

Algoritmi e Strutture Dati

ASD: Endgame (`endgame`)

Testo del problema

Slides originali su: judge.science.unitn.it/slides/asd20/prog2.pdf

Premessa. Una nuova minaccia incombe nell'Universo degli Algoritmi. L'Ineluttabile Thanos vuole fare dimenticare a tutti gli esseri umani l'esistenza degli Algoritmi Efficienti. Per fortuna, in difesa degli algoritmi, si è schierato il prodigioso Capitan A(SD) e tutta la sua squadra di supereroi: gli Avengers.

Le Pietre della Terminazione e il Guanto della Decidibilità. Per sventare il piano di Thanos, i nostri eroi devono collezionare le Pietre della Terminazione¹ ed usare il Guanto della Decidibilità². Esistono M tipi di pietre diverse, ciascuna con una propria massa (m_i) e un livello di energia (e_i). Per mantenere attivo il guanto viene consumata una certa quantità di energia per unità di tempo (R)³, ma raccogliendo abbastanza pietre nel minor tempo possibile i nostri eroi potranno usare il Guanto e fermare Thanos.

Il mondo quantico. Ci sono M tipi di pietre diverse, sparse nello spaziotempo in N città raggiungibili dal mondo quantico. Ogni città può costituire un numero variabile di pietre maggiore o uguale a zero. Passare da una città all'altra ha un costo in termini di tempo⁴ e ogni salto quantistico tra due città può andare in entrambe le direzioni. Il teltrasporto quantistico permette ad ogni città di essere collegata ad ogni altra città. La base degli Avengers si trova nella città S . I nostri eroi partono e devono tornare alla base. Una possibile mappa del mondo è presente in Figura 1.

Il mondo quantico è abbastanza strano e quindi ci sono svariate limitazioni:

- I nostri eroi dovranno visitare ogni città, una ed una sola volta. Non è possibile passare per una stessa città più volte.
- In ogni città si può prendere al massimo 1 pietra, quindi è possibile che non venga raccolta alcuna pietra in una città.
- Se si vuole prendere una pietra nella città di partenza S , questa viene raccolta alla partenza e non quando si torna.

I limiti del guanto. Anche l'utilizzo del guanto è soggetto ad alcune regole:

- Nonostante la grande potenza dei nostri eroi, nessuno può sopportare un Guanto troppo carico di pietre, Il Guanto ha una portata di massa pari a C .
- Un guanto carico di pietre rallenta i suoi spostamenti quantici, Capitan A(SD) comincia il suo percorso con il Guanto scarico potendo muoversi con una certa v_{\max} , tuttavia la velocità calerà in maniera proporzionale al peso accumulato nel tempo, fino a raggiungere una velocità v_{\min}

$$v = v_{\max} - W \frac{v_{\max} - v_{\min}}{C} \quad (1)$$

dove:

¹https://en.wikipedia.org/wiki/Halting_problem

²https://en.wikipedia.org/wiki/Undecidable_problem

³Se il fatto che le unità di tempo non siano specificate esplicitamente vi infastidisce, potete pensare che questo consumo di energia sia *orario*.

⁴Similmente a quanto sopra, potete considerare che il tempo necessario per andare da una città all'altra sia espresso in *ore*.

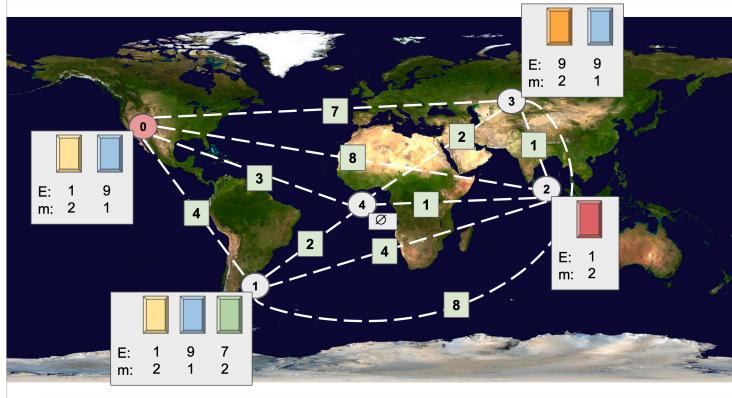


Figura 1: Una mappa della regione con 5 città $\{0, 1, 2, 3, 4\}$. La base degli Avengers si trova nella città 0. Ci sono 5 pietre disponibili $P_i(m_i, e_i)$: $P_1(2, 1)$ (pietra gialla), $P_2(1, 9)$ (pietra azzurra), $P_3(2, 7)$ (pietra verde), $P_4(2, 1)$ (pietra rossa), $P_5(2, 9)$ (pietra arancio). Le disponibilità nelle città sono le seguenti: $A_1 : \{0, 1\}$ ($\ell_{A_1} : 2$), $A_2 : \{0, 1, 3\}$ ($\ell_{A_2} : 3$), $A_3 : \{1\}$ ($\ell_{A_3} : 1$), $A_4 : \{2\}$ ($\ell_{A_4} : 1$), $A_5 : \{3\}$ ($\ell_{A_5} : 1$).

- v_{\min} e v_{\max} sono rispettivamente la velocità minima (guanto a pieno carico) e massima (guanto vuoto);
- W è la massa totale delle pietre attualmente caricate nel guanto;
- C è la capacità massima del guanto;

Obiettivo

Il vostro compito è aiutare gli Avengers a stabilire come visitare le varie città attraverso il mondo quantico e quali pietre raccogliere di modo tale da **massimizzare l'energia finale del guanto**.

La formula per il calcolo dell'energia finale è la seguente:

$$E(p, t) = G(p) - R \cdot T(p, t) \quad (2)$$

dove:

- p è la lista che indica quali pietre sono state raccolte in quali città;
- t è il percorso seguito ovvero la lista delle città visitate, in ordine. Questa lista deve necessariamente iniziare e terminare dalla base degli Avengers, S ;

e quindi:

- $G(p)$ è l'energia delle pietre raccolte e caricate nel guanto, che dipende solamente dalle pietre raccolte;
- $T(p, t)$ è il tempo impiegato per effettuare visitare le città, che dipende da quali pietre vengono raccolte e dall'ordine in cui sono visitate le varie città;

Nota: è possibile che l'energia finale sia **negativa**, in ogni caso l'obiettivo è accumulare quanta più energia possibile. Una soluzione del problema è presentata in Figura 2. L'energia finale E è ottenuta secondo l'Equazione 2. Nell'esempio in Figura 2

- $G = 11$, la somma delle energie delle pietre raccolte
- $R = 1$, $v_{\max} = 11$, $v_{\min} = 1$, $C = 5$
- T è dato dal tempo impiegato per percorrere gli archi del percorso, con una certa velocità influenzata dal peso trasportato

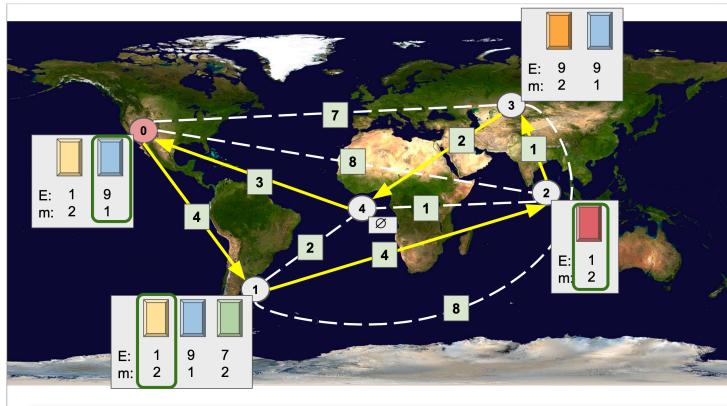


Figura 2: Una soluzione valida del problema con $C = 5$.

- $v_{01} = 11 - \frac{1*10}{5} = 9 \Rightarrow t_{01} = 4/9 = 0.44\dots$
- $v_{12} = 11 - \frac{3*10}{5} = 5 \Rightarrow t_{12} = 4/5 = 0.8$
- $v_{23} = 11 - \frac{5*10}{5} = 1 \Rightarrow t_{23} = 1/1 = 1$
- $v_{34} = 11 - \frac{5*10}{5} = 1 \Rightarrow t_{34} = 2/1 = 2$
- $v_{40} = 11 - \frac{5*10}{5} = 1 \Rightarrow t_{40} = 3/1 = 3$

Quindi il tempo finale del giro è:

$$T = t_{01} + t_{12} + t_{23} + t_{34} + t_{40} = 7.244444444$$

E l'energia finale è data da:

$$E = 11 - 1 * 7.244444444 = 3.755555555$$

Esempi di soluzioni non valide

Le Figure 3, 4, 5 e 6 mostrano dei casi di soluzioni non valide.

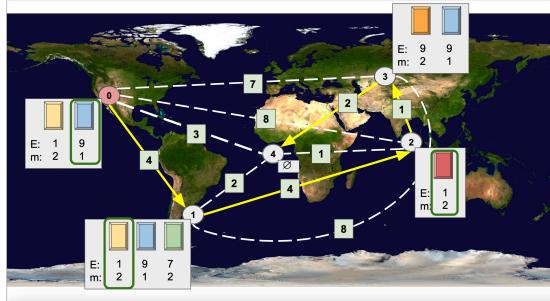


Figura 3: Questa soluzione non è valida perché il giro termina nella città 4, invece di tornare alla base situata in città 0.

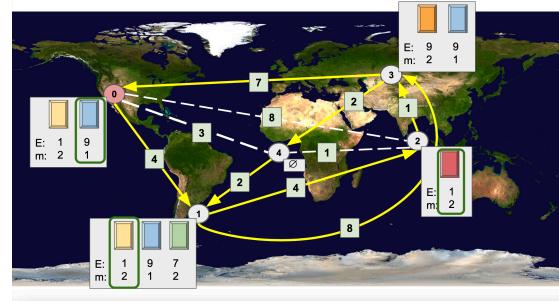


Figura 4: In questo caso il giro proposto è: [0, 1, 2, 3, 4, 1, 3, 0]. Questa soluzione non è valida perché il giro passa due volte dalle città 1 e 3.

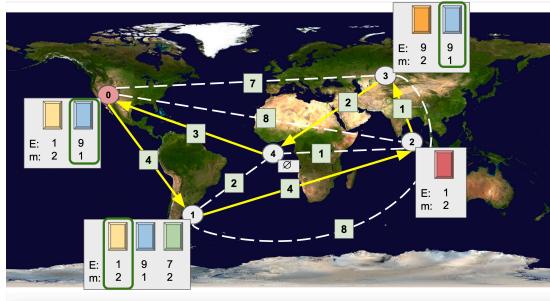


Figura 5: Questa soluzione non è valida perché la pietra blu (pietra 1) viene presa due volte..

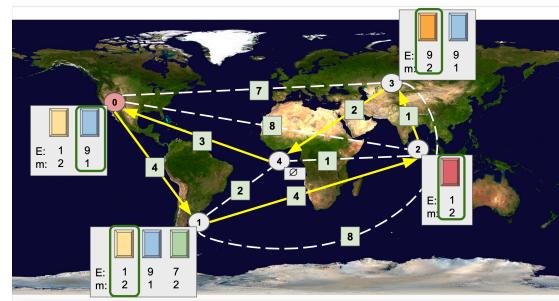


Figura 6: Questa soluzione non è valida perché prendiamo pietre per una massa totale di 7, che è maggiore della portata massima del Guanto (uguale a 5).

Input/Output

Nota: gli indici delle città e degli item partono da 0.

Input: un file con una mappa delle varie città, dei loro collegamenti attraverso il teletrasporto quantico e le pietre disponibili. In particolare, il file di input è costituito da $2 + M + 2M + N - 1$ righe.

- La prima riga riporta 2 numeri interi: N (int) e S (int), rispettivamente il numero di città e la città di partenza, dove si trova la base degli Avengers.
- La seconda riga riporta 5 numeri: M (int), C (int), R (double), v_{\min} (double), v_{\max} (double) il numero di diversi tipi di pietre, la capacità del Guanto, il consumo di energia del Guanto per unità di tempo, la velocità minima e la velocità minima degli Avengers.

- Le successive M righe sono costituite da 2 interi ciascuna m_i (`int`) e e_i (`int`), che rispettivamente la massa e l'energia dell' i -esima tipo di pietra;
- Le successive $2M$ righe riportano le disponibilità delle pietre nelle varie città. Per ogni pietra, vengono stampate due righe:
 - la prima contiene un intero che riga la lunghezza ℓ_{A_i} (`int`) della lista di disponibilità dell' i -esima pietra;
 - la riga successiva stampiamo contiene ℓ_{A_i} interi: gli id delle città (`int`) in cui la pietra i -esima è presente;
- Le successive $N - 1$ righe descrivono le distanze tra le città: la prima riga contiene il peso (`int`) dell'arco da 1 a 0; la seconda riga contiene i pesi degli archi da 2 a 0 e 1, la terza riga contiene i pesi degli archi da 3 a 0, 1 e 2, ecc. La i -esima riga contiene i valori.

Specifica dell'input:

```

N S
M C R v_min v_max
m_1 e_1
...
m_M e_M
l_{A_1}
a_11 ... a_11_{A_1}
...
l_{A_M}
a_M1 ... a_11_{A_M}
w(1, 0)
w(2, 0) w(2, 1)
...
w(N-1, 0) w(N-1, N-2)

```

Output: un file con le vostre proposte di soluzione sulla mappa, ogni proposta è composta da 4 righe:

- 3 numeri: E (`double`), G (`double`) e T (`double`), rispettivamente l'energia finale del guanto, l'energia delle pietre raccolte e il tempo impiegato per il giro come definito nell'Equazione 2;
- la lista p che indica quali pietre sono state raccolte in quali città: p_1, \dots, p_M . Se una pietra non è stata raccolta stampare -1 per quella pietra;
- la lista t che indica il percorso seguito, ovvero la lista delle città visitate, in ordine: t_0, \dots, t_N con $t_0 = S$ e $t_N = S$;
- tre asterischi ***;

Specifica dell'output:

```

E G T
p_1 ... p_M
t_0 ... t_{N-1} t_N
***
```

Precisione numerica

Le variabili E, G, T nell'output vanno memorizzare come variabili di tipo `double`. Va utilizzata la libreria `<iomanip>` per stampare l'output in notazione scientifica e con 10 cifre decimali.

```
#include <iomanip>
...
int main() {
    double result;
    ...
    out << scientific << setprecision(10) << E << " ";
    out << scientific << setprecision(10) << G << " ";
    out << scientific << setprecision(10) << T << endl;
    ...
}
```

Saranno accettati gli output in cui le variabili E, G, T dichiarate nell'output si discostano meno di $\frac{1}{10.000}$ dal valore calcolato usando p e t e l'Equazione 2.

Punteggio

- Ci sono 20 casi di test: ogni test assegna un punteggio di massimo 5 punti per un totale massimo teorico di 100 punti.
- Una soluzione è valida se rispetta tutte le richieste. Soluzioni non valide fanno **zero punti!**
- I percorsi validi ottengono questo punteggio:
- Il punteggio viene calcolato usando l'**ultima soluzione** terminata con tre asterischi ***.

I parametri di valutazione sono i seguenti:

- G - energia contenuta nel guanto;
- R - consumo di energia del guanto per unità di tempo;
- T - tempo impiegato per visitare le città;
- $maxT$ - upper bound del tempo per visitare le città;
- $minT$ - lower bound del tempo per visitare le città;
- $maxG$ - energia ottima caricabile nel guanto;

Per ogni caso di test per cui la vostra soluzione fornisce un output entro i limiti di tempo e memoria otterrete il seguente punteggio P :

$$P = \frac{G + R(maxT - T)}{maxG + R(maxT - minT)} * 5$$

nota: non è necessario che calcoliate il punteggio della vostra soluzione - vi servirebbero i parametri dei bound $minT$, $maxT$ e $maxG$ che non avete. Il vostro obiettivo è in ogni caso di **massimizzare E** .

Esempi (punteggio)

Nell'esempio di cui sopra, la risposta del sistema del valutazione è:

```
Soluzione valida: E(3.76) G(11.00) T(7.24) maxG(25.00) minT(1.09) maxT(35.00)
                    randMaxT(7.24) scoreRandMaxT(0.35) 0.353090
```

Il punteggio è calcolato nel modo seguente:

$$P = \frac{11 + 1 \cdot (7,244444 - 7,244444)}{25 + 1 \cdot (7,244444 - 1,090909)} \cdot 5 = 0,35308994 \cdot 5 \approx 1,77$$

Valutazione

Per la valutazione del progetto:

- Conta il punteggio dell'**ultimo sorgente** inviato al sistema;
- Il progetto è superato con un punteggio non inferiore a 50 punti;
- C'è un limite di 40 sottoposizioni per gruppo;

Limiti e assunzioni

- $1 \leq N \leq 2.000$, $0 \leq S < N$
- $0 \leq M \leq 10.000$
- $0 \leq C \leq 10.000.000$
- $0 \leq R \leq 5.000$
- $0 \leq v_{\min} \leq v_{\max} \leq 1.000$
- $1 \leq m_i \leq 100.000$, $1 \leq e_i < 100.000$
- Ogni grafo è completo.
- Ogni grafo è non diretto.

Casi di test

- Ci sono 20 casi di test in totale.
- In almeno 3 casi su 20 in ogni città c'è una ed una sola pietra, non ci sono due città con pietre uguali.
- In almeno 3 casi su 20 tutti gli archi del grafo hanno lo stesso peso.
- In almeno 4 casi su 20 vale la disuguaglianza triangolare per i pesi degli archi del grafo.
- In almeno 10 casi su 20 non ci sono particolari limitazioni.

Limiti delle risorse

- Tempo di esecuzione: 5 secondi
- Memoria: 48 MB

Dataset di esempio

Per gli input forniti nel dataset di esempio non è stata calcolata una soluzione ottima. Per questo motivo il dataset non contiene anche i relativi output, solitamente messi a disposizione.

Istruzioni di compilazione

Di seguito riportiamo le istruzioni per testare i vostri programmi su vari sistemi. Si suppone che il sorgente con il vostro codice si chiami file `endgame.cpp`. I file `endgame.cpp`, `grader.cpp` e `endgame.h` devo stare nella stessa cartella.

Sistemi GNU/Linux

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o endgame endgame.cpp grader.cpp
```

Sistemi Mac OS X

Su sistemi Mac OS X usate il seguente comando di compilazione:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -o endgame endgame.cpp grader.cpp
```

Se ottene un errore del tipo: `use of undeclared identifier quick_exit`, sostituite in `grader.cpp` l'istruzione `quick_exit(EXIT_SUCCESS);` con `exit(EXIT_SUCCESS);`.

Sistemi Windows

Per il sistema Windows 10 potete installare il “Windows Subsystem for Linux”⁵. Successivamente potete installare i tool necessari per usare Visual Studio Code⁶ o Visual Studio 2017⁷ seguendo le relative guide riportate nelle note. Usando questo sistema fate attenzione a dove salvate i file e a quale nome gli date in quanto potreste avere delle difficoltà con percorsi che contengano spazi e caratteri speciali.

In alternativa, o per sistemi precedenti a Windows 10 potete installare *Cygwin*⁸, un ambiente completamente POSIX-compatibile per Windows. Anche in questo caso esistono guide per configurare i comuni editor disponibili su Windows di modo che utilizzino l'ambiente Cygwin, come per esempio Visual Studio⁹.

Una volta installato Cygwin è possibile simulare quanto avviane su arena compilando il proprio sorgente senza includere l'header `endgame.h` e il grader `grader.cpp`:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o endgame endgame.cpp
```

e lanciare il comando come:

```
timeout.exe 5 ./endgame
```

`timeout.exe` arresterà il programma dopo 5 secondi.

-
- 5 <https://docs.microsoft.com/en-us/windows/wsl/install-win10>
 - 6 <https://code.visualstudio.com/docs/cpp/config-wsl>
 - 7 <https://devblogs.microsoft.com/cppblog/targeting-windows-subsystem-for-linux-from-visual-studio/>
 - 8 <https://www.cygwin.com/>
 - 9 <https://devblogs.microsoft.com/cppblog/using-mingw-and-cygwin-with-visual-cpp-and-open-folder/>

Esempi di input/output

| File input.txt | File output.txt |
|---|---|
| <pre> 5 0 5 5 1.0 1.0 11.0 2 1 1 9 2 7 2 1 2 9 2 0 1 3 0 1 3 1 1 1 2 1 3 4 8 4 7 8 1 3 2 1 2 </pre> | <pre> 3.7555555556 11.0 7.2444444444 1 0 -1 2 -1 0 1 2 3 4 0 ***</pre> |
| <pre> 6 3 6 0 1 1.0 1.0 1 1 1 1 1 1 1 1 1 1 1 1 6 0 1 2 3 4 5 6 0 1 2 3 4 5 6 0 1 2 3 4 5 6 0 1 2 3 4 5 8 9 9 9 10 3 5 4 9 3 7 7 2 8 8 </pre> | <pre> -3.100000000e+01 0.000000000e+00 3.100000000e+01 -1 -1 -1 -1 -1 3 2 5 0 4 1 3 ***</pre> <pre> -2.700000000e+01 0.000000000e+00 2.700000000e+01 -1 -1 -1 -1 -1 3 2 5 0 1 4 3 ***</pre> |

| File input.txt | File output.txt |
|---|---|
| <pre> 5 2 8 100 0 1.0 1.0 1 6 1 3 1 9 1 4 15 8 11 16 15 6 15 1 5 1 3 2 4 0 2 0 3 5 3 4 2 0 1 1 0 1 4 1 0 2 2 4 4 4 2 1 3 1 1 1 1 1 1 1 1 1 1 </pre> | <pre> 2.7000000000e+01 2.7000000000e+01 5.0000000000e+00 0 3 2 -1 4 -1 -1 1 2 0 1 3 4 2 *** 3.0000000000e+01 3.0000000000e+01 5.0000000000e+00 1 3 2 0 4 -1 -1 -1 2 1 3 0 4 2 *** 3.2000000000e+01 3.2000000000e+01 5.0000000000e+00 0 3 1 -1 4 -1 2 -1 2 0 1 3 4 2 *** 4.0000000000e+01 4.0000000000e+01 5.0000000000e+00 3 -1 2 -1 4 0 -1 1 2 0 3 1 4 2 *** 4.2000000000e+01 4.2000000000e+01 5.0000000000e+00 1 3 2 -1 4 0 -1 -1 2 1 0 3 4 2 *** 4.5000000000e+01 4.5000000000e+01 5.0000000000e+00 3 -1 1 -1 4 0 2 -1 2 1 0 3 4 2 ***</pre> |