Algoritmi e Strutture Dati

AmongASD (`amogasd`)

Testo del problema

Slides originali su: judge.science.unitn.it/slides/asd22/prog1.pdf
Il professor Montresus, docente del corso di ASD, è convinto che il modo migliore per insegnare agli studenti gli algoritmi su grafi sia l'apprendimento sul campo e non lo studio sui libri. L'università non è altro che un grafo dove varie stanze sono collegate tra loro da corridoi che possono essere attraversate in un certo numero di unità di tempo. Per motivi logistici, i corridoi possono essere percorsi solo in una direzione.
È quasi ora della sessione invernale e gli studenti di ASD sono alla disperata ricerca di un modo per passare l'esame. Delle voci di corridoio dicono che in una remota stanza di Povo - il FabLab[1] - ci sia nascosta la Bibbia^{TM2} che li può aiutare a superare l'esame.
Ma un **impostore** si mescola tra gli studenti! Preoccupato che tutti possano passare l'esame troppo facilmente, il prof. Montresus cerca di arrivare per primo al FabLab. Il professor Montresus ha un asso nella manica. Può attivare delle gigantesche ventole che rendono più veloce o più lento il passaggio attraverso alcuni corridoi tramite il suo centro di comando.

Domanda

Chi arriverà per primo al FabLab, il prof. Montresus o gli studenti? O arriveranno insieme?

- Quanto saranno lunghi il percorso del prof. Montresus e quello degli studenti?
- Come verranno accesi i ventilatori?
- Quale sarà il percorso migliore per l'impostore?

Esempi

Un esempio di mappa dell'università è riportata in Figura 1. In questo caso ci sono: 9 stanze e 11 corridoi, di cui 8 semplici e 3 con ventole. I corridoi semplici hanno un tempo di percorrenza fissato, mentre i corridoi con ventole hanno un range di tempi di percorrenza possibili. Il professor Montresus si trova inizialmente nella stanza al nodo 3, gli studenti al nodo 6 ed il FabLab è al nodo 0.

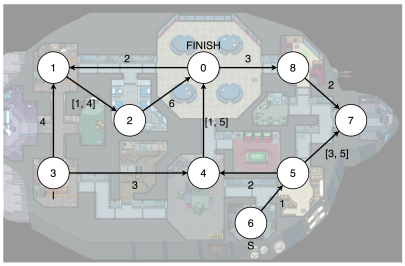


Figura 1: Mappa di esempio: l'impostore (I) parte dal nodo 3, gli studenti (S) partono dal nodo 6 e devono raggiungere il FabLab (FINISH) nel nodo 0.

[1] https://fablab.unitn.it/
[2] Bertossi, Alan, e Alberto Montresor. *"Algoritmi e strutture di dati"*. CittàStudi; 3ª edizione (21 marzo 2014). ISBN-10: 8825173954. ISBN-13: 978-8825173956.

Esempio 1: pareggio (0)

Nell'esempio riportato in Figura 2, gli studenti possono al massimo pareggiare. Il loro percorso più breve (ma anche l'unico) per arrivare da S ad F è lungo 4: gli studenti attraversano 2 corridoi semplici impiegando 3 unità di tempo e un corridoio con ventola che è stato manipolato per essere percorribile in 1 unità di tempo.
Anche il prof. Montresus non può far meglio che raggiungere F in 4 unità di tempo (percorso: $[3, 4, 0]$), pareggiando. Montresus ha anche un percorso alternativo per arrivare ad F: 3, 1, 2, 0. Tuttavia, riesce a percorrerlo in 11 unità di tempo (notare che attraversa un corridoio manipolato in modo da essere percorribile il più velocemente possibile).

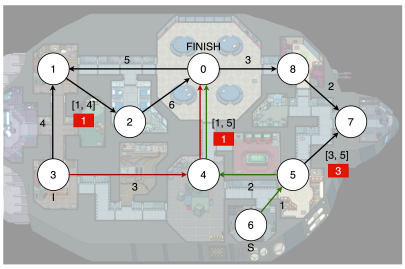


Figura 2: Un caso di pareggio: l'impostore (I) e gli studenti (S) raggiungono il FabLab (F) nello stesso momento in 4 unità di tempo.

Esempio 2: il professor Montresus vince (1)

Nell'esempio riportato in Figura 3, gli studenti perdono sempre. Il loro percorso più breve, che è anche l'unico, per arrivare da S ad F è lungo 8. Gli studenti attraversano 2 corridoi semplici impiegando 3 unità di tempo e un corridoio con ventola che è stato manipolato per essere percorribile in 5 unità di tempo.
L'impostore manipola i percorsi a proprio favore, così facendo può sfruttare il percorso 3, 1, 2, 0 per arrivare ad F in 7 unità di tempo, arrivando quindi prima degli studenti.

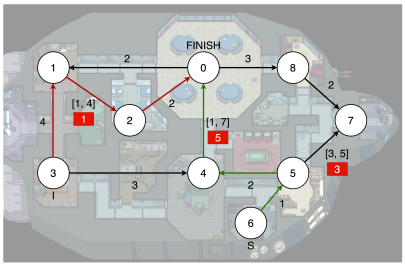
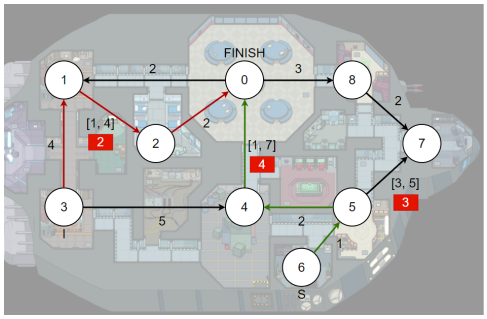
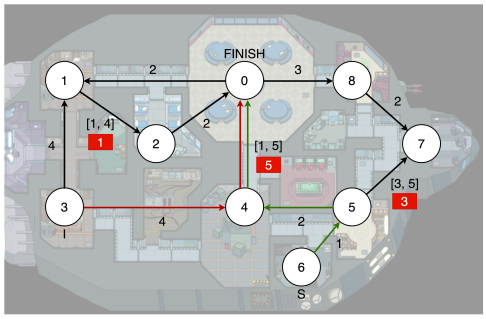


Figura 3: Un caso in cui gli studenti perdono sempre: manipolando le ventole l'impostore fa in modo che gli studenti (S) raggiungano il FabLab (F) in 8 unità di tempo, mentre lui (I) lo raggiunge in 7 unità di tempo.

La Figura 4 riporta due casi di soluzioni **errate**. In Figura 4a, l'impostore manipola le ventole in modo errato finendo per svantaggiarsi e perdere. La ventola nel corridoio $(1, 2)$ è impostata a 2, mentre la ventola nel corridoio $(4, 5)$ è impostata a 4. In questo modo, il percorso più breve per l'impostore è $[3, 1, 2, 0]$ ed è lungo 8. Gli studenti però arrivano a destinazione in 7 unità di tempo, battendo l'impostore. In Figura 4b, con le ventole nei corridoi $(1, 2)$ e $(4, 0)$ impostate rispettivamente a 1 e 5. L'impostore sceglie il percorso $[3, 4, 0]$ lungo 9 che non è il percorso più breve che ha a disposizione, perdendo anche in questo caso.



(a) manipolazione ventole errata



(b) percorso non ottimale

Figura 4: Esempi di soluzioni errate.

Esempio 3: gli studenti vincono (2)

Nell'esempio riportato in Figura 5, gli studenti, nonostante debbano percorrere un corridoio manipolato, hanno a disposizione un percorso percorribile in 8 unità di tempo. Invece, il percorso migliore che Montresus possa percorrere per arrivare in F impiega 9 unità di tempo.
Ovvero gli studenti arrivano al Fablab sempre più velocemente del prof. Montresus.

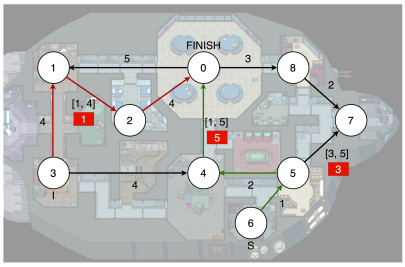


Figura 5: Un caso in cui gli studenti vincono: anche con le manipolazioni a lui più favorevoli, l'impostore (I) raggiunge il FabLab (F) in 9 unità di tempo, mentre gli studenti (S) ne impiegano solo 8.

Input/Output

Input: Un file con $2 + M + K$ righe.

- La prima riga riporta 3 numeri interi: N (`int`), M (`int`) e K (`int`), rispettivamente il numero di stanze, corridoi semplici (non modificabili) e corridoi con ventole.
- La seconda riga riporta 3 numeri interi: I (`int`), S (`int`) e F (`int`), rispettivamente il nodo di partenza dell'impostore, degli studenti e il nodo dove si trova il FabLab.
- Le successive M righe riportano i corridoi semplici che collegano le stanze. Ogni riga riporta 3 interi U (`int`), V (`int`) e T (`int`), rappresentanti la stanza di partenza, la stanza di arrivo e il tempo di percorrenza del corridoio.
- Le successive K righe riportano i corridoi con ventole, e sono costituite da 4 interi ciascuna: U_v (`int`), V_v (`int`), T_{min} (`int`), T_{max} (`int`), che rispettivamente sono la stanza di partenza, la stanza di arrivo, e i tempi di percorrenza minimo e massimo che è possibile ottenere regolando la potenza della ventola.

Output: le informazioni da stampare in output constano di una parte obbligatoria e di una facoltativa.

Parte obbligatoria. Questi elementi devono sempre essere presenti nell'output:

1. La prima riga del file di output deve essere:
   - 1, se è possibile per l'impostore arrivare **prima** degli studenti al FabLab, regolando la potenza delle ventole in modo opportuno;
   - 2, se l'impostore arriva sempre **dopo** gli studenti, in qualsiasi modo regoli la potenza delle ventole;
   - 0, se l'impostore e gli studenti arrivano al FabLab **nello stesso istante**.
2. La seconda riga contiene due interi: le distanze minime con cui I ed S raggiungono il FabLab;
3. La terza riga contiene K interi: i tempi di percorrenza dei corridoi con ventole dopo l'opportuna regolazione della potenza, stampati nello stesso ordine in cui compaiono nell'input. Se $K = 0$, lasciare una riga vuota;

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

4. La quarta riga contiene un intero R: il numero di stanze visitate dall'impostore per arrivare al FabLab (comprese I e F);
5. La quinta riga contiene R interi: le stanze visitate, in ordine, dall'impostore (comprese I e F).

Note:

- L'ordine con il quale stampate i tempi di percorrenza dei K corridoi a percorrenza variabile è importante: dovete stamparli nello stesso ordine in cui compaiono nell'input.

Assunzioni e casi di test

Assunzioni

- $1 \leq N \leq 10.000$
- $1 \leq M \leq 10.000$
- $0 \leq K \leq 1.000$
- $1 \leq T, T_{min}, T_{max} \leq 10.000$
- Ogni grafo è diretto
- $I \neq S \neq F$
- Il nodo di arrivo F è sempre raggiungibile sia da I che da S
- Non ci sono self-loop, cioè archi (U, U)
- Non ci sono archi doppi, nemmeno tra corridoi con o senza ventole, ovvero esiste una sola tripla (U, V, T) o quadrupla (U, V, T_{min}, T_{max}) per una data coppia (U, V)

Casi di test Ci sono 20 casi di test in totale:

- in 6 casi su 20, il peso di tutti gli archi è pari a 1 e $K = 0$;
- in 10 casi su 20, $K = 0$;
- in 4 casi su 20, $K = 1$;
- in 5 casi su 20, $K \geq 2$.

I limiti di tempo e memoria sono:

- Tempo limite massimo: 1 secondo;
- Memoria massima: 8 MB;

Punteggi e correttore

Ogni caso di test vale 5 punti se completo (parte obbligatoria e facoltativa), 3 punti se parziale (solo parte obbligatoria). Il punteggio massimo è di 100 punti.
In caso di output errato si prendono sempre 0 **punti**:

- se il risultato che date **(vincita, perdita o pareggio)** non è corretto, si ottengono 0 **punti**.
- se il risultato è corretto ma le distanze sono errate, si ottengono 0 **punti**.
- se il risultato è corretto, ma i K archi hanno pesi non coerenti con le distanze dichiarate, si ottengono 0 **punti**.
- se il risultato è corretto, i K archi hanno pesi coerenti con le distanze dichiarate, ma il percorso stampato non è coerente, si ottengono 0 **punti**.

Valutazione

Per valutazione del progetto:

- Conta il punteggio dell'**ultimo sorgente** inviato al sistema;
- Il progetto è superato con un punteggio non inferiore a 30 punti;
- C'è un limite di 40 sottoposizioni per gruppo;

Dataset di esempio

Potete trovare un dataset equivalente con cui esercitarvi in locale:
judge.science.unitn.it/slides/asd22/dataset_amogasd.zip.
Nota: Il dataset di esempio contiene in output solo la prima riga, che indica se è possibile far vincere l'impostore (1), se gli studenti vincono sempre (2), o se si può arrivare a un pareggio (0), senza ulteriori informazioni.

Esempi di input/output

File `input.txt`	File `output.txt`

`9 8 3`
`3 6 0`
`0 1 2`
`0 8 3`
`2 0 2`
`3 1 4`
`3 4 3`
`5 4 2`
`6 5 1`
`8 7 2`
`1 2 1 4`
`4 0 1 7`
`5 7 3 5`
	`1`
	`7 8`
	`1 5 3`
	`4`
	`3 1 2 0`

File `input.txt`	File `output.txt`

`9 8 3`
`3 6 0`
`0 1 5`
`0 8 3`
`2 0 6`
`3 1 4`
`3 4 3`
`5 4 2`
`6 5 1`
`8 7 2`
`1 2 1 4`
`4 0 1 5`
`5 7 3 5`
	`0`
	`8 8`
	`1 5 5`
	`3`
	`3 4 0`

File `input.txt`	File `output.txt`

`9 8 3`
`3 6 0`
`0 1 5`
`0 8 3`
`2 0 4`
`3 1 4`
`3 4 4`
`5 4 2`
`6 5 1`
`8 7 2`
`1 2 1 4`
`4 0 1 5`
`5 7 3 5`
	`2`
	`9 8`
	`1 5 3`
	`4`
	`3 1 2 0`

File `input.txt`	File `output.txt`

`5 10 4`
`3 1 4`
`0 2 2`
`4 1 3`
`2 1 2`
`4 0 2`
`2 4 3`
`0 1 3`
`3 2 2`
`1 4 2`
`1 2 2`
`1 3 1`
`3 4 1 2`
`0 4 2 2`
`3 1 1 1`
`3 0 2 3`
	`1`
	`1 2`
	`1 2 1 2`
	`2`
	`3 4`