiesto su G'?

Diamo innanzitutto una stima sulla dimensione di G'. Abbiamo visto che i nodi di G' corrispondono a una coppia ordinata di elementi in V, da ciò deriva che $|V'| \leq n^2$ e quindi $|V'| = O(n^2)$.

Per quanto riguarda gli archi, con una prima stima grossolana possiamo dire che $|E'| = O(n^4)$ in quanto al più gli archi di un grafo possono essere quadratici rispetto al numero dei nodi. Tuttavia è possibile fare un analisi più raffinata che ci da un bound migliore. Denotiamo con $\delta_H(x)$ il grado di un generico nodo x in un generico grafo H. Abbiamo:

$$|E'| \leq \sum_{\langle u,v\rangle \in V'} \delta_{G'}(\langle u, v\rangle) \leq \sum_{\langle u,v\rangle \in V'} (\delta_G(u) + \delta_G(v)) = \sum_{\langle u,v\rangle \in V'} \delta_G(u) + \sum_{\langle u,v\rangle \in V'} \delta_G(v) \leq$$
$$\sum_{u,v \in V} \delta_G(u) + \sum_{u,v \in V} \delta_G(v) \leq n \sum_{u \in V} \delta_G(u) + n \sum_{v \in V} \delta_G(v) \leq 2nm + 2nm = O(nm)$$

Andiamo ora a vedere la complessità che ha la costruzione del grafo G' e il calcolo del cammino minimo su di esso. Per costruire il grafo è necessario calcolare le distanze tra tutte le coppie di nodi in V in modo da poter aggiungere a V' solo quelle a distanza almeno k. Ricordando che tramite la visita BFS è possibile calcolare le distanze rispetto ad un nodo sorgente, per trovare le distanze fra tutte le coppie è necessario effettuare n visite BFS (ogni volta utilizzando un nodo diverso come sorgente). Avendo una visita costo $O(m)$ (utilizzando liste di adiacenza) avremo che il costo di costruzione del grafo è $O(nm)$. Una volta costruito G' per trovare il cammino minimo da $\langle s_1, s_2\rangle$ a $\langle t_1, t_2\rangle$ utilizzeremo una visita BFS su G' utilizzando come sorgente il nodo $\langle s_1, s_2\rangle$. Tale computazione avrà un costo di $O(|E'| + |V'|) = O(mn + n^2) = O(nm)$. In definita in tempo totale $O(nm)$ è possibile trovare la sequenza di mosse per i robot.

Esercizio 2 *(l'uccisione di Babbo Natale)*
Anche quest'anno a Chinonsò il Grinch sta cercando di far fallire il natale. Ha un piano perfetto: ucciderà Babbo Natale prima che consegni i regali. Sa che il panzone con la barba arriverà alle porte di Chinonsò fra esattamente Δ ore. E ha scoperto dove trovare il coltello magico, l'unica arma che può uccidere Babbo Natale. Il coltello si trova dentro una grotta, sotto il cadavere del grillo. Il Grinch non sa se farà in tempo ad andare a recuperare il coltello prima che Babbo Natale arrivi, ma ha un complice che (se serve) può recuperare il coltello per lui: il suo fedele cane e amico Max. La mappa di Chinonsò è rappresentata come un grafo non orientato $G = (V, E)$. Il Grich e Max si trovano rispettivamente nei nodi v_G e v_M, mentre la grotta che nasconde il coltello e le porte di Chinonsò sono rispettivamente nei nodi v_C e v_P. Attraversare un arco richiede un'ora per il Grinch e due ore per Max (che è più lento del suo amico bipede). Il coltello può essere recuperato sia dal Grinch che da Max, ma deve essere il Grinch a sferrare il colpo fatale a Babbo Natale, e questo può avvenire solo in v_P nel momento esatto in cui Babbo Natale si presenterà. Max e il Grinch possono scambiarsi il coltello in qualsiasi nodo del grafo in un tempo trascurabile, così come è trascurabile il tempo necessario, una volta in v_C, per recuperare il coltello. Mancano esattamente Δ ore. La neve sta cadendo. Le stelle sono punte di spillo. Progettate un algoritmo con complessità lineare nella dimensione del grafo che decide se c'è una strategia che consente al Grinch di assassinare Babbo Natale.

Soluzione Esercizio 2 Il problema sembra a prima vista piuttosto complicato, ma tutto diventa più chiaro se si ragiona sulla struttura della soluzione cercata. La domanda cruciale è la seguente: quale è la migliore soluzione se il Grinch e Max si incontrano su uno specifico nodo x? Una volta risposto a questa domanda l'idea è quella di "provare" tutti i possibili nodi $x \in V$, guardare la migliore soluzione per quel nodo di incontro, e fra tutte le n soluzioni prendere la migliore.

Assumiamo quindi che sia Max che andrà a prendere il coltello e che il passaggio del coltello fra Max e il Grinch avvenga sul nodo x. Denotiamo con $d(u, v)$ la distanza in G fra u e v. Il tempo (in ore) impiegato dal Grinch per arrivare su x è chiaramente $d(v_G, x)$, mentre quello di Max è $2d(v_M, v_C) + 2d(v_C, x)$. Si noti il fattore 2 nell'ultima formula, necessario perché Max necessita di due ore per attraversare un arco del grafo. Qundi, visto che i due amici si spostano contemporaneamente, si incontreranno su x esattamente dopo $\max\{d(v_G, x), 2d(v_M, v_C) + 2d(v_C, x)\}$ ore, e il Grinch potrà arrivare alle porte della citta di Chinosò con un numero di ore pari a:

$$\tau(x) := \max\{d(v_G, x), 2d(v_M, v_C) + 2d(v_C, x)\} + d(x, v_P).$$

Si noti che, dato x, possiamo calcolare $\tau(x)$ in tempo $O(m + n)$, in quanto è sufficiente calcolare quattro cammini minimi in G fra quattro coppie di nodi. Visto che dobbiamo calcolare $\tau(x)$ per ogni possibile $x \in V$, questo approccio porterebbe ad un costo complessivo di $O(mn)$. Possiamo in realtà fare meglio. L'osservazione cruciale è che se conoscessimo la distanza fra ogni x e i quattro nodi v_G, v_M, v_C, v_P, il valore di $\tau(x)$ sarebbe disponibile in tempo costante. Usando questa idea otteniamo il seguente algoritmo. Si noti che è necessario anche considerare a parte la soluzione in cui il Grinch va da solo a prendere il coltello e poi alle porte della città per aspettare Babbo Natale.

Algorithm 1: `LUccisioneDiBN`$(G, v_G, v_M, v_C, v_P, \Delta)$

```
1 calcola distanze/albero dei cammini minimi di G con radici v_G, v_M, v_C, v_P ;
2 if d(v_G, v_C) + d(v_C, v_P) ≤ Δ then
3 |   il Grinch va da solo a prendere il coltello e poi ad uccidere Babbo Natale
4 else
5 |   x* = arg min_{x∈V} τ(x);
6 |   if τ(x*) ≤ Δ then
7 |   |   il Grinch e Max si scambiano il coltello in x* e poi il Grinch va ad uccidere
  |   |     Babbo Natale
8 |   else
9 |   |   Banno Natale si salva!
```

La correttezza dell'algoritmo segue dal ragionamento sulla struttura della soluzione e dal fatto che proviamo tutti i possibili nodi x (più la soluzione in cui il Grinch va solo).

Per quanto riguarda la complessità, invece, la Linea 1 consiste in 4 visite BFS, di costo complessivo $O(m+n)$, la Linea 5 costa $O(n)$ (perché come già osservato $\tau(x)$ può essere calcolato in tempo costante per uno specifico x), mentre tutte le altre linee hanno costo costante. Abbiamo quindi progettato un algoritmo che risolve in problema in tempo $O(m+n)$, ovvero in tempo lineare nella dimensione del grafo.

Esercizio 3 *(e tornava l'emigrante)*
Fabiano è a Tor Vergata e vuole tornare a casa per natale ma non ha la macchina e deve raggiungere la sua lontanissima città, Potenza. La rete di trasporti è modellata come un grafo orientato $G = (V, E, w)$ dove il peso di un arco (u, v) rappresenta il tempo necessario per spostarsi con un opportuno mezzo da u a v. Fabiano ha diversi abbonamenti (della metro, dei treni regionali e interregionali, tessera per affittare biciclette, slitte e monopattini) che gli permettono di viaggiare gratis su molte tratte. Però ci sono delle tratte che non sono coperte dai suoi abbonamenti. Più precisamente, alcuni degli archi del grafo sono *a pagamento* e ogni volta che Fabiano vuole attraversarne uno deve pagare 1 euro. Nel portafoglio ha k euro. Tor Vergata è sul nodo s mentre casa sua è nel nodo t. L'obiettivo di Fabiano è arrivare il prima possibile a casa. Progettare un algoritmo efficiente che calcoli il miglior cammino che porta Fabiano a casa.

Soluzione Esercizio 3 Se chiamiamo gli archi corrispondenti alle tratte a pagamento archi *blu*, il problema può essere riformulato come quello di trovare un cammino minimo da s a t vincolato ad avere *al più* k archi blu.

L'idea è ancora una volta quella di usare la tecnica della riduzione e ricondurre il problema in esame al problema del calcolo di un cammino minimo in un grafo ausiliario G' che "codifichi" in qualche modo i vincoli del problema, in particolare quello di utilizzare al più k archi blu.

In particolare dato il grafo G, G' sarà composto da $k+1$ "copie" di G, ognuna delle quali è essenzialmente definita come G senza però gli archi blu. Le copie sono organizzate in "livelli", ogni livello indica il numero di archi blu utilizzati. Le copie ai livelli $0, \cdots, k-1$

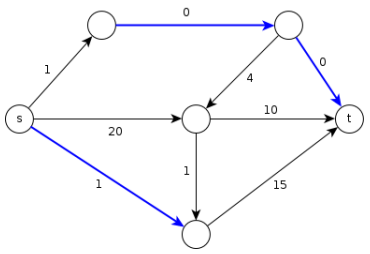
sono collegate nel seguente modo: per ogni arco blu (u, v) uscente da u, si rimpiazza tale arco con l'arco (u, v') dove v' è la replica del nodo v appartenente al livello successivo. Nella copia al livello k non ci sono archi blu. Infine tutti i nodi t di ogni copia sono collegati con un arco diretto di peso 0 ad un unico nuovo nodo destinazione chiamato T. L'idea è questa: in ogni livello ci si può spostare liberamente senza usare però archi blu, se si vuole cambiare livello e andare al successivo, invece, si è costretti a usare esattamente un arco blu. Quindi se si considera un generico cammino da s (nella copia di livello 0) a un generico nodo al livello i, questo cammino ha esattamente i archi blu.

Più formalmente, G' è definito nel seguente modo:

- **nodi:** $\forall v \in V$ ho $k+1$ copie $v_0, v_1, \cdots, v_k$, è presente inoltre un unico nodo destinazione T
- **archi:**
  - $\forall$ arco (u, v) non blu ho gli archi (u_i, v_i), $i = 0, \cdots, k$ con $w(u_i, v_i) = w(u, v)$;
  - $\forall$ arco (u, v) blu ho gli archi (u_i, v_{i+1}), $i = 0, \cdots, k-1$ con $w(u_i, v_{i+1}) = w(u, v)$;
  - ho gli archi (t_i, T) con $w(t_1, T) = 0$ per $i = 0, \cdots, k$.

Al fine di chiarire come G' è definito, mostriamo un esempio: l'istanza G (Figura 3) e il corrispondente grafo G' per $k = 1$ (Figura 4).

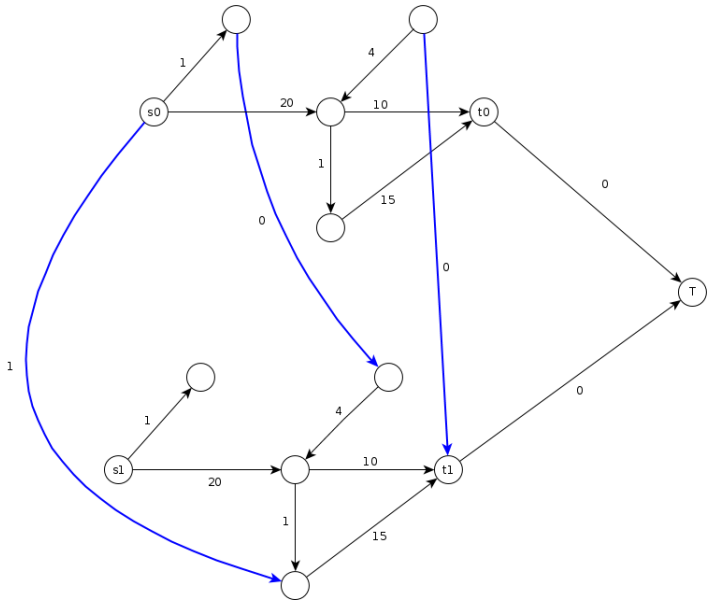
Figura 3: Un'istanza del problema. Le tratte a pagamento sono mostrate come archi di colore blu.



Una volta costruito il grafo G' per trovare la soluzione al nostro problema sarà sufficiente trovare il cammino minimo π' in G' tra il nodo s_0 ed il nodo T e poi riconvertilo in un cammino π in G. In particolare sia π' il cammino trovato dall'algoritmo su G', il cammino π cercato su G è definito nel seguente modo. Prima rimuoviamo da π' l'arco (t_j, T), dove t_j è la replica di t da cui abbiamo raggiunto il target T. Poi, un generico arco in π' del tipo (u_i, v_i) (ovvero un arco che non ci fa cambiare livello) è sostituito con l'arco (u, v), e un generico arco in π' della forma (u_i, v_{i+1}) (ovvero un arco che ci fa cambiare livello) viene sostituito con l'arco blu (u, v).

È facile osservare che il cammino π cosi prodotto contiene al più k archi blu, visto che ogni volta che ne viene utilizzato uno si avanza al livello successivo, ed una volta giunti all'ultimo livello non ci sono più archi blu da utilizzare. Ora una domanda più intrigante è questa: è π il cammino di costo minimo fra tutti quelli che usano al più k archi

Figura 4: Il grafo ausiliario G' relativo all'istanza in Figura 3 per $k = 1$.



blu? La risposta a questa domanda è sì e deriva dalla seguente proprietà (la cui semplice dimostrazione è lasciata come esercizio allo studente).

Proprietà: esiste un cammino in G da s a t che ha al più k archi blu di lunghezza W se e soltanto se esiste un cammino in G' da s_0 a T di lunghezza W.

La proprietà appena enunciata da sola è sufficiente per dimostrare la correttezza dell'algoritmo. Ma quale è la complessità temporale? Si ha che $|V'| = \Theta(k|V|)$ mentre $|E'| = \Theta(k|E|)$ da ciò la complessità di costruzione di G' risulta essere $O(k|V|+k|E|)$. L'applicazione dell'algoritmo di Dijkstra su G' avrà complessità $O(k|E| + k|V| \log |V|)$ quindi l'algoritmo proposto ha complessità totale $O(k|E| + k|V| \log |V|)$.