Primo Progetto ASD 2023/2024


Algoritmi greedy
a Barbieland

Barbieland

A Barbieland tutto scorre in maniera tranquilla ed efficiente. I suoi abitanti sono tutti felici, e la vita è semplice e priva di controesempi che invalidano soluzioni basate su tecniche greedy. Infatti, ogni abitante è convinto che prendere di volta in volta la decisione più invitante porti sempre a risolvere correttamente tutti i propri problemi.

Un risveglio traumatico

Tuttavia, un giorno Barbie si sveglia con un dilemma: *e se non fosse vero che gli algoritmi greedy portino sempre a una soluzione ottima?*

Questa domanda la porta verso una crisi interiore non facile da gestire.

Barbie Professoressa di Algoritmi

Barbie decide di cercare una risposta e si affida a Barbie Professoressa di Algoritmi, nota per essere l'abitante di Barbieland più anticonformista in fatto di algoritmi.

Barbie Professoressa di Algoritmi le confessa che, in realtà, i problemi risolvibili con tecniche greedy sono solo una piccola parte, e c'è un mare di problemi lì fuori che necessita di soluzioni più sofisticate.

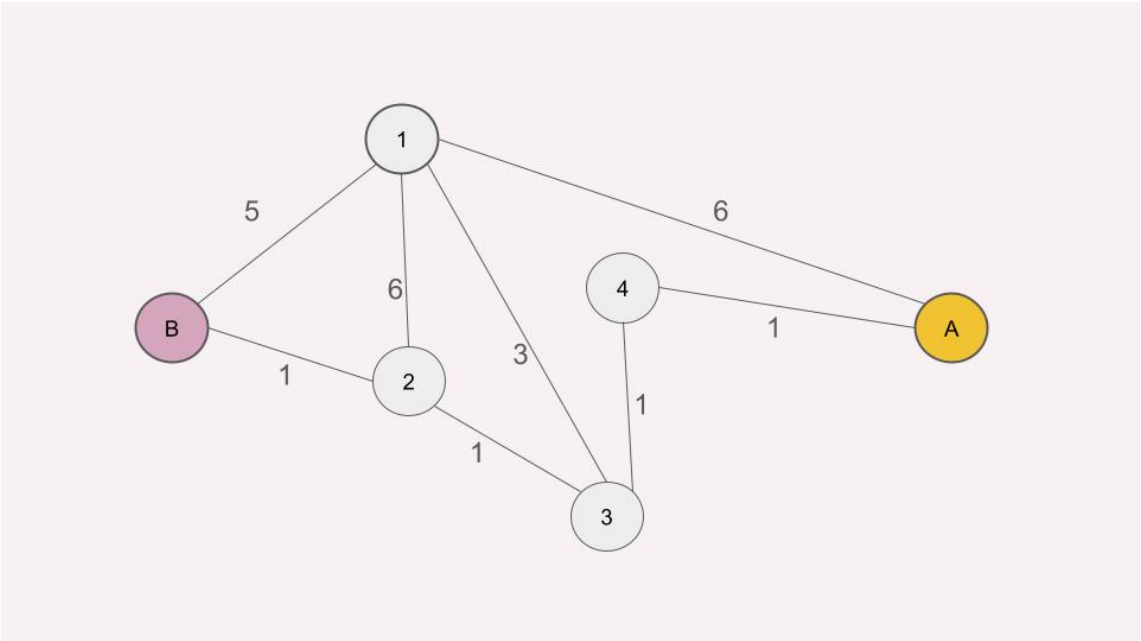
Algoritmia

Dopo questa rivelazione, Barbie Professoressa di Algoritmi invita Barbie ad esercitarsi, per padroneggiare l'arte del problem solving.

Le consiglia quindi di dirigersi verso Algoritmia, un posto pieno di nuovi problemi da scoprire e non limitato dalle tecniche che conosce già.

Il Mondo di Barbie

Il Mondo di Barbie è infatti composto da una serie di città. Ogni coppia di città può essere collegata o meno da una strada con un certo tempo di percorrenza. Da una città si possono raggiungere sempre tutte le altre, potenzialmente attraversando una o più città intermedie. Qui Barbieland è indicata con *B*, Algoritmia con *A* e il resto delle città con un numero identificativo. I tempi di percorrenza sono interi positivi.



GraphCruise

Dato il viaggio da affrontare, Barbie Professoressa di Algoritmi presta a Barbie la sua auto personale: la GraphCruise. Questa automobile ha una particolarità: una volta selezionata la città di partenza e la città di destinazione, ci arriverà seguendo sempre un percorso di costo minimo tra tutti i percorsi possibili.

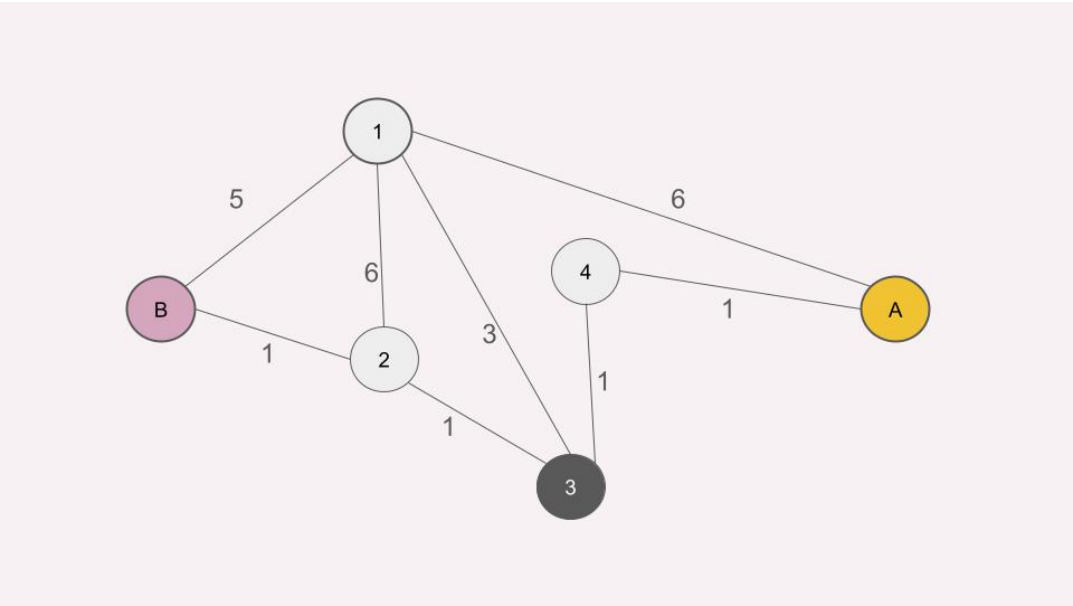
Inoltre, regala a Barbie un libro da cui studiare nuove tecniche algoritmiche.

Nemici in agguato

Tutto sembra perfetto. Tuttavia, il famigerato gruppo nemico degli algoritmi efficienti, la *Miseri Algoritmi Tremendamente Tediosi E Lenti*, ha ricevuto una soffiata ed è pronta a sventare il piano di Barbie.

Nemici in agguato

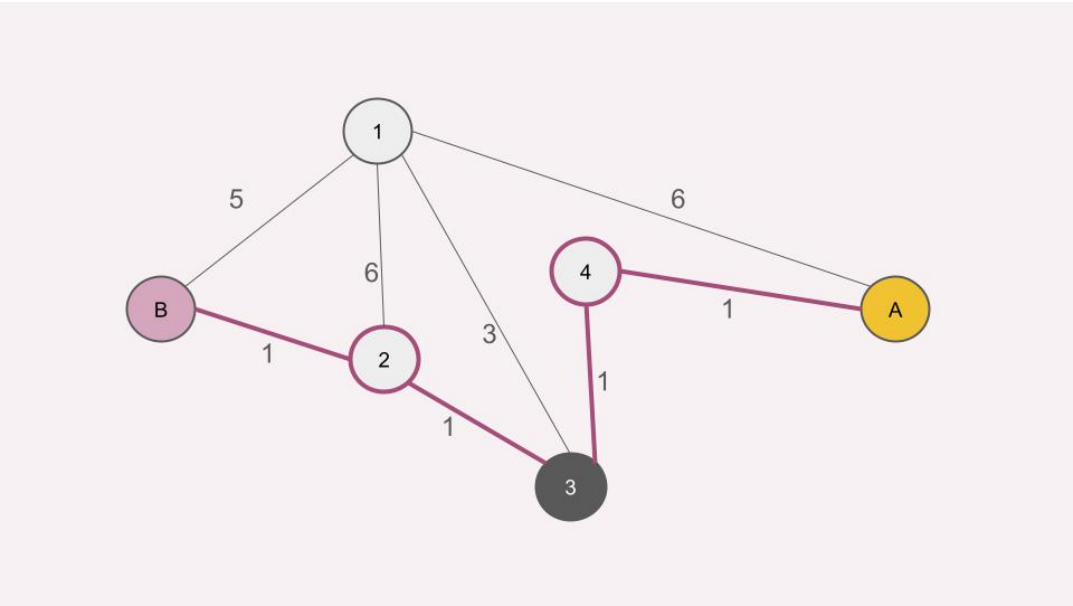
La M.A.T.T.E.L. ha inviato alcuni scagnozzi in giro per le varie città. Se la GraphCruise passasse da una città occupata, verrebbe fermata immediatamente.



UNA QUESTIONE DA RISOLVERE

Ovviamente, la GraphCruise ha le sue regole, che seguirà alla perfezione: la macchina sceglierà uno tra i percorsi più brevi considerando tutta la mappa.

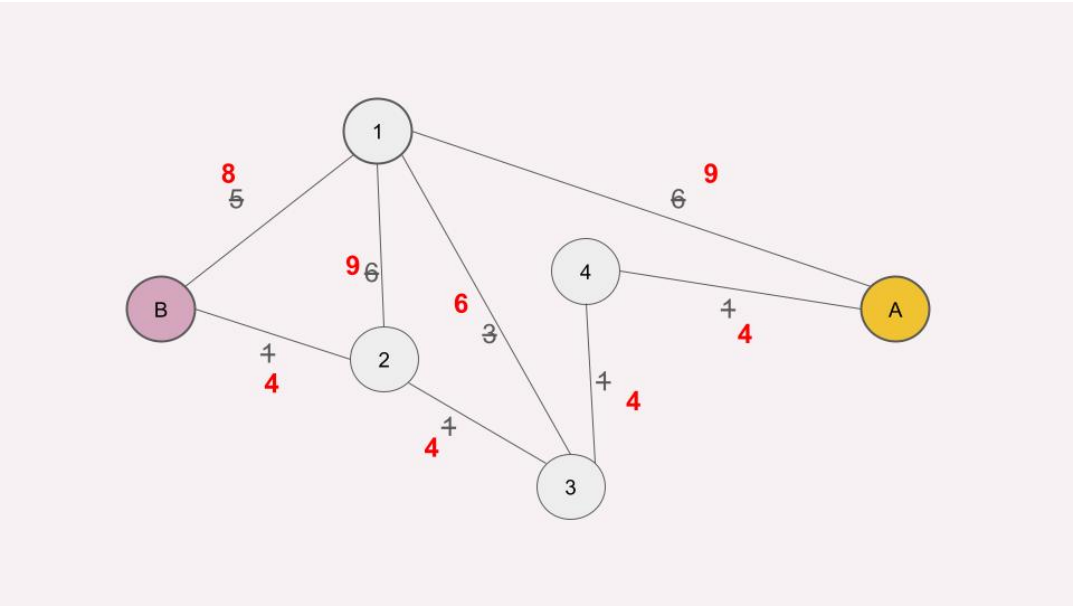
Potrebbe decidere quindi autonomamente di dover passare su una città occupata, nel caso in cui esistesse un percorso di costo minimo che lo prevede. Qui il percorso minimo è segnato in rosa, e passerebbe per una città occupata.



Un potere

Data la guida autonoma della GraphCruise, Barbie passerà il viaggio studiando dal libro ricevuto. Ovviamente, più tempo studierà, più sarà preparata all'arrivo.

Fortunatamente, Barbie Professoressa di Algoritmi ha un potere segreto: è in grado di allungare a piacimento tutte le strade del Mondo di Barbie in un colpo solo, ma tutti gli incrementi dei tempi di percorrenza delle strade saranno dello stesso valore K.



E ora? Il vostro compito

Barbie Professoressa di Algoritmi cercherà di usare il suo potere in aiuto di Barbie, permettendole di passare più tempo possibile studiando in macchina. Aiutatela nel suo lavoro.

Qual è il massimo valore di K che Barbie Professoressa di Algoritmi può scegliere in modo che il percorso scelto dalla GraphCruise debba passare necessariamente per città non occupate? Inoltre, qual è un percorso di costo minimo quando la Professoressa decide di non usare il suo potere?

Oppure, detto in altri termini, qual è il massimo valore di K per cui tra i percorsi minimi tra Barbieland e Algoritmia non c'è nessun percorso che preveda di passare per una città occupata? Qual è un percorso di costo minimo quando $K = 0$?

Input

Un file con $1 + S + 1 + M$ righe.

- La prima riga riporta 2 numeri interi: C (`int`) e S (`int`), rispettivamente il numero di città del Mondo di Barbie e il numero di strade.
- Le successive S righe descrivono le strade. Ogni riga riporta 3 interi a (`int`), b (`int`) e w (`int`), rappresentanti gli identificativi delle due città collegate e il tempo di percorrenza della strada.
- La riga successiva riporta un intero M (`int`): il numero di nodi occupati.
- Le successive M righe riportano ciascuna un intero o (`int`): l'identificativo di una città occupata.

Output (parte obbligatoria)

1. La prima riga del file di output deve riportare il valore di K (`int`) individuato:
   - Se $K \in \mathbb{N}^+$, riportare K.
   - Se K è illimitato, riportare -1.
   - Se è impossibile individuare un valore $K > 0$, riportare -2.

Output (parte facoltativa)

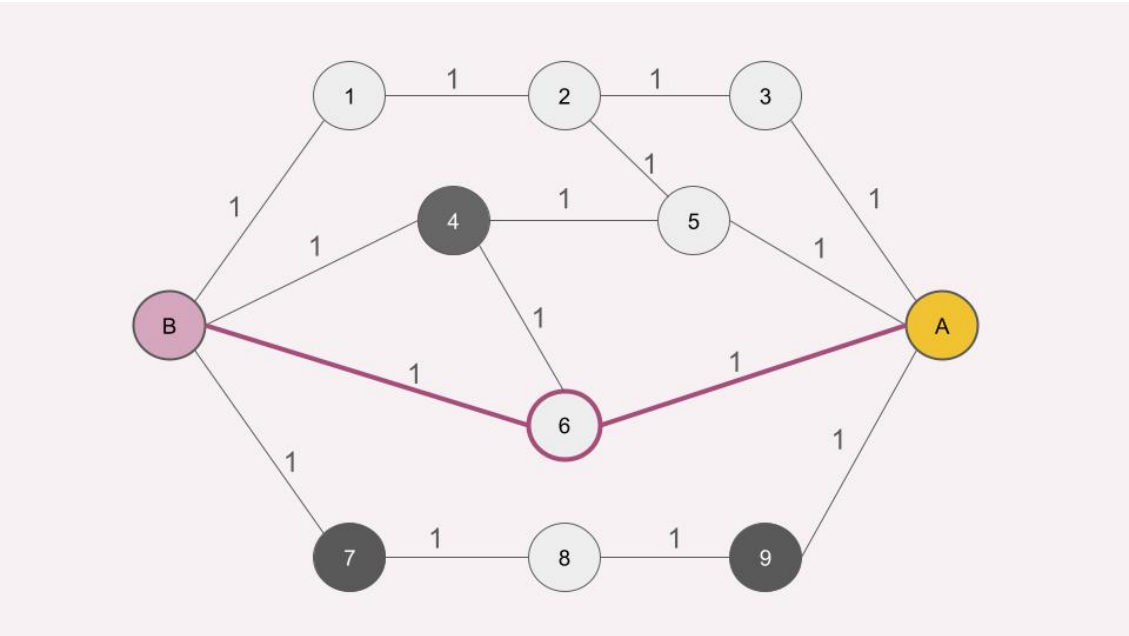
Opzionalmente, l'output può contenere anche le seguenti due righe:

2. La seconda riga contiene un intero R: il numero di città che Barbie visiterebbe se $K = 0$ per raggiungere Algoritmia partendo da Barbieland (comprese B e A) seguendo uno qualsiasi tra tutti i percorsi minimi (non importa se si attraversano città occupate);
3. La terza riga contiene R interi: le città che sarebbero visitate secondo questo percorso, in ordine, da Barbie (comprese B e A), se $K = 0$.

La soluzione parziale vale **3 punti**, la soluzione completa **5 punti.**

Esempio - Tutte strade con costo 1

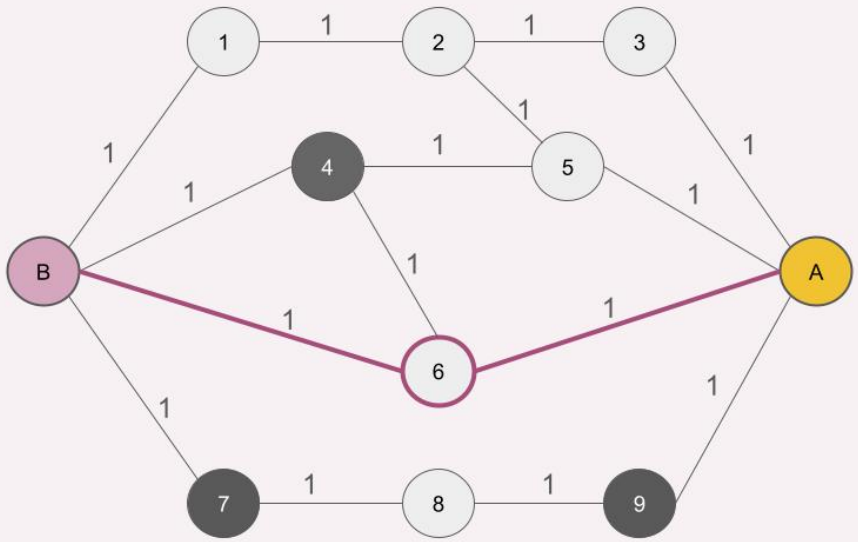
- 11 città e 15 strade
- Le strade hanno tutte tempo di percorrenza 1
- I nodi grigi sono quelli occupati, Barbieland è il nodo rosa, Algoritmia è il nodo giallo
- Il percorso più breve (2) è quello segnato in rosa, è unico e non passa per nodi grigi.
- Qualsiasi valore di K non cambierebbe il percorso minimo: K è illimitato (output -1).



ESEMPIO - OUTPUT

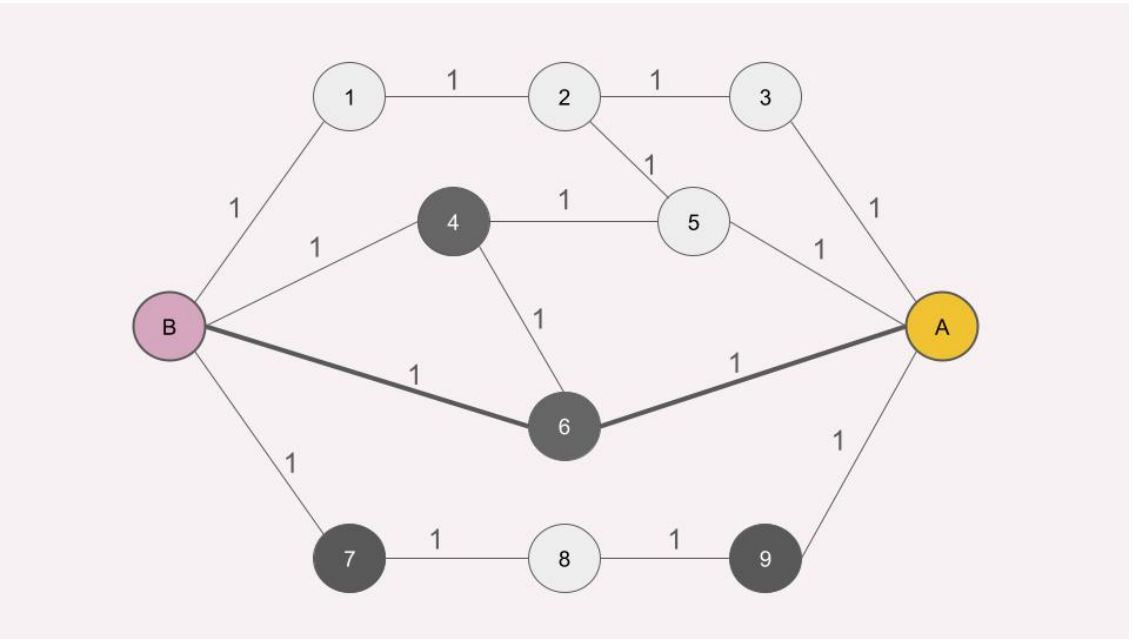
Nell'esempio precedente, l'output valido sarebbe il seguente:

```
-1
3
0  6  10
```



ESEMPIO - K NON ESISTENTE

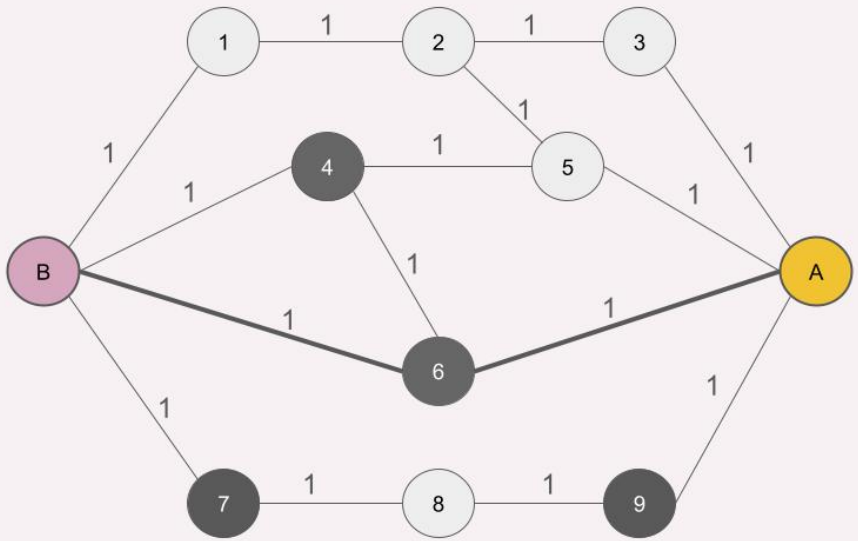
- Grafo uguale a prima, ma il nodo 6 ora è occupato.
- Non ci sono altri percorsi minimi rispetto a B–6–A.
- Qualsiasi valore di K non cambierebbe il fatto che B–6–A sia il percorso minimo: output -2.



ESEMPIO - OUTPUT

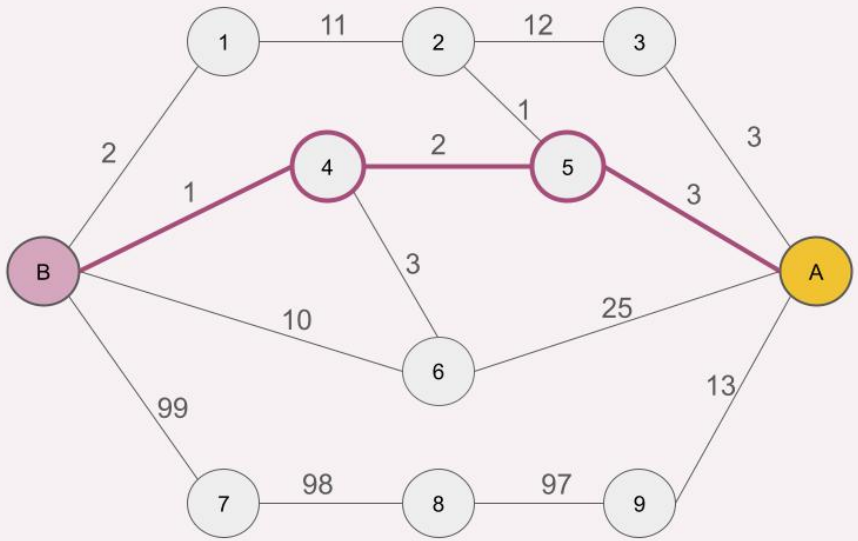
Nell'esempio precedente, l'output valido sarebbe il seguente:

```
-2
3
0  6  10
```



Esempio - Tutte città libere

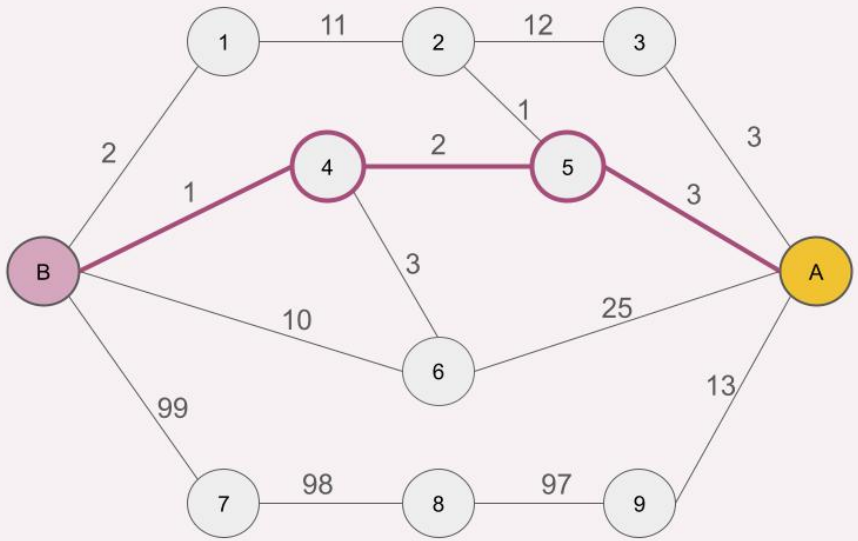
- In questo caso non ci sono città occupate.
- Ovviamente qualsiasi valore di K va bene, quindi K è illimitato (output -1).
- Il percorso minimo quando $K = 0$ è B–4–5–A di costo 6.



ESEMPIO - OUTPUT

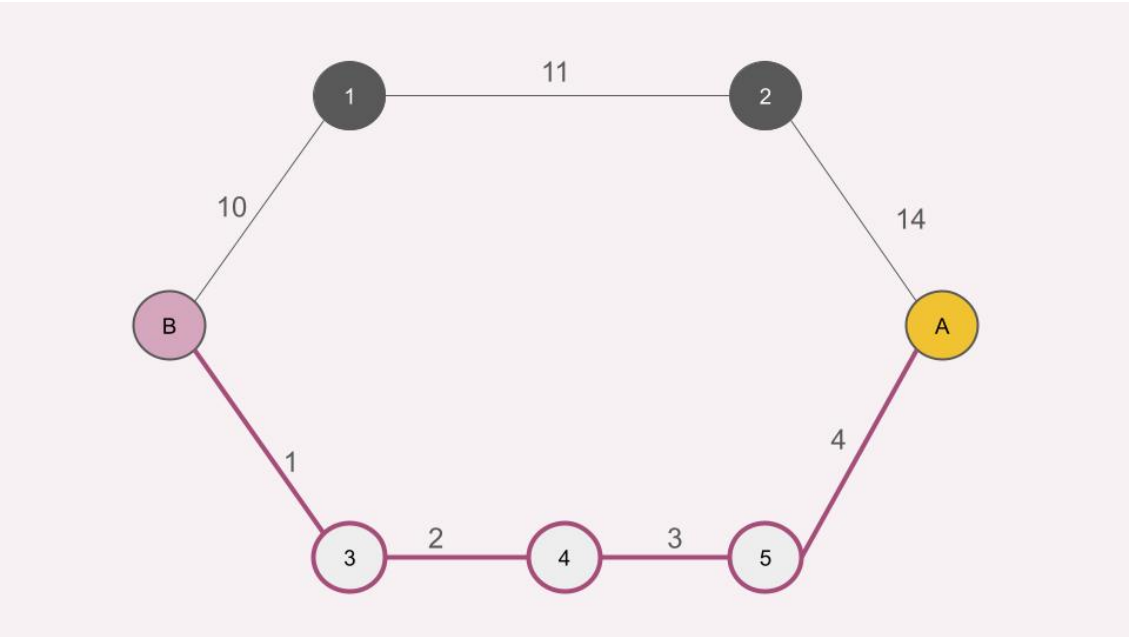
Nell'esempio precedente, l'output valido sarebbe il seguente:

```
-1
4
0 4 5 10
```



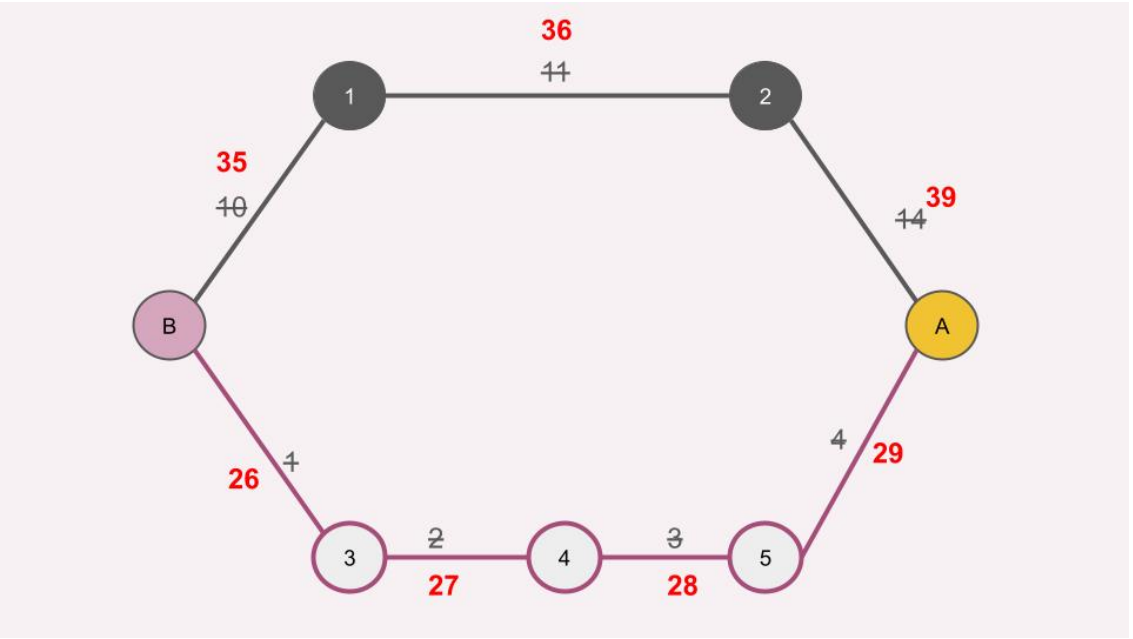
Esempio - Percorsi disgiunti

- In questo caso il percorso minimo ha costo 10 e prevede di passare solo per nodi liberi.
- Esiste un altro percorso per andare da *B* ad *A* che ha costo 35, e prevede di passare per nodi occupati.



Esempio - Percorsi disgiunti

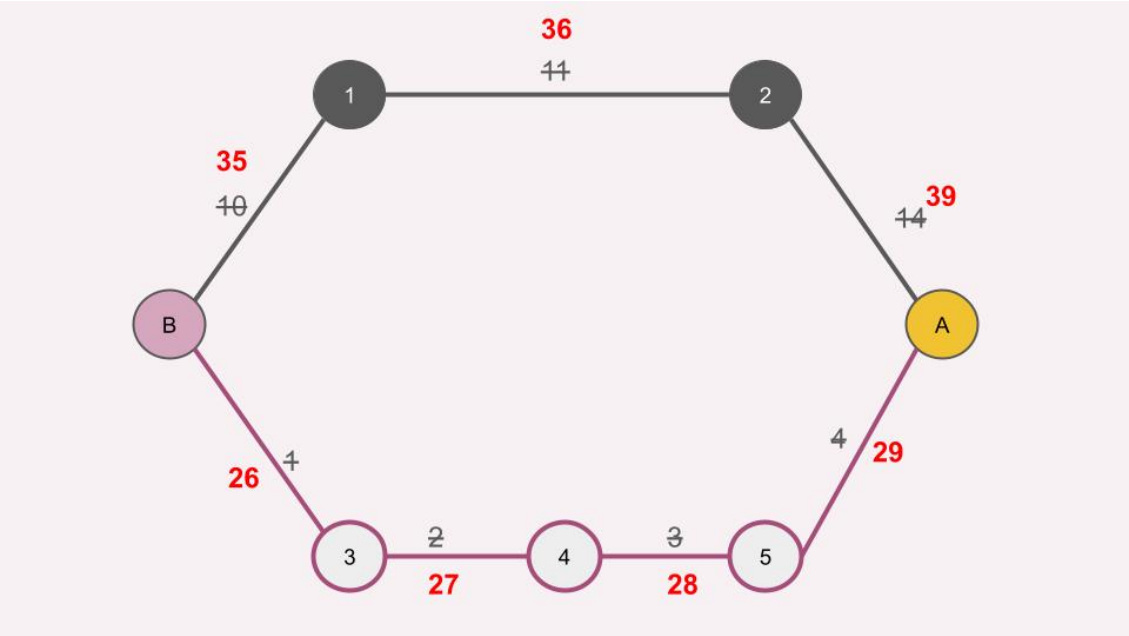
- Selezionamo $K = 25$ e aggiorniamo tutti i pesi (rosso).
- A questo punto i due percorsi sono entrambi di costo minimo 110. Quindi Barbie rischierebbe di prendere un percorso minimo che passa per una città occupata.
- $K = 24$ risulta il massimo valore di K per cui è garantito che Barbie passi solo per nodi non occupati.



ESEMPIO - OUTPUT

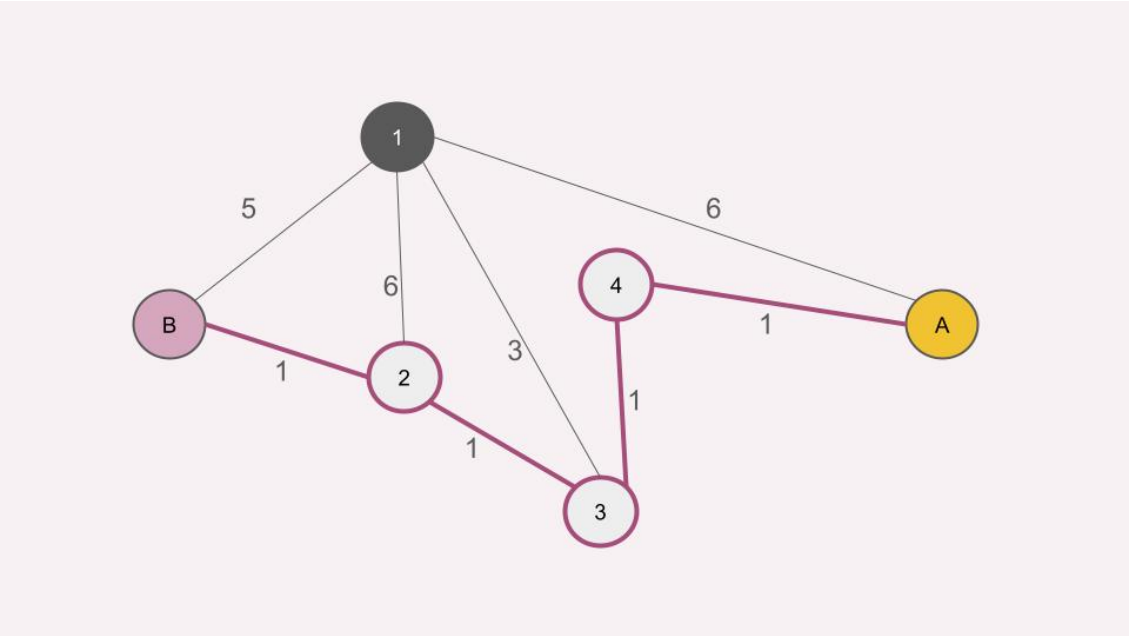
Nell'esempio precedente, l'output valido sarebbe il seguente:

```
2 4
5
0 3 4 5 6
```



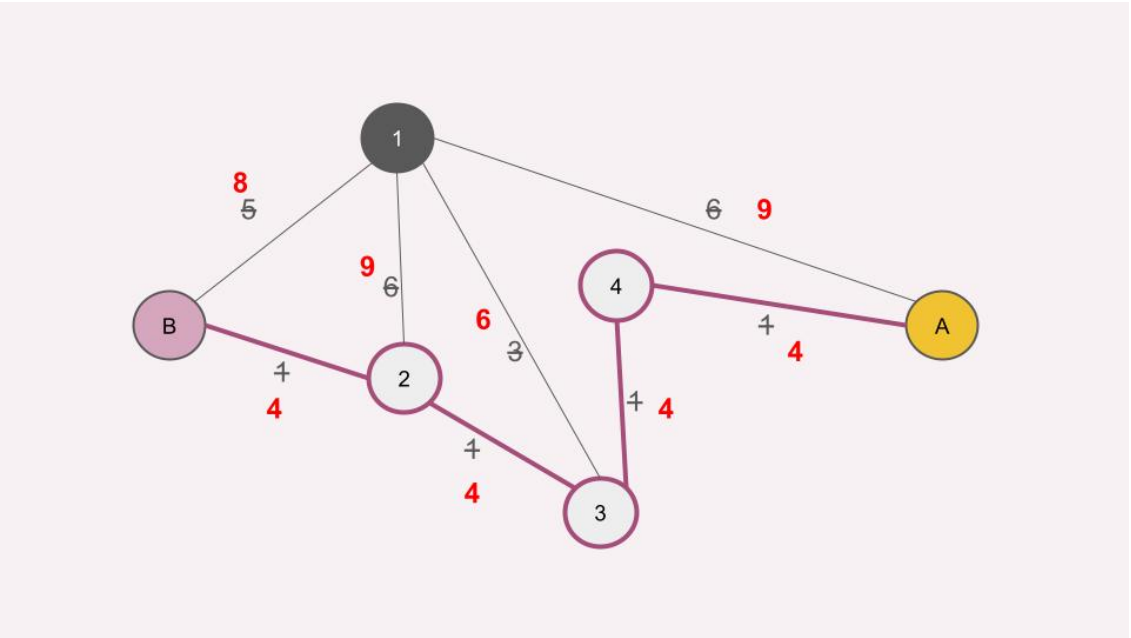
Esempio - Generale

- Qui proponiamo un esempio più generale
- Con $K = 0$ abbiamo che il percorso minimo è unico e passa per città non occupate.
- Il percorso è *B*–2–3–4–*A* ed ha costo 4.
- Esistono comunque altri percorsi per andare da *B* ad *A*, ma non di peso minimo ed eventualmente passando per città occupate.



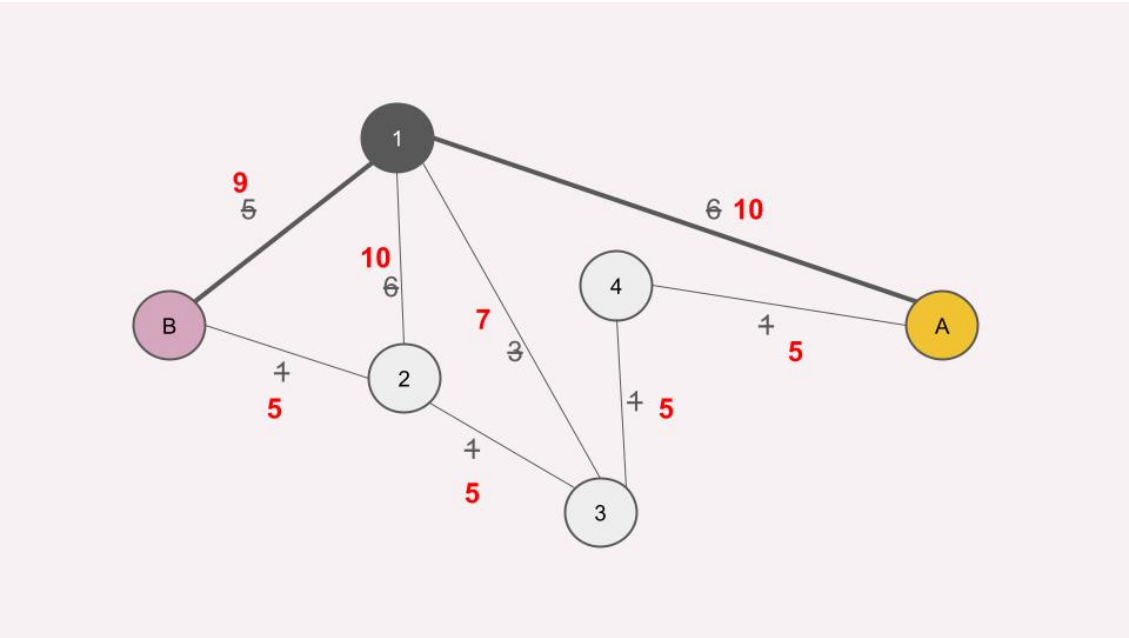
Esempio - Generale

- Proviamo a scegliere $K = 3$ ed aggiorniamo i pesi (rosso).
- Il percorso minimo rimane B–2–3–4–A, ma ora ha costo 16.



Esempio - Generale

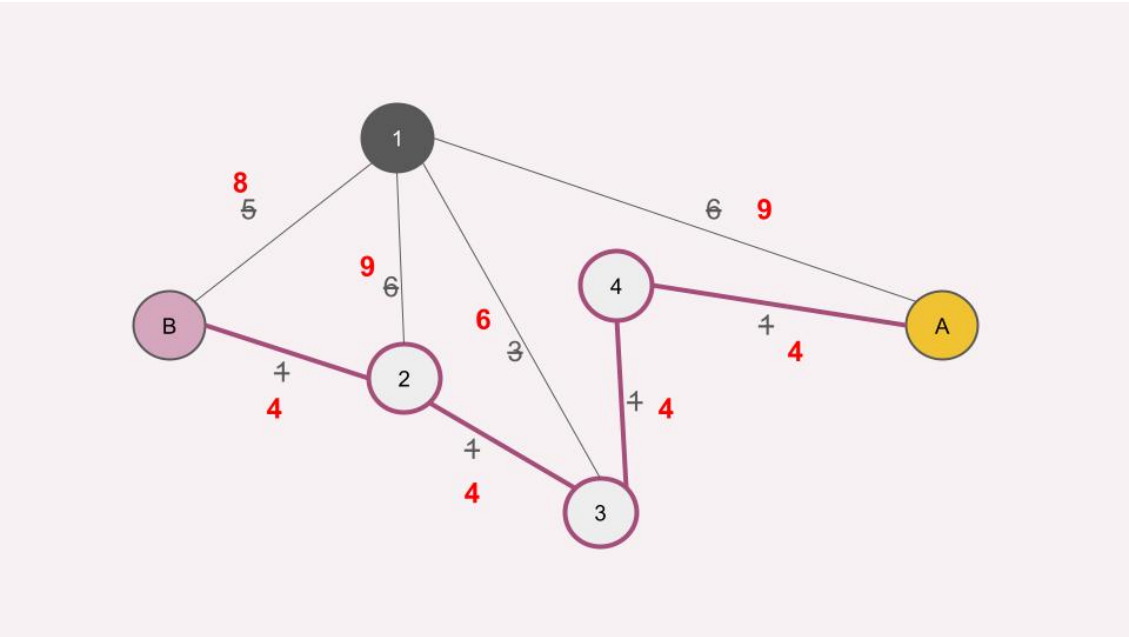
- Proviamo a scegliere $K = 4$ ed aggiorniamo i pesi (rosso).
- A questo punto il percorso minimo è B–1–A con costo 19. Tuttavia bisogna notare che questo percorso attraversa una città occupata. Non è quindi un valore di K valido.
- Il percorso B–2–3–4–A ora ha costo 20.



ESEMPIO - OUTPUT

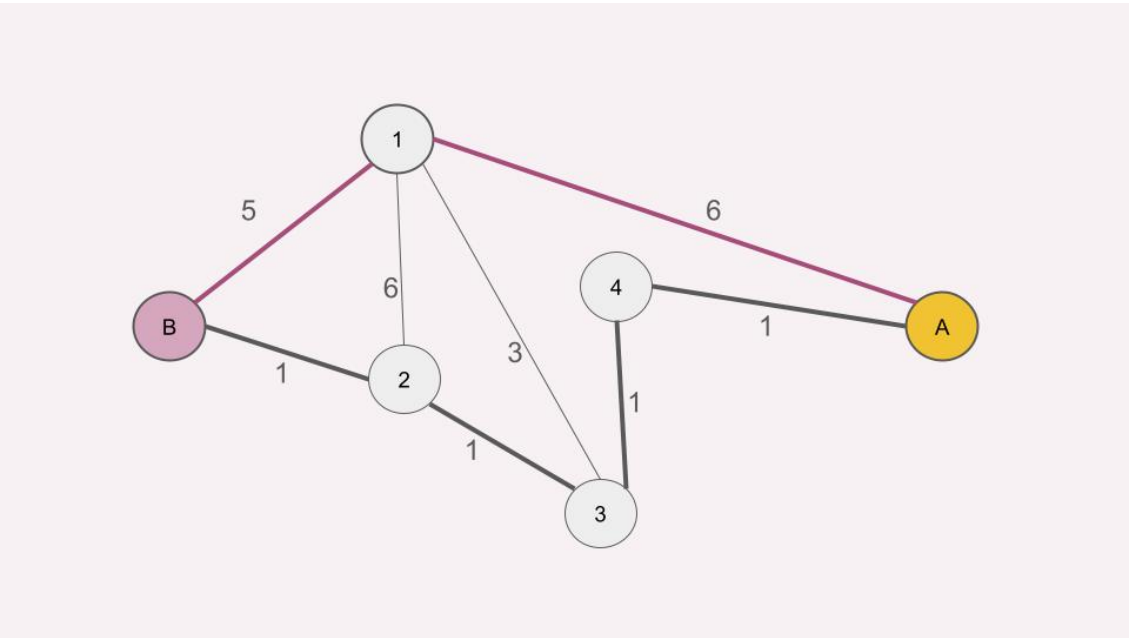
Nell'esempio precedente, l'output valido sarebbe il seguente:

```
3
5
0 2 3 4 5
```



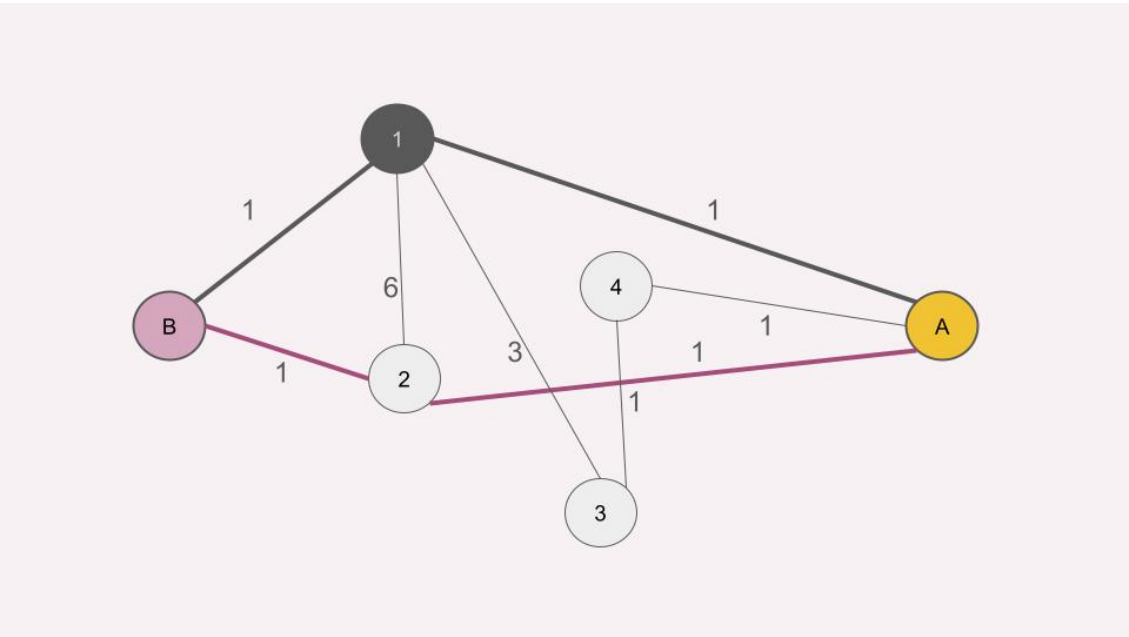
Esempio - altri possibili errori (I)

- Qui abbiamo un esempio in cui K è illimitato. Bisogna trovare il percorso minimo quando $K = 0$.
- Supponiamo che abbiate dato in output il percorso B–1–A, evidenziato in rosa, di costo 11.
- Esiste però il percorso evidenziato in nero, che costa 4. Il vostro output è errato, in quanto percorso non minimo.



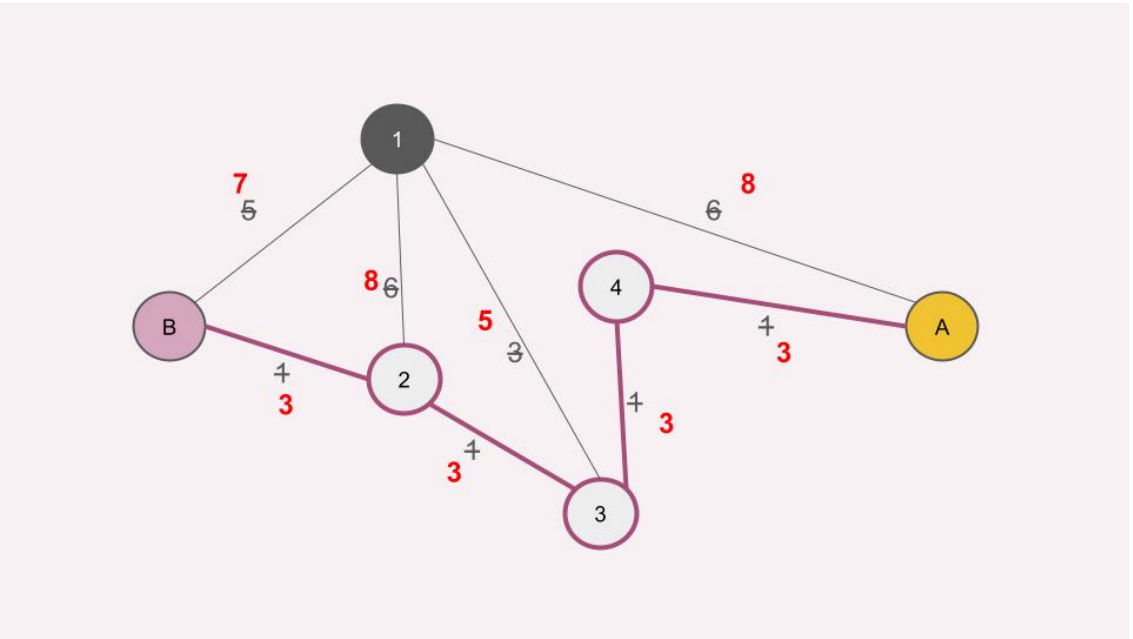
Esempio - altri possibili errori (II)

- Supponiamo qui abbiate dato come percorso minimo *B*–2–*A* e *K* illimitato.
- Il percorso è corretto, va bene uno tra *B*–2–*A* e *B*–1–*A*, tuttavia trovare *K* è impossibile. Per qualsiasi valore di *K*, entrambi i percorsi manterrebbero lo stesso costo, ergo esisterebbe sempre un percorso di costo minimo con una città occupata.



Esempio - altri possibili errori (III)

- Esempio che abbiamo già visto.
- Supponiamo abbiate identificato $K = 2$; con questo valore di K il percorso minimo non passa per alcuna città occupata. Tuttavia K non è massimale, infatti anche per $K = 3$ il percorso di costo minimo non passa per città occupate.



Note su Input

Assunzioni generali

- $2 \leq C \leq 1.000$
- Barbieland è al nodo 0, Algoritmia è al nodo $C - 1$
- Ogni altra città è identificata da un numero tra 1 e $C - 2$
- $0 \leq M \leq C - 2$
- Barbieland e Algoritmia sono sempre NON occupate
- $0 \leq S \leq 10.000$
- Per il tempo di percorrenza w_i dell'i-esima strada vale $1 \leq w_i \leq 100.000$
- Ogni grafo è indiretto.
- Ogni grafo è connesso.
- Non ci sono self-loop, cioè archi (a, a) dove a è una città
- Non ci sono archi doppi, ovvero esiste al massimo una tripla (a, b, w) per una data coppia non ordinata $\{a, b\}$

Casi di test

Ci sono 20 casi di test in totale:

- ★☆☆☆ In 6 casi su 20, tutti gli archi hanno lo stesso tempo di percorrenza;
- ★★☆☆ In 4 casi su 20, o nessuna città è occupata, o tutte le città sono occupate (*B* ed *A* escluse);
- ★★★☆ In 4 casi su 20, i percorsi da *B* ad *A* sono tutti disgiunti, ovvero non condividono città intermedie;
- ★★★★ Nel restante dei casi non ci sono particolari limitazioni.

Per la sufficienza si possono risolvere i 6 casi di difficoltà ★☆☆☆, (includendo la parte facoltativa).

Limiti di tempo e memoria

I limiti di tempo e memoria sono:

- Tempo limite massimo: 1 secondo.
- Memoria massima: 16 MB.

$\Rightarrow$ Limite di 50 sottoposizioni per gruppo.

$\Rightarrow$ Potete provare con un dataset equivalente sulla vostra macchina (sito: https://judge.science.unitn.it/slides/).

Nota: Il dataset di esempio contiene in output solo K, senza ulteriori informazioni.

Punteggio

Ogni caso di test vale 5 punti se completo, 3 punti se parziale. Il punteggio massimo è di 100 punti.

⇒ La **sufficienza è posta a 30 punti**.

- ✗ se K non è corretto, si ottengono 0 punti.
- ✗ se K è corretto, ma il percorso stampato non è possibile, si ottengono 0 punti.
- ✗ se K è corretto, ma il percorso stampato non è minimo, si ottengono 0 punti.

Punti bonus per l'esame

L'assegnazione punti avviene in maniera competitiva:

- **3 punti** ai gruppi nel primo terzile della classica (primo terzo della classifica);
- **2 punti** ai gruppi nel secondo terzile della classifica (secondo terzo della classifica);
- **1 punto** ai gruppi nel terzo terzile della classifica (ultimo terzo della classifica).

Vengono considerati nella classifica per l'assegnazione dei punti solamente i **gruppi che raggiungono la sufficienza** (punteggio maggiore o uguale a 30).

$\Rightarrow$ Classifica:
https://judge.science.unitn.it/arena/ranking/

Consegna

Consegna: lunedì 18 dicembre 2023 ore 18:00

Per caricare il vostro codice, recatevi su
`https://judge.science.unitn.it/arena/`

Ricordiamo che nel calcolo del punteggio verrà considerata **l'ultima** soluzione consegnata.

Suggerimenti

Cominciate subito a lavorare al progetto per presentarvi al prossimo ricevimento (giovedì 15 dicembre) con tutte le domande che vorrete fare.
In ogni caso, sappiate che:

- potete venire a ricevimento
- rispondiamo su Telegram
- risponderemo alle vostre mail

Do's

È permesso:

- Discutere all'interno del gruppo
- Chiedere chiarimenti sul testo
- Chiedere opinioni su soluzioni
- Sfruttare codice fornito nei laboratori
- Utilizzare pseudocodice da libri o Wikipedia
- Richiedere aiuto (anche pesante) per la soluzione "minima"
- Venire a ricevimento
- Mandare meme (algoritmici) sul nostro canale Telegram: it's your time to shine!

Don'ts

È vietato:

- Discutere con altri gruppi
- Mettere il proprio codice su repository pubblici
- Utilizzare codice scritto da altri
- Condividere codice (abbiamo potenti mezzi!)
- Chiedere suggerimenti online (es: stackoverflow)
- Fare riferimenti a soluzioni del progetto nei meme

Ricevimento

Date e orari

- martedì 12 dicembre 2023 dalle 10:30 alle 12:30 (A101);
- giovedì 14 dicembre 2023 dalle 13:30 alle 15:30 (A101);

⇒ Negli orari di ricevimento saremo a disposizione sempre online, quando avrete bisogno di un aiuto scrivetelo sul gruppo telegram e risponderemo in chat privata o tramite una chiamata zoom.

⇒ Per qualsiasi domanda mandateci una mail a: `asd.disi@unitn.it` oppure contattateci su Telegram.