Esercitazione
12 gennaio 2023

Esercizio 1:
Babbo Natale e le nuove tecnologie (Ex 1, PS 2020)

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?

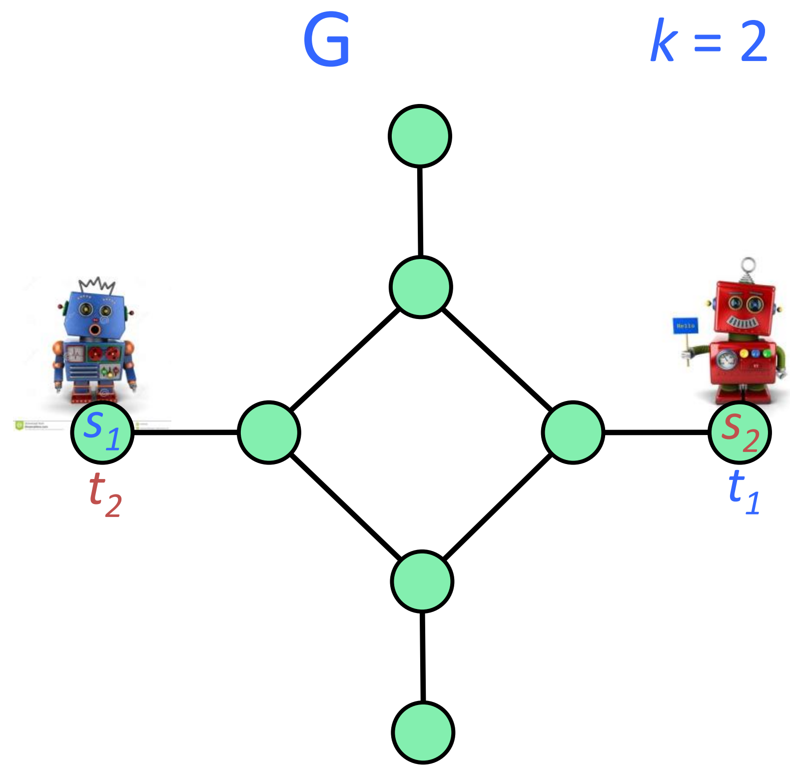
La mappa del mondo è rappresentata come un grafo non orientato e non pesato $G = (V, E)$ dove i due robot si possono muovere. All'inizio i due robot, ognuno con il proprio regalo da consegnare, sono posizionati su due nodi del grafo, diciamo s_1 ed s_2, mentre le due case in cui vanno consegnati i regali 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.

obiettivo: spostare il robot blu da s_1 a t_1
& il robot rosso da s_2 a t_2

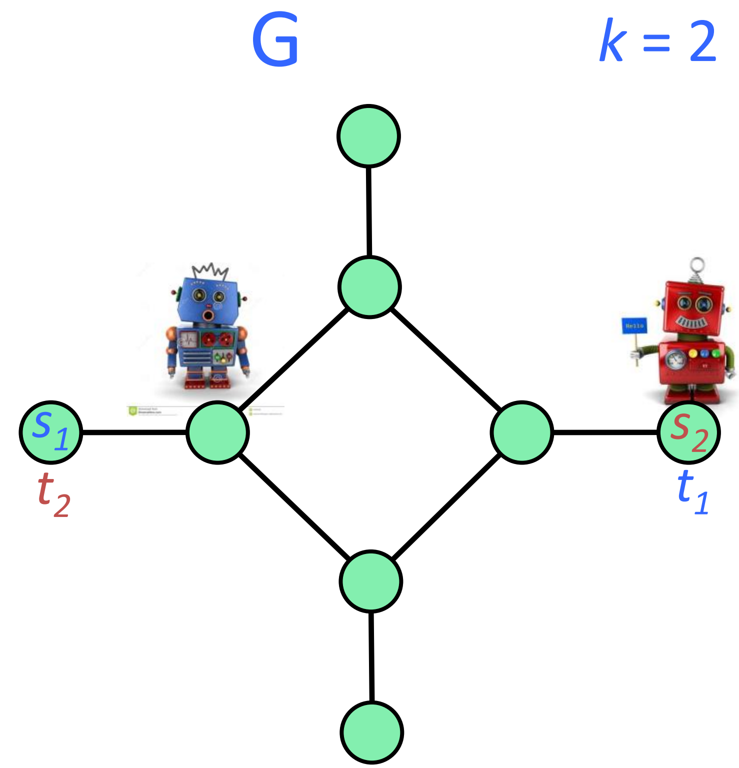
mossa: sposta un robot dal nodo corrente a uno adiacente.

vincolo: i robots devono essere sempre a distanza almeno k fra loro



trovare la sequenza minima di mosse!

obiettivo: spostare il robot blu da s_1 a t_1
& il robot rosso da s_2 a t_2



mossa: sposta un robot dal nodo corrente a uno adiacente.

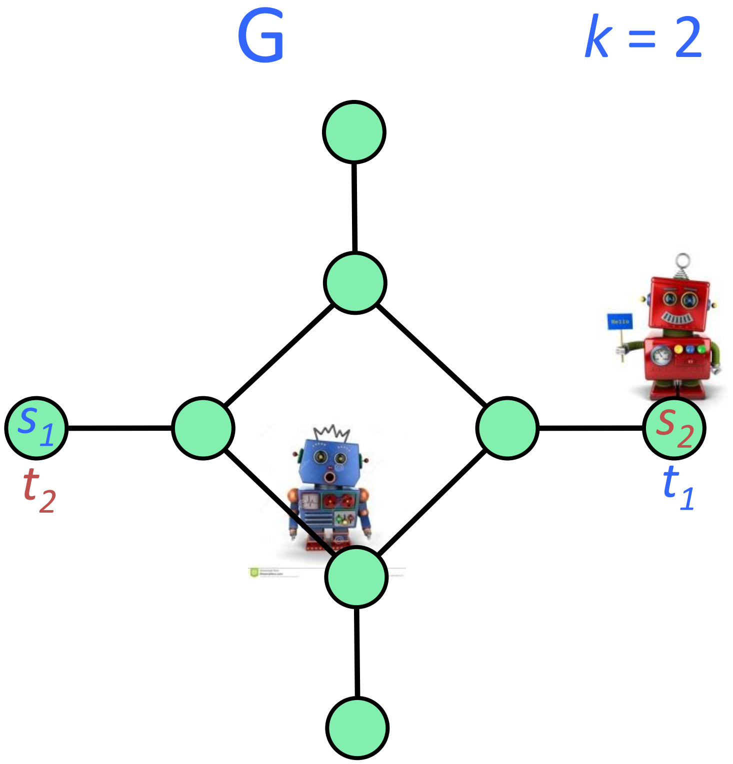
vincolo: i robots devono essere sempre a distanza almeno k fra loro

trovare la sequenza minima di mosse!

obiettivo: spostare il robot blu da s_1 a t_1 & il robot rosso da s_2 a t_2

mossa: sposta un robot dal nodo corrente a uno adiacente.

vincolo: i robots devono essere sempre a distanza almeno k fra loro

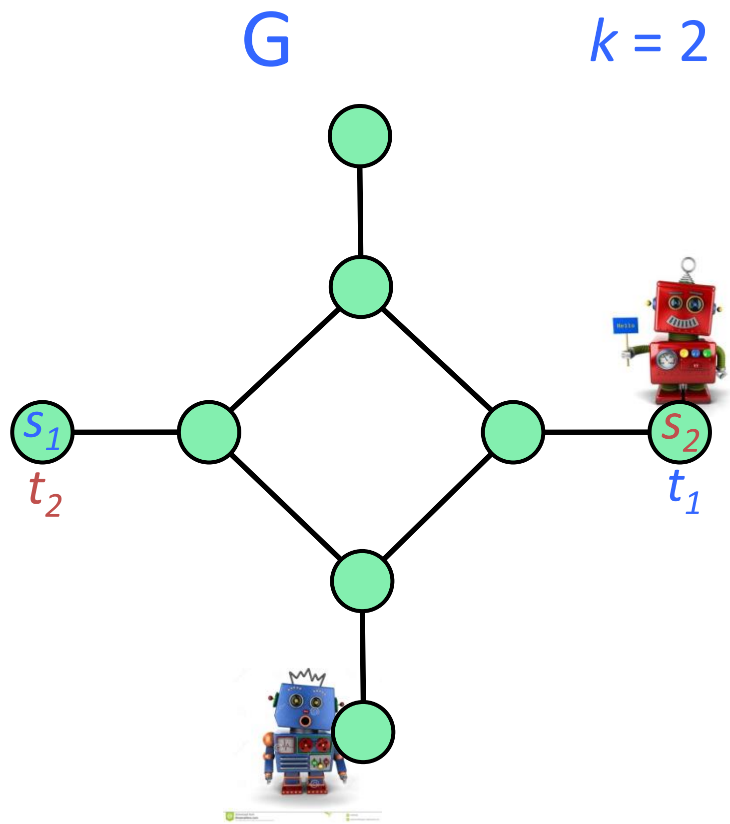


trovare la sequenza minima di mosse!

obiettivo: spostare il robot blu da s_1 a t_1
& il robot rosso da s_2 a t_2

mossa: sposta un robot dal nodo corrente a uno adiacente.

vincolo: i robots devono essere sempre a distanza almeno k fra loro



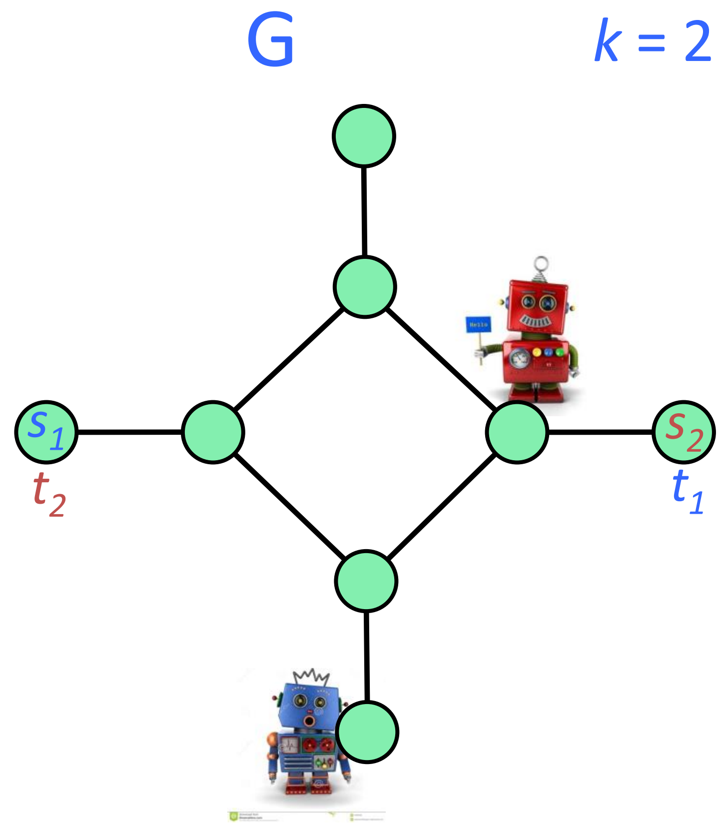
trovare la sequenza minima di mosse!

obiettivo: spostare il robot blu da s_1 a t_1 & il robot rosso da s_2 a t_2

mossa: sposta un robot dal nodo corrente a uno adiacente.

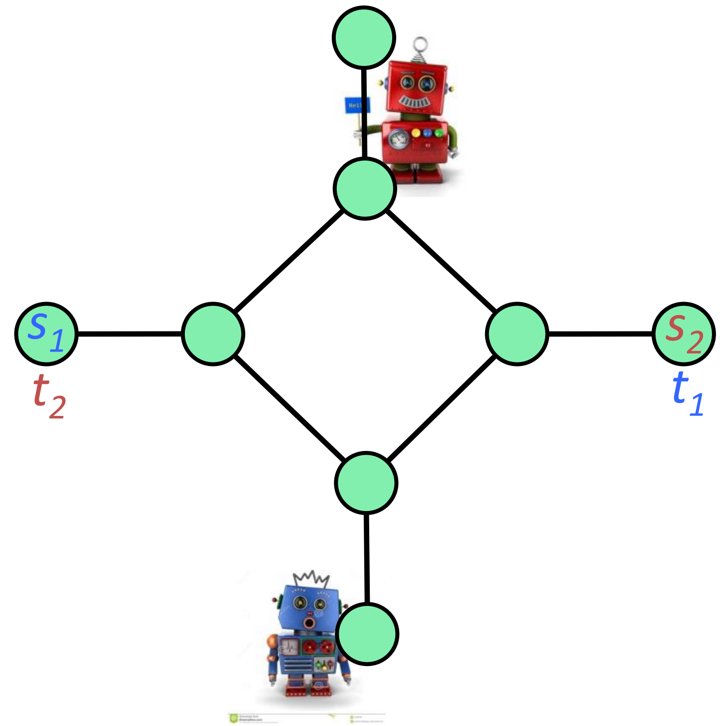
vincolo: i robots devono essere sempre a distanza almeno k fra loro

trovare la sequenza minima di mosse!



obiettivo: spostare il robot blu da s_1 a t_1 & il robot rosso da s_2 a t_2

G $\quad$ $k = 2$



mossa: sposta un robot dal nodo corrente a uno adiacente.

vincolo: i robots devono essere sempre a distanza almeno k fra loro

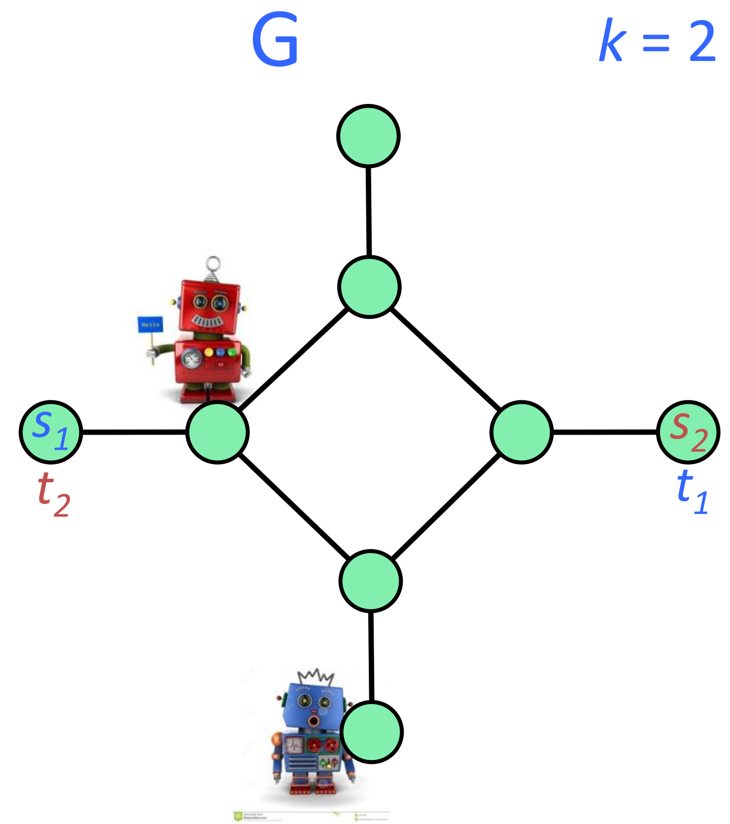
trovare la sequenza minima di mosse!

obiettivo: spostare il robot blu da s_1 a t_1
& il robot rosso da s_2 a t_2

mossa: sposta un robot dal nodo corrente a uno adiacente.

vincolo: i robots devono essere sempre a distanza almeno k fra loro

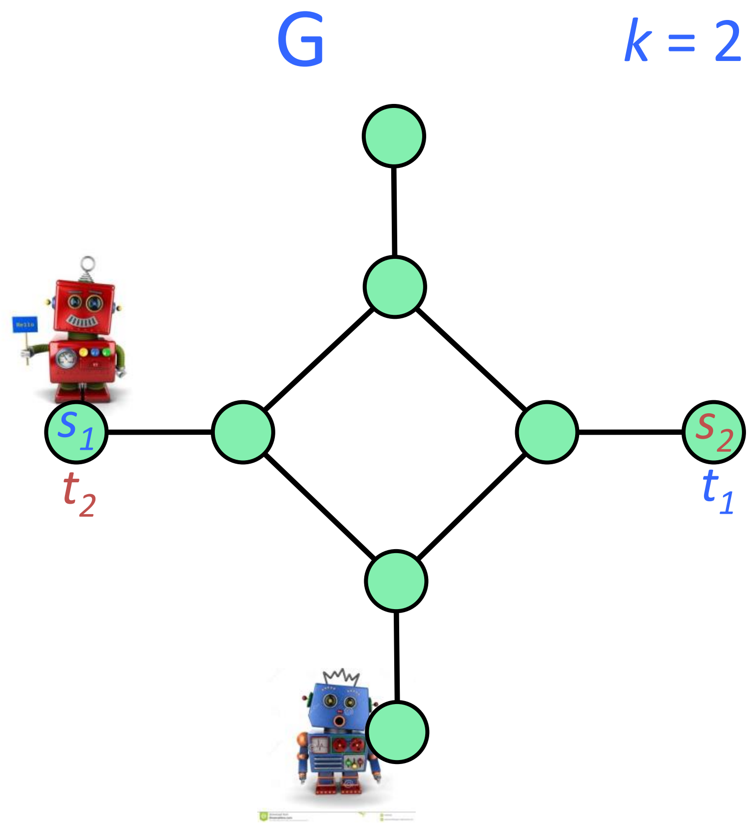
trovare la sequenza minima di mosse!



obiettivo: spostare il robot blu da s_1 a t_1
& il robot rosso da s_2 a t_2

mossa: sposta un robot dal nodo corrente a uno adiacente.

vincolo: i robots devono essere sempre a distanza almeno k fra loro

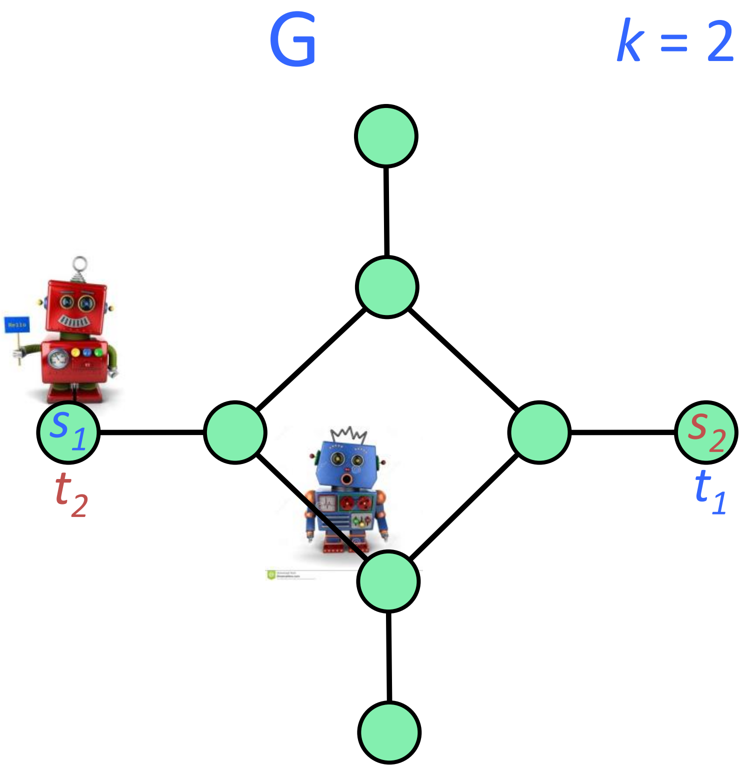


trovare la sequenza minima di mosse!

obiettivo: spostare il robot blu da s_1 a t_1
& il robot rosso da s_2 a t_2

mossa: sposta un robot dal nodo corrente a uno adiacente.

vincolo: i robots devono essere sempre a distanza almeno k fra loro

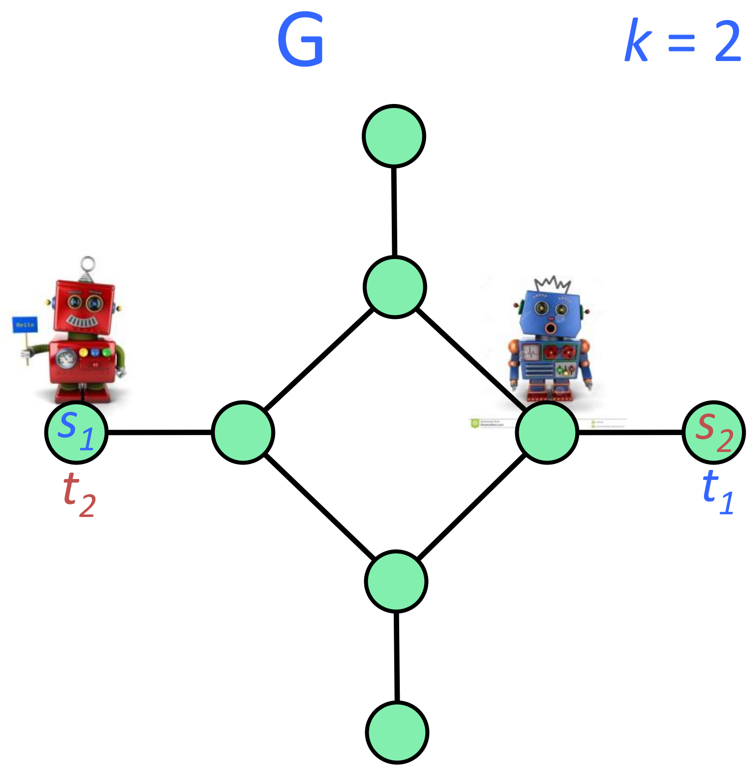


trovare la sequenza minima di mosse!

obiettivo: spostare il robot blu da s_1 a t_1
& il robot rosso da s_2 a t_2

mossa: sposta un robot dal nodo corrente a uno adiacente.

vincolo: i robots devono essere sempre a distanza almeno k fra loro

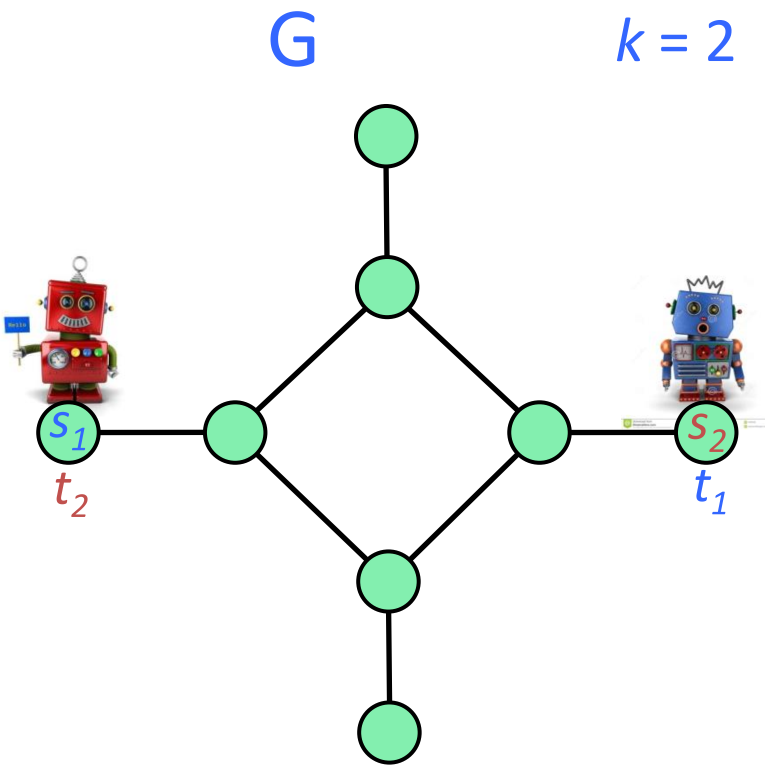


trovare la sequenza minima di mosse!

obiettivo: spostare il robot blu da s_1 a t_1
& il robot rosso da s_2 a t_2

mossa: sposta un robot dal nodo corrente a uno adiacente.

vincolo: i robots devono essere sempre a distanza almeno k fra loro



trovare la sequenza minima di mosse!

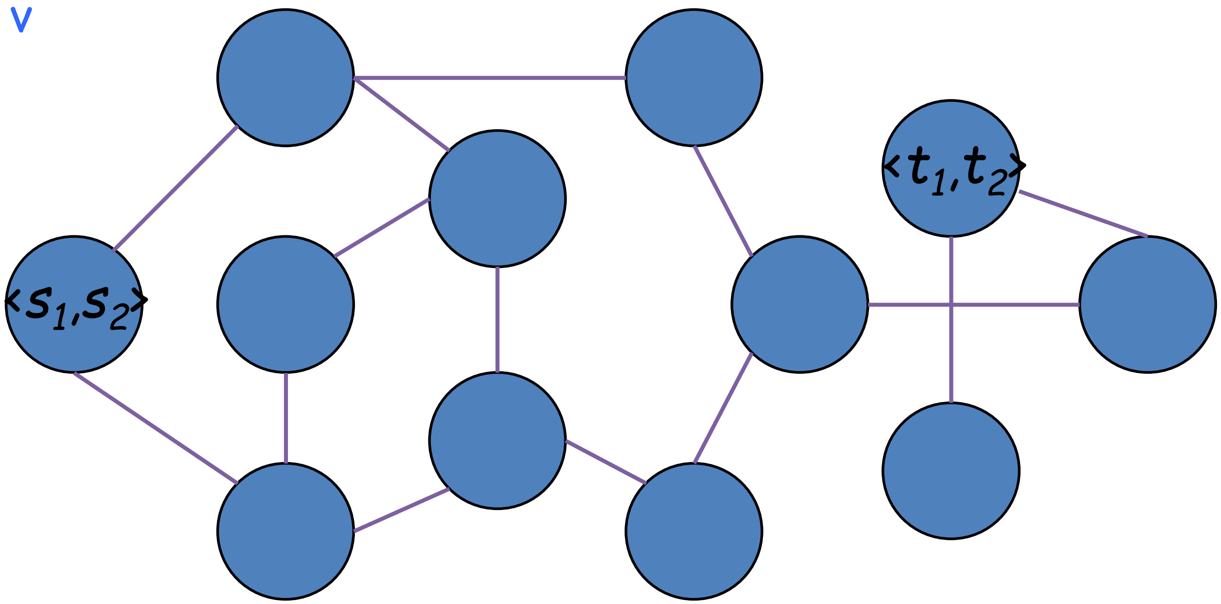
Idea: ridurre il problema al calcolo di un cammino su un opportuno grafo delle configuraizoni

definire un grafo ausiliario G'=(V',E')

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

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

robot blu è su u &
robot rosso è su v



cerco il cammino minimo da <s_1,s_2> a <t_1,t_2>

correttezza:

Proprietà:

Esiste una sequenza di k mosse che porta i robot nella posizione finale se e soltanto se esiste un cammino in G' da $\langle s_1,s_2 \rangle$ a $\langle t_1,t_2 \rangle$ di lunghezza k.

complessità:

dimensione di G':

$$|V'| = O(n^2)$$

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

-costruzione di G':

$O(|V'|+|E'|)$
$=O(mn)$

se trovo le distanze fra
tutte le coppie in $O(mn)$
(n visite BFS, una da ogni vertice)

-calcolo cammino minimo in G':

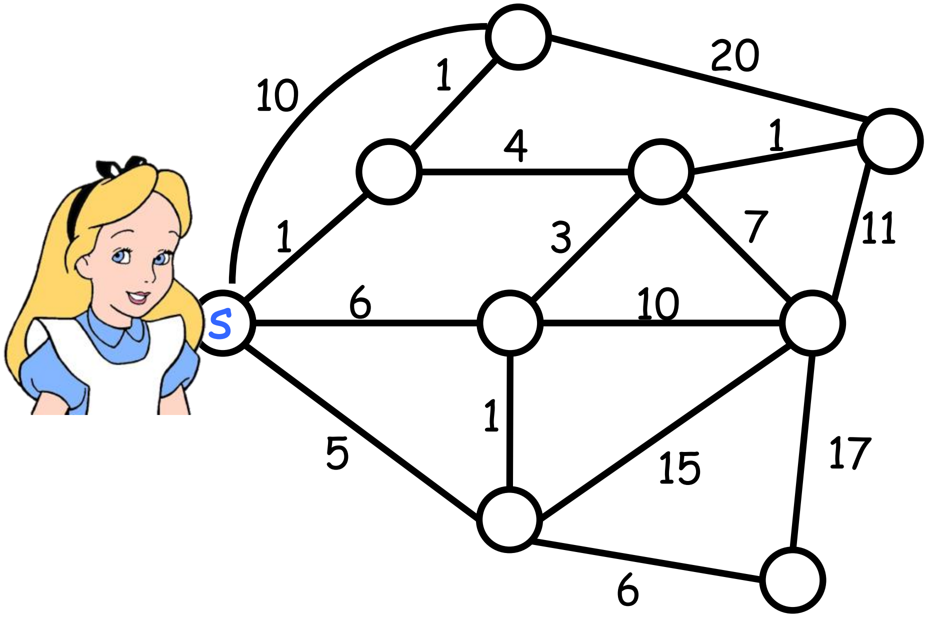
$O(|V'|+|E'|)$
$=O(n^2+mn)$
$=O(mn)$

$O(mn)$

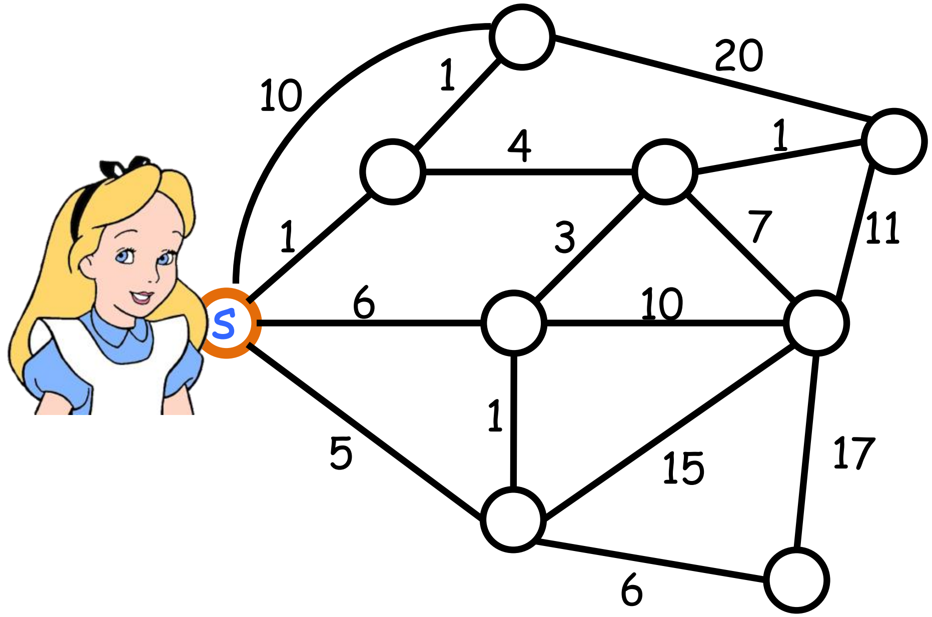
Esercizio 2:

Alice diventa grande.

Esercizio Alice sta crescendo e vuole vedere il mondo, che è modellato come un grafo non orientato $G = (V, E)$ di n nodi e m archi, dove ogni nodo rappresenta un posto bellissimo e ogni arco $(u, v) \in E$ indica la possibilità di spostarsi da u a v. Alice si trova sul nodo s e vorrebbe visitare tutti i restanti $n-1$ posti meravigliosi. Purtroppo, però, non tutti gli archi sono percorribili a tutte le età. Ad ogni arco $e \in E$ quindi è associato un valore $w(e)$ che indica l'età minima necessaria per attraversare l'arco. Diremo che un posto v è raggiungibile da Alice all'età x se esiste un cammino da s a v composto di soli archi di peso minore o uguale a x. Progettate un algoritmo che calcoli l'età minima che consente ad Alice di vedere il mondo intero, ovvero la più piccola età x per cui tutti i posti sono raggiungibili da Alice all'età x.

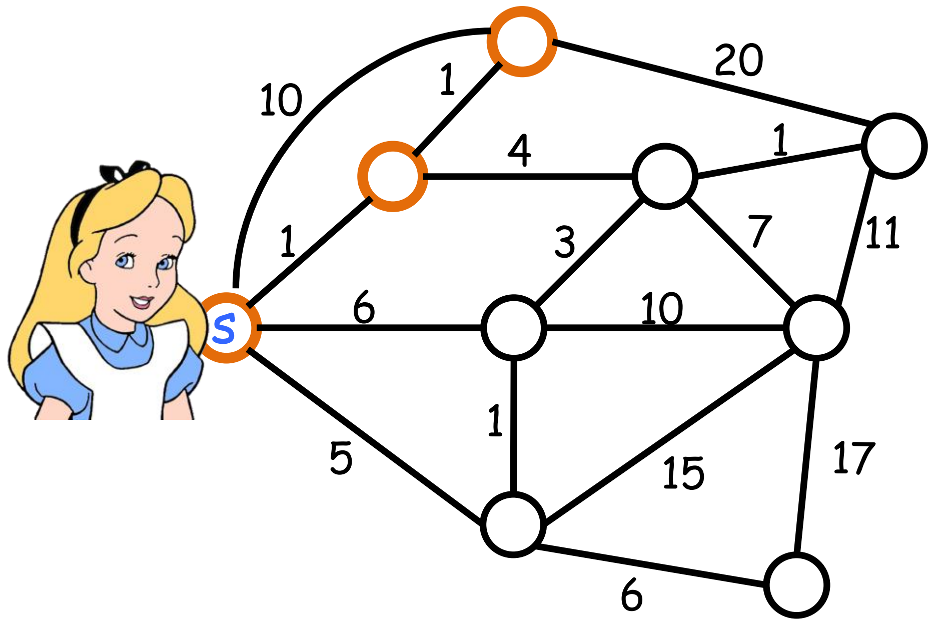


s
10
1
20
1
4
1
1
3
7
11
6
10
1
5
15
17
6

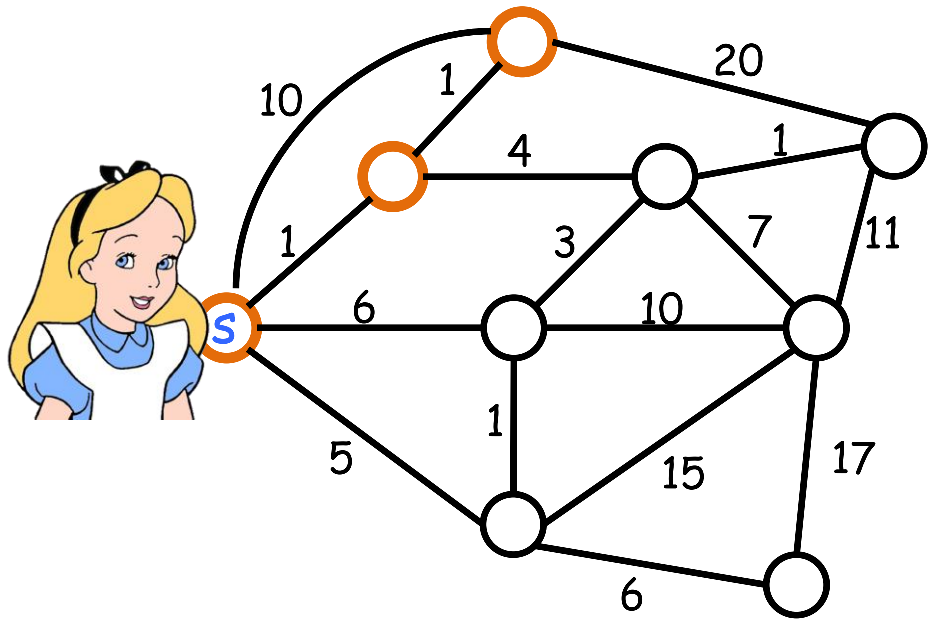


età

0

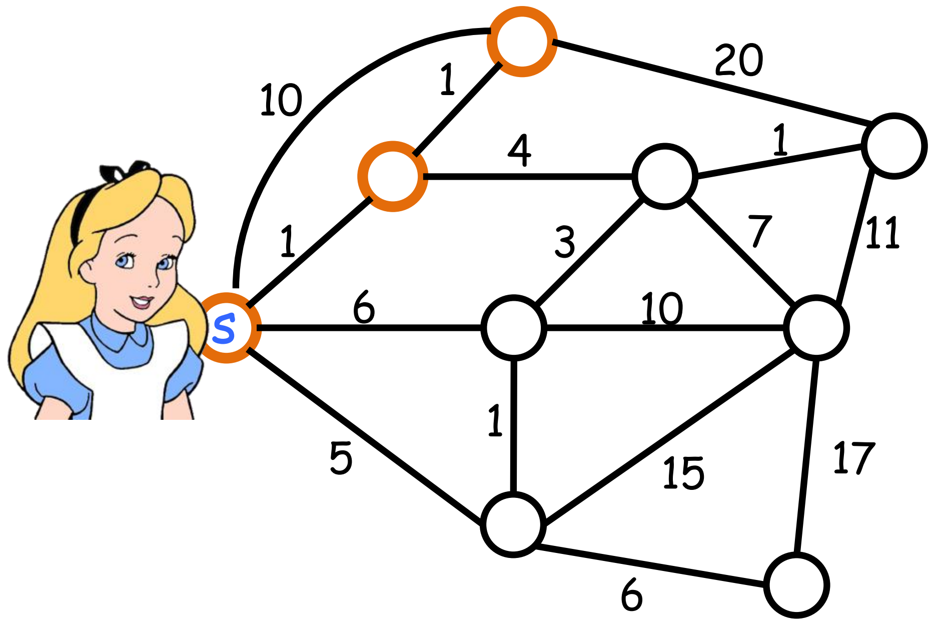

10
1
20
1
4
1
3
7
11
6
10
s
1
5
15
17
6


età
0
1



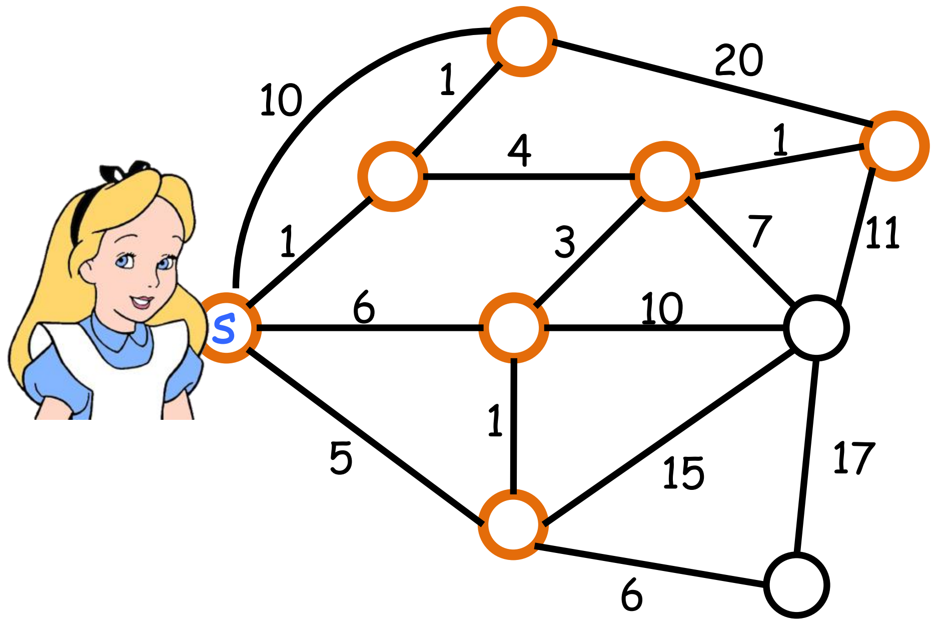
s
10
1
20
1
4
1
3
7
11
6
10
1
5
15
17
6

0
1
2
età



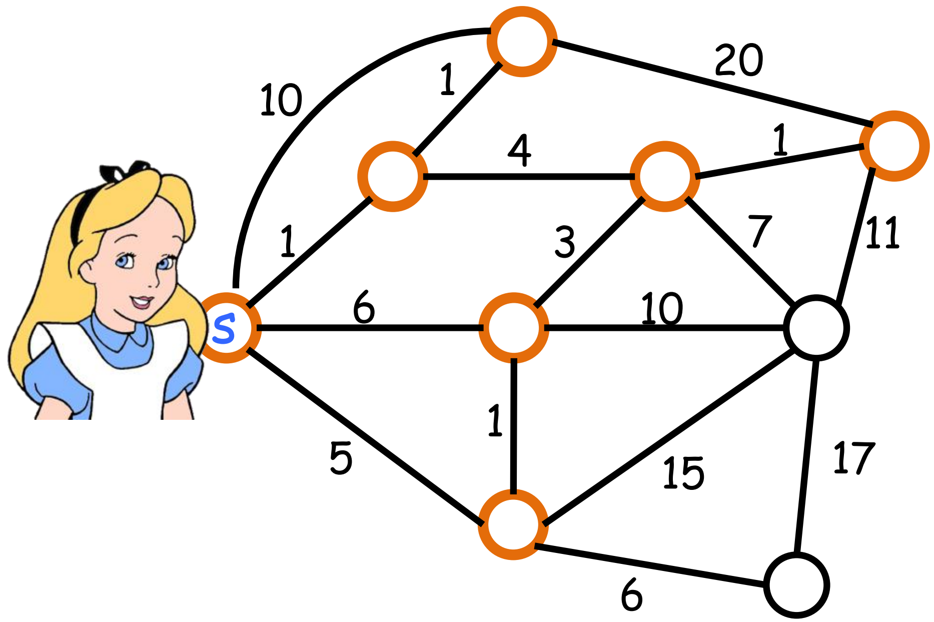
0 1 2 3

età

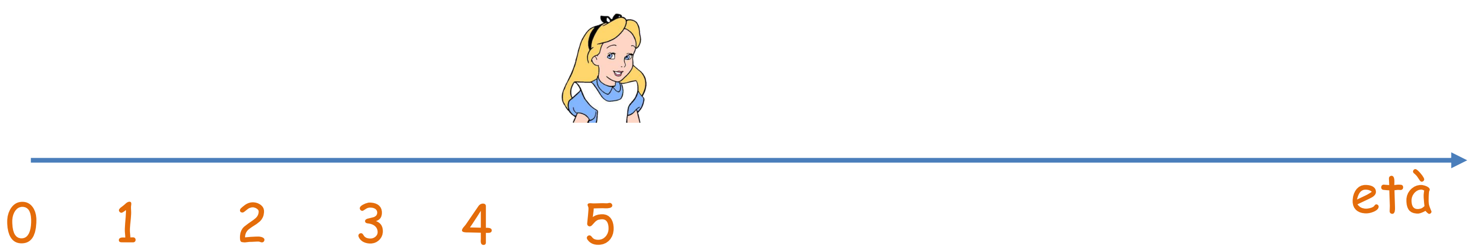


10
1
20
4
1
1
3
7
11
s
6
10
1
5
15
17
6

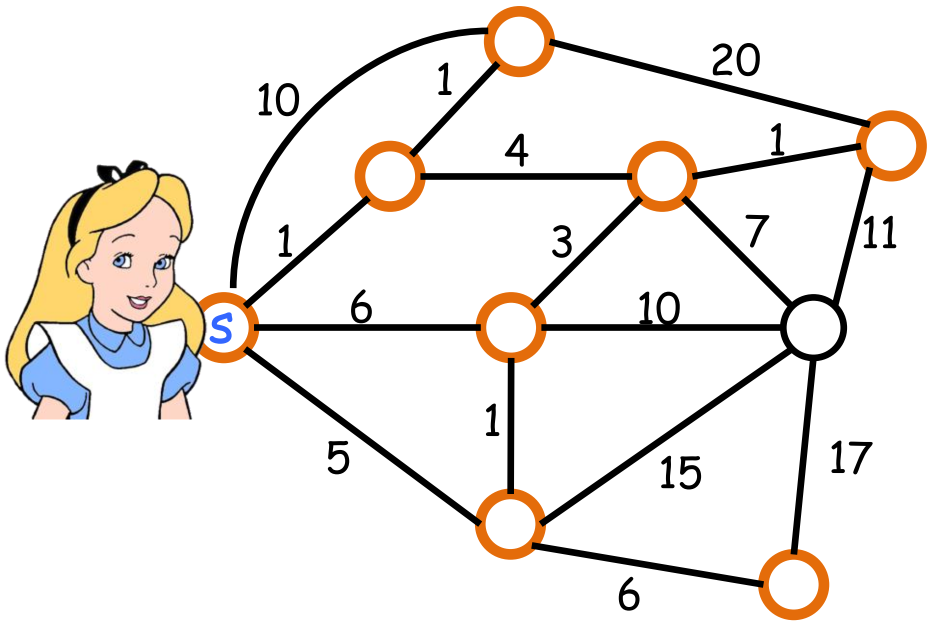
età
0 1 2 3 4



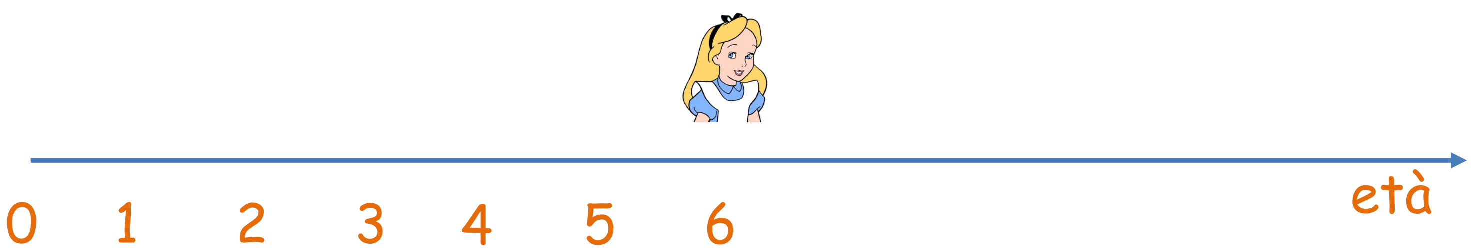
s
10
1
20
4
1
1
3
7
11
6
10
5
1
15
17
6



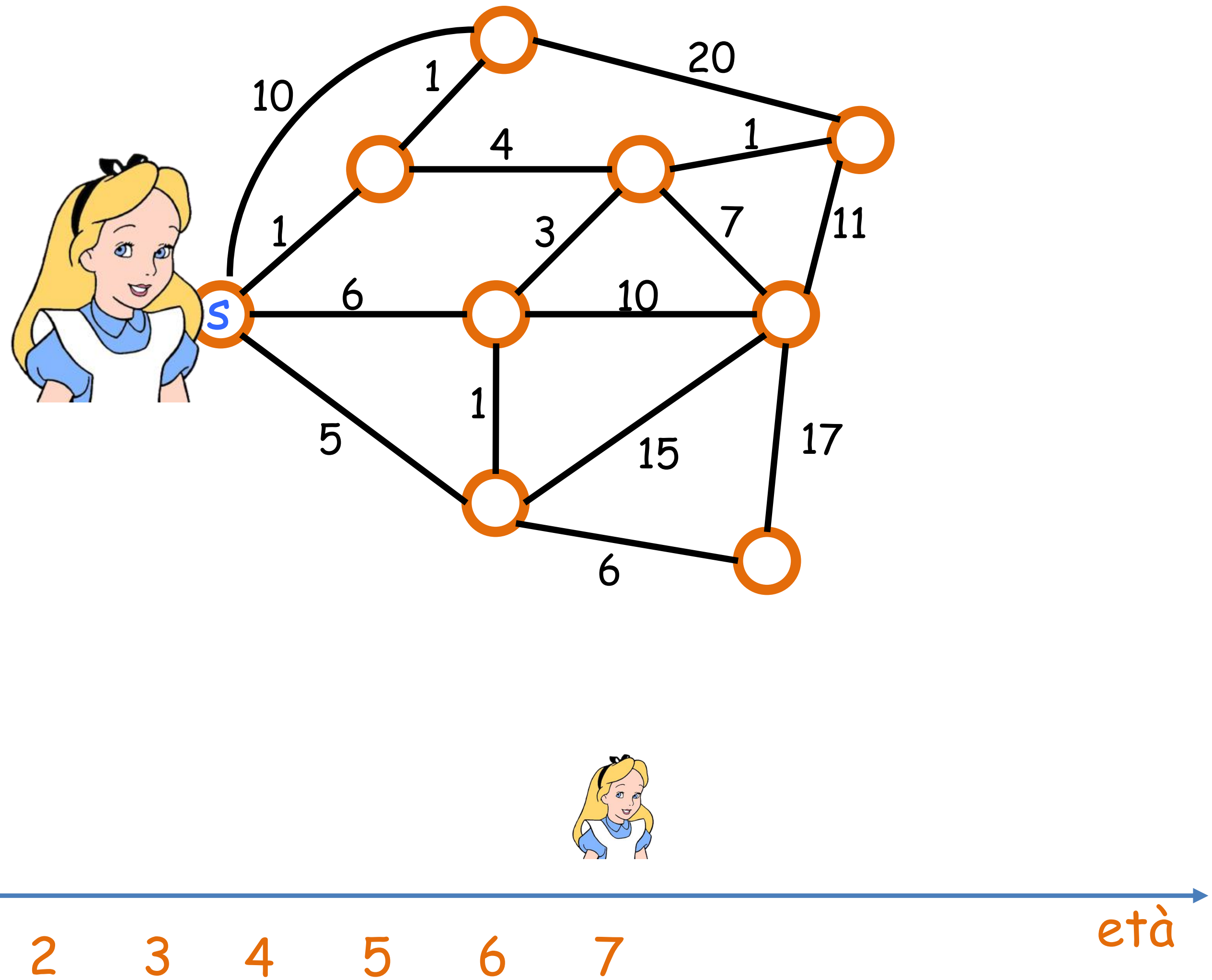
0 1 2 3 4 5
età



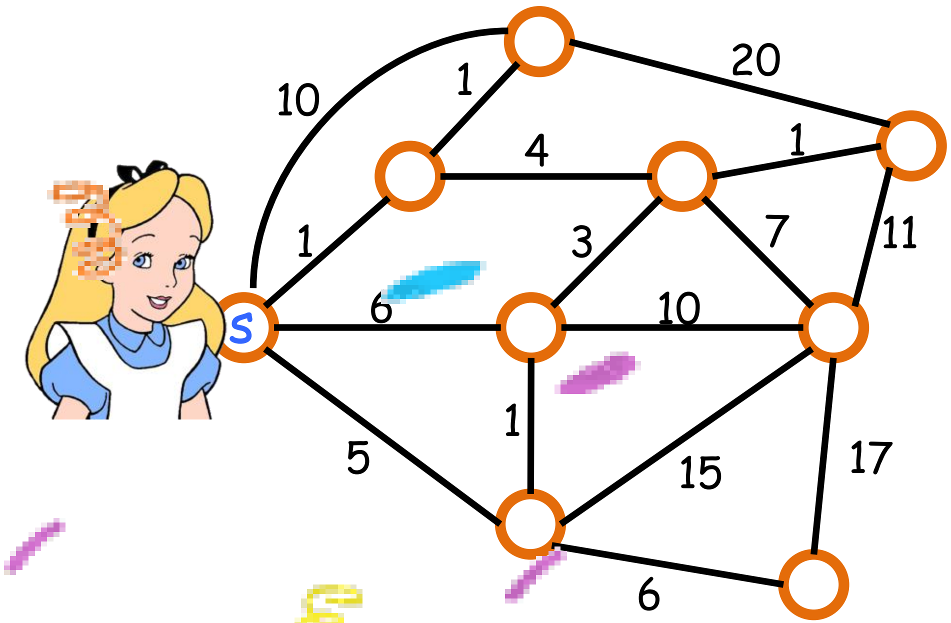
s
10
1
20
1
4
1
3
7
11
6
10
1
5
15
17
6

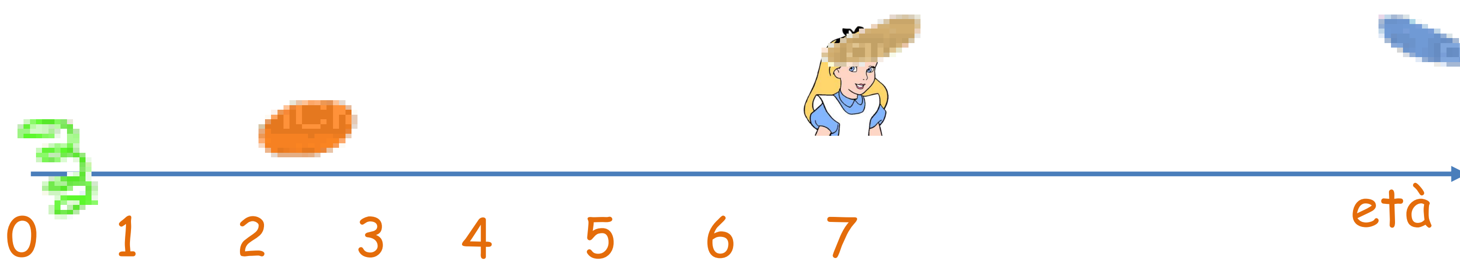


età
0 1 2 3 4 5 6



s
10
1
20
4
1
1
3
7
11
6
10
1
5
15
17
6
0 1 2 3 4 5 6 7
età


s
10
1
20
4
1
1
3
7
11
6
10
1
5
15
17
6


0 1 2 3 4 5 6 7
età

idea: far crescere l'insieme degli archi utilizzabili con l'età

Considera gli archi di G ordinati in ordine crescente di peso:

$$\underbrace{e_1, e_2, e_3, \ldots \quad \ldots \quad e_i}_{E_i} \quad \ldots \quad e_{m-1}, e_m$$

$G_i=(V, E_i=\{e_1, e_2, \ldots, e_i\})$

osservazione: se G_i è connesso allora Alice può vedere il mondo intero all'età di $w(e_i)$.

goal: trovare il minimo i per cui G_i è connesso corretto?

sì: per oss. di prima

complessità?

SearchAge(G)

1. ordina gli archi E di G in ordine crescente di peso e siano essi $e_1,e_2,\dots e_m$. } O(m log m)=O(m log n)
2. **for** i=1 to m
   - costruisci G_i=(V,{$e_1,e_2,\dots e_i$})
   - **if** G_i è connesso **then return** $w(e_i)$

   } O(m+n)
3. **return** "nessuna età"

$\Rightarrow$ O(m(m+n)=O(m^2)

posso fare meglio?

cerco l'indice i usando l'approccio della ricerca binaria!

corretto?

sì

complessità?

SearchAge(G)

1. **if** G non è connesso **then return** “nessuna età”
2. ordina gli archi E di G in ordine crescente di peso e siano essi $e_1,e_2,\ldots e_m$.
3. **return** BinarySearchAge(G,1,m)

assume sempre G_j connesso

BinarySearchAge(G, i, j)

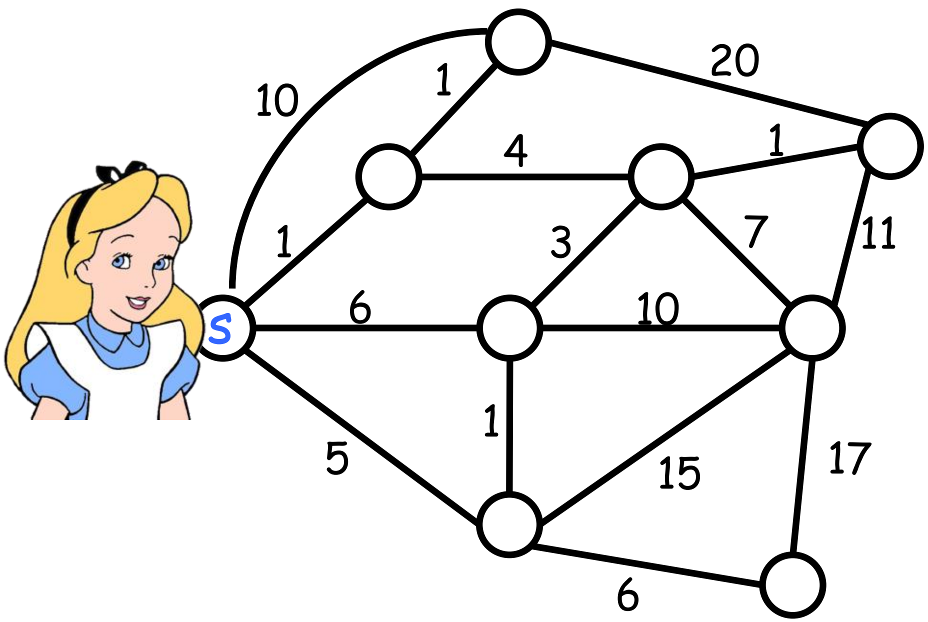
1. **if** (i=j) **then return** $w(e_i)$
2. h=$\lfloor (i+j)/2 \rfloor$
3. costruisci $G_h=(V,\{e_1,e_2,\ldots e_h\})$
4. **if** G_h è connesso
   then return BinarySearch(G, i, h)
   else return BinarySearch(G, h+1, j)

BinarySeachAge guarda
O(log m)=O(log n)
grafi G_h
per ognuno dei quali
spende tempo O(m+n)

⇒ O(m log n)

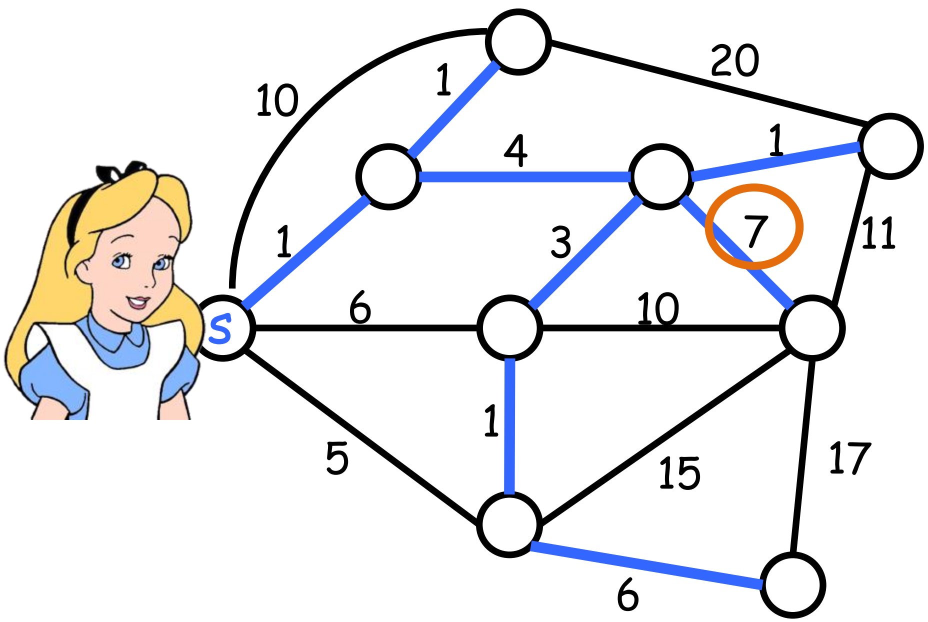
un’altra soluzione

(attenzione: spoiler su ASD modulo 2!)



un'altra soluzione

(attenzione: spoiler su ASD modulo 2!)



calcola il Minimum Spanning Tree T e restituisci il peso dell'arco più pesante di T

Qualche altro esercizio su grafi

Problema (mandare la pedina in buca)

Sia $G = (V, E)$ un grafo non orientato e non pesato, dove ogni arco $e \in E$ ha uno fra tre possibili colori $\{1, 2, 3\}$, e sia $U \subseteq V$ un sottoinsieme di nodi *speciali*. Su un nodo $s \in V$ è posizionata una pedina, inizialmente di colore 1. Il vostro obiettivo è quello di portare la pedina su un nodo *target* $t \in V$. Per far ciò potete usare le seguenti *mosse*:

- `sposta`: se la pedina si trova sul nodo $u \in V$, ha colore i, e c'è un arco $(u, v) \in E$ di colore i, allora potete spostare la pedina da u a v. Il colore della pedina resta i;

- `cambia colore`: se la pedina si trova su un nodo speciale $u \in U$, potete ricolorare la pedina del colore che volete. La pedina in questo caso, dopo la mossa, resta sul nodo u.

Progettate un algoritmo efficiente che trova, se ne esiste una, la sequenza più corta di mosse che risolve l'istanza del gioco.

Problema (una storia romantica di un amore contrastato)

Romeo e Giulietta si amano e vogliono incontrarsi il prima possibile ma l'amore, si sa, è una cosa difficile. Il mondo in cui vivono è modellato come un grafo non orientato e pesato $G = (V, E, w)$ in cui ad ogni arco $e \in E$ è associato un peso $w(e)$. Romeo vive in $s \in V$ mentre Giulietta in $t \in V$. Gli archi del grafo, che possono essere utilizzati per muoversi da un nodo verso un suo vicino, possono essere di tue tipi, e quindi E è partizionato in due insiemi E_A e E_B. Gli archi in E_A sono relativi a tratte che possono essere percorse soltanto in aereo (la compagnia che gestisce i voli è la Montecchi Airlines), mentre gli archi in E_B sono tratte percorribili solo in bus (corse erogate dalla Capuleti Transporti S.p.a.). In ogni caso il peso di un arco rappresenta il tempo di percorrenza della tratta. Ora, Romeo e Giulietta possono muoversi *contemporaneamente* nel grafo e possono incontrarsi in un qualunque nodo. Però, come in tutti gli amori contrastati che si rispettino, c'è un problema: Romeo soffre di mal d'auto e non può prendere i bus, mentre Giulietta ha paura di volare. Che fare? Riuscite ad aiutarli?

Progettate un algoritmo (il più efficiente possibile) che calcoli le strade che i nostri eroi devo seguire per incontrarsi il prima possibile.

Problema Una rete stradale di una città è modellata come un grafo non orientato e non pesato $G = (V, E)$. Dovete portare un pacco ad un cliente che si trova nel nodo t. Il pacco inizialmente è depositato in un magazzino che si trova nel nodo s. Avete a disposizione due droni, posizionati inizialmente nei nodi d_1 e d_2. I droni sono perfettamente uguali. Ogni drone può:

- spostarsi dal nodo in cui si trova verso un nodo adiacente usando un arco del grafo;
- caricare il pacco, se il drone e il pacco si trovano nello stesso nodo;
- depositare il pacco su un certo nodo u, se il drone si trova nel nodo u.

Per questioni di batteria, i droni possono percorrere una distanza di al più k archi, mentre caricare e scaricare il pacco non consuma energia. Si progetti un algoritmo più efficiente possibile che decida se c'è un modo per portare il pacco a destinazione.