Secondo Progetto ASD 2020/2021

ASD: ENDGAME

Una minaccia incombe

Una nuova minaccia incombe nell'Universo degli Algoritmi. L'Ineluttabile Thanos vuole fare dimenticare a tutti gli esseri umani l'esistenza degli Algoritmi Efficienti.

I NOSTRI EROI

Per fortuna, in difesa degli algoritmi, si è schierato il prodigioso Capitan A(SD)...

I NOSTRI EROI

... 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 Terminazione ed usare il Guanto della Decidibilità.

Le Pietre della Terminazione e il Guanto della Decidibilità

Per sventare il piano di Thanos, i nostri eroi devono collezionare le Pietre della Terminazione ed usare il Guanto della Decidibilità.
Esistono M tipi di Pietre diverse, ciascuna con una diversa massa (m_i) e con un diverso livello di energia (e_i).

Le Pietre della Terminazione e il Guanto della Decidibilità

Per sventare il piano di Thanos, i nostri eroi devono collezionare le Pietre della Terminazione ed usare il Guanto della Decidibilità.
Esistono M tipi di Pietre diverse, ciascuna con una diversa massa (m_i) e con un diverso livello di energia (e_i). L'utilizzo del Guanto causa un consuma di energia per unità di tempo pari a R.

Il mondo quantico (I)

- Le pietre sono in N città raggiungibili con il teletrasporto attraverso il mondo quantico.
- Ogni città custodisce un numero di pietre maggiore o uguale a zero.
- Andare da una città all'altra ha un proprio costo in termini di tempo e ogni salto quantistico tra due città può andare in entrambe le direzioni.
- Il trasporto quantistico permette ad ogni città di essere collegata ad ogni altra città.

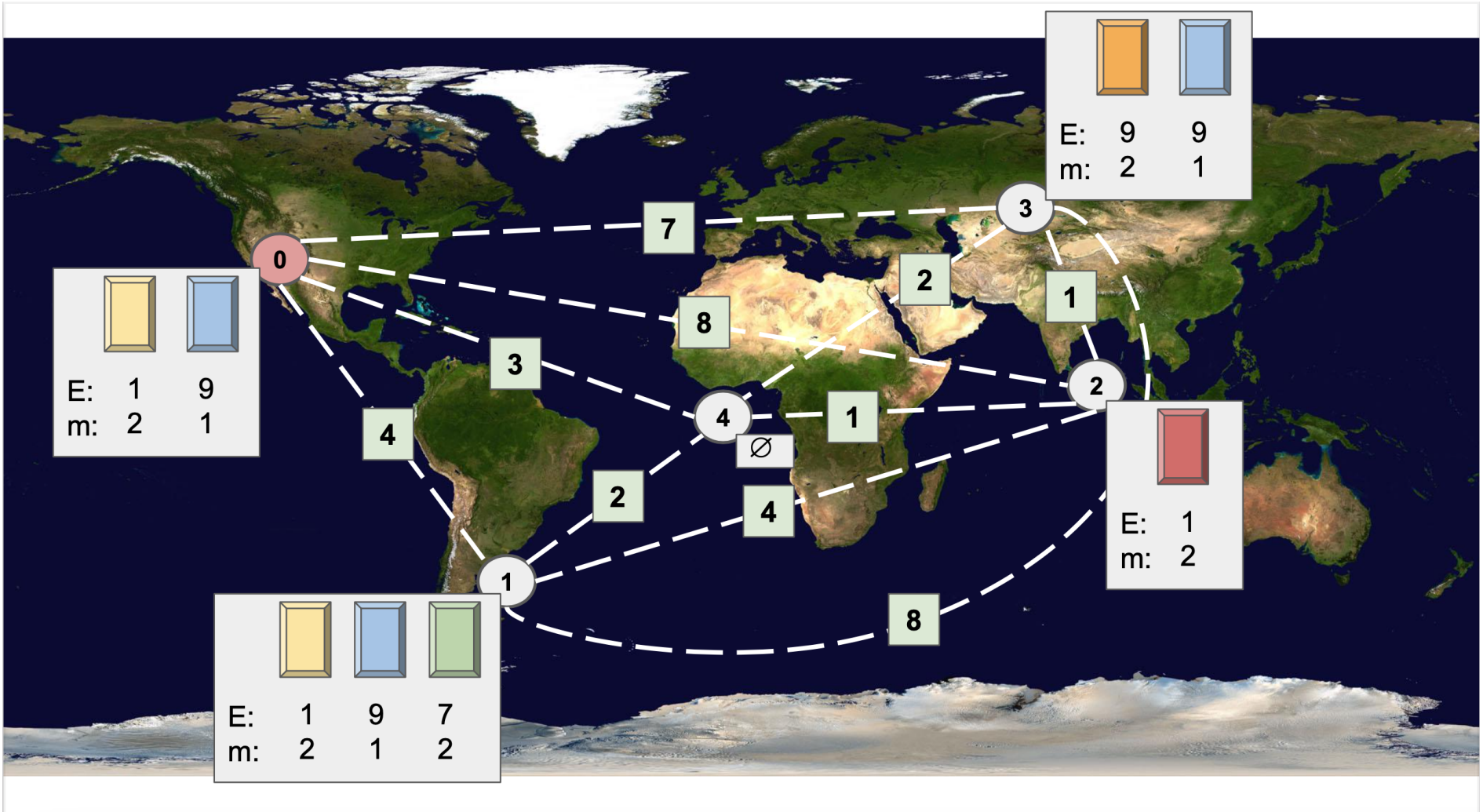
I nostri eroi partono dalla base S e devono tornarci.

I limiti quantico (II)

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).

Mappa



Le incognite

Un peso insostenibile

Ogni pietra ha una massa. Purtroppo, nonostante la grande potenza dei nostri eroi, nessuno può sopportare un Guanto troppo carico di pietre. Infatti, il Guanto ha una portata di massa pari a C.

Un peso sulle spalle

Nonostante Capitan A(SD) sia dotato di forti muscoli, trasportare un Guanto carico è una faticaccia, e 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}.

Un peso sulle spalle

La velocità con il quale Capitan A(SD) viaggia tra due città viene calcolata in questo modo:

$$v = v_{\max} - W \cdot \frac{v_{\max} - v_{\min}}{C}$$

dove $v_{\max}$ e $v_{\max}$ sono le velocità massime e minime date in input, C è la capacità del guanto data in input e W è la massa delle pietre raccolte fino a quel momento.

Il vostro compito

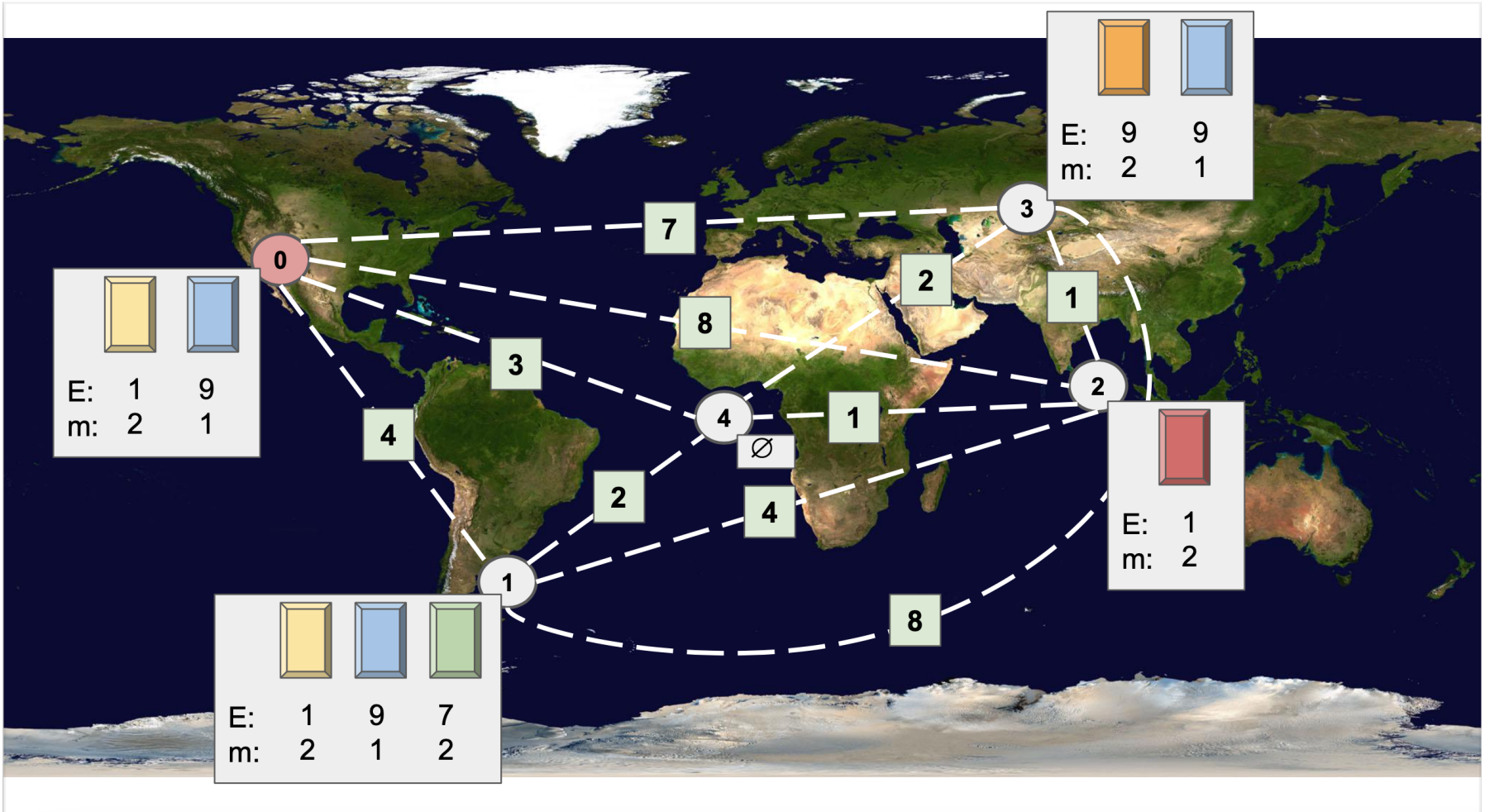
Capitan A(SD) si prepara per questa impresa, e nel frattempo chiede a voi di sviluppare una strategia per la raccolta delle pietre. Vi chiede quindi di trovare una strategia che massimizzi il valore:

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

dove $G(p)$ è l'energia raccolta tramite le pietre, R è il tasso di consumo di energia per unità di tempo usato per il trasporto del Guanto e $T(p,t)$ è il tempo impiegato per visitare le città.

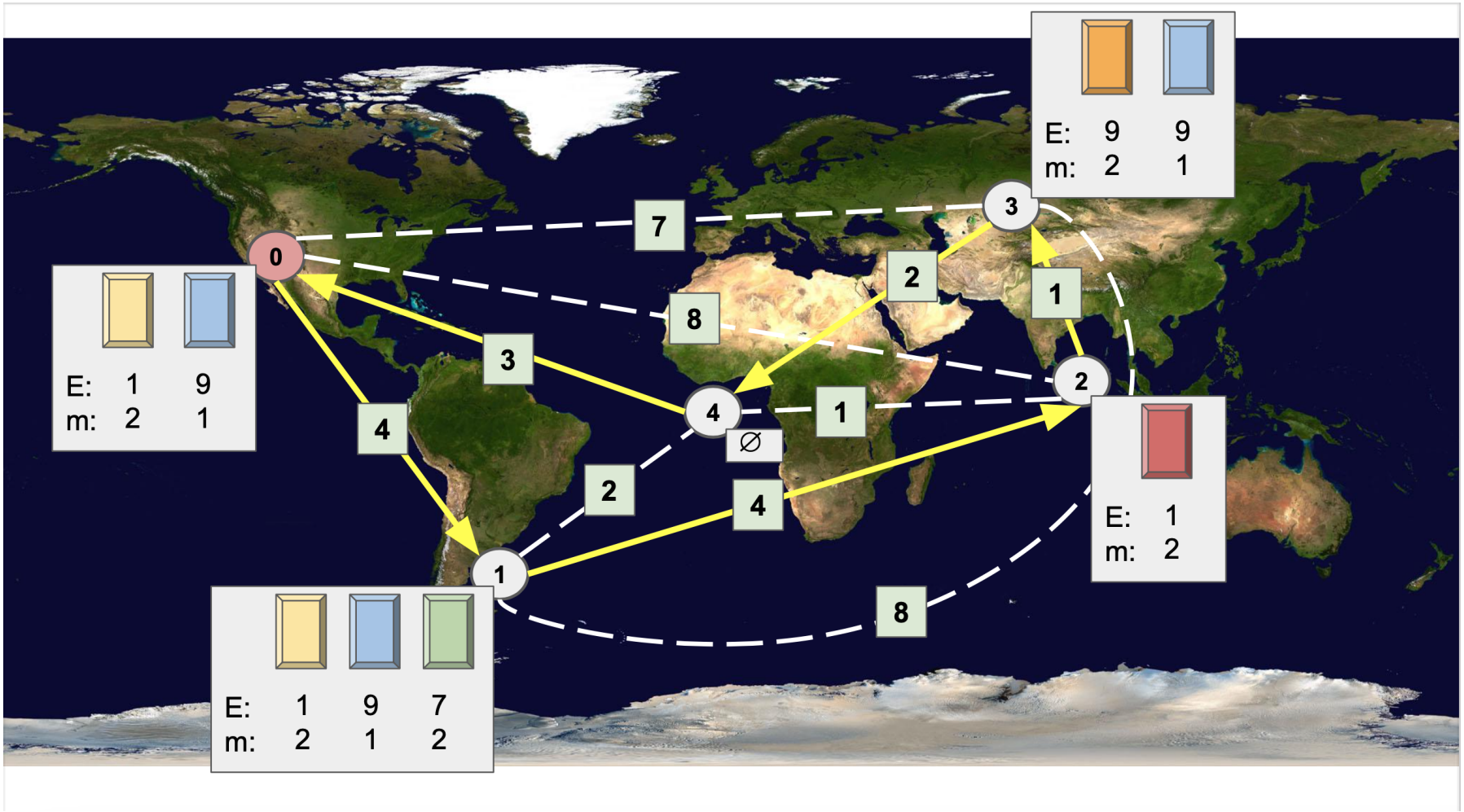
Avengers, compute!

Esempio



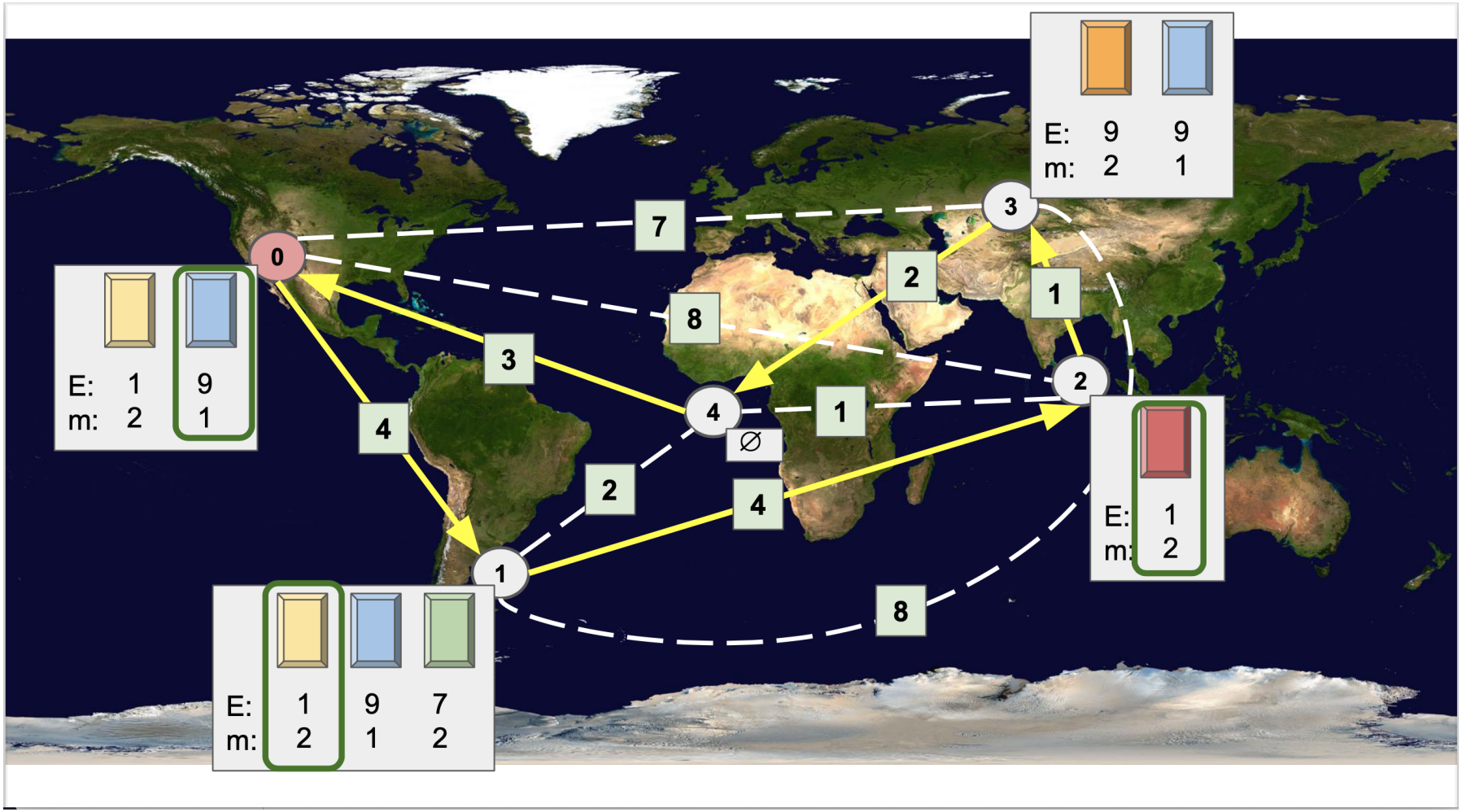
La città di partenza si trova nel nodo 0 del grafo completo dato dai collegamenti quantici. Ogni città custodisce un certo numero di pietre, tranne la città 4, che non ne custodisce nessuna.

Esempio



Percorso valido: parto da 0 e torno in 0, ogni città è visitata una sola volta.

Esempio



Assumiamo una portata massima dello zaino pari a 5. In ogni città che viene visitata si può prendere o meno una pietra. In questo caso collezioniamo tramite le pietre una energia pari a 11.

Esempio

L'energia finale E è ottenuta in questo modo:

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

In questo esempio:

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

  $T = t_{01} + t_{12} + t_{23} + t_{34} + t_{40} = 7.24$

  $E = 11 - 1 * 7.24 = 3.76$

Input

Un file con $2 + M + 2M + N - 1$ 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.

Input

- 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..

Output

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

- o_1 : 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
- o_2 : la lista p che indica quali pietre sono state raccolte in quali città: $p_1, \ldots, p_M$. Se una pietra non è stata raccoltà stampare -1 per quella pietra;
- o_3 : la lista t che indica il percorso seguito, ovvero la lista delle città visitate, in ordine: $t_0, \ldots, t_N$ con $t_0 = S$ e $t_N = S$;
- o_4 : tre asterischi $***$;

Esempio I

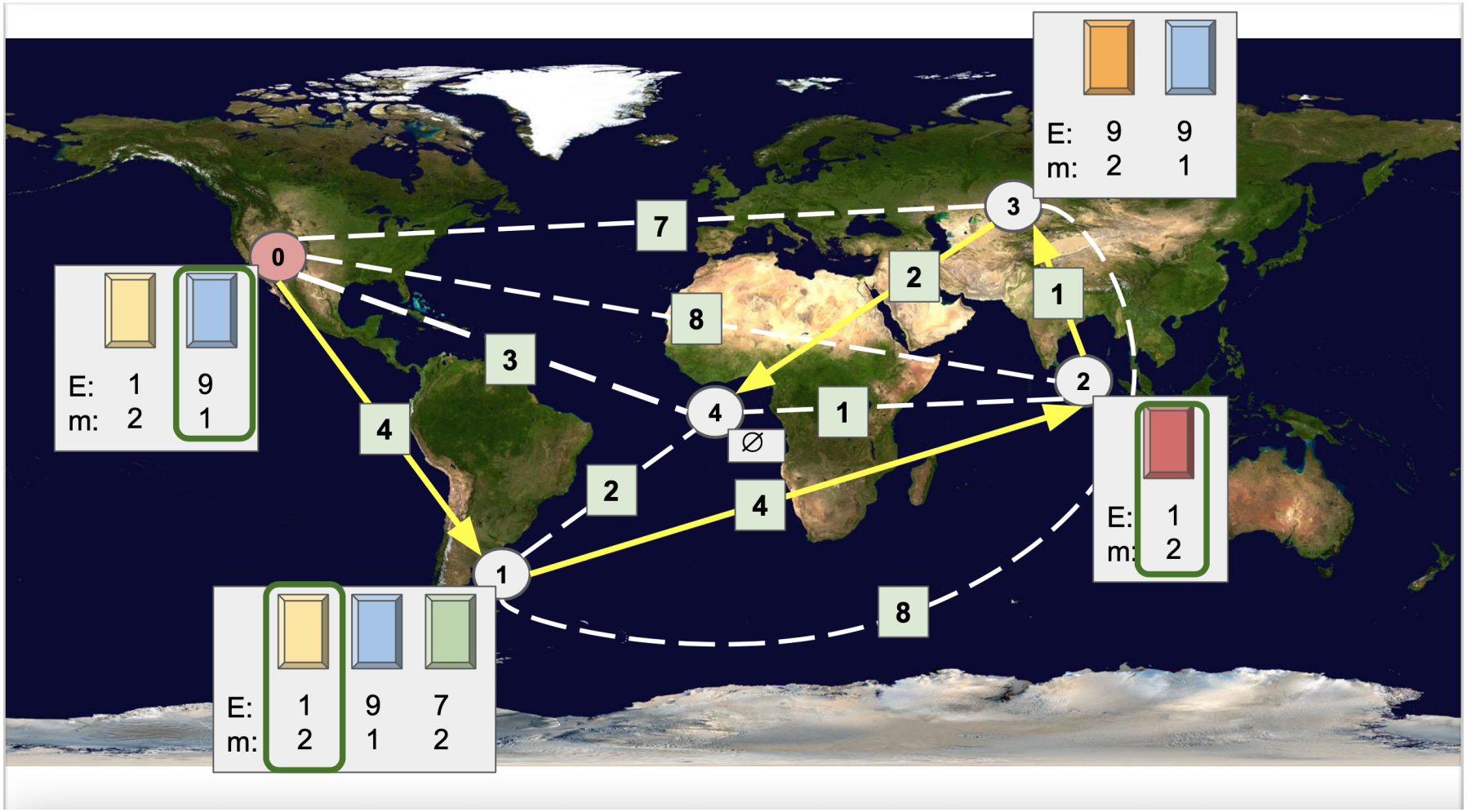
Input:

```
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
```

Output:

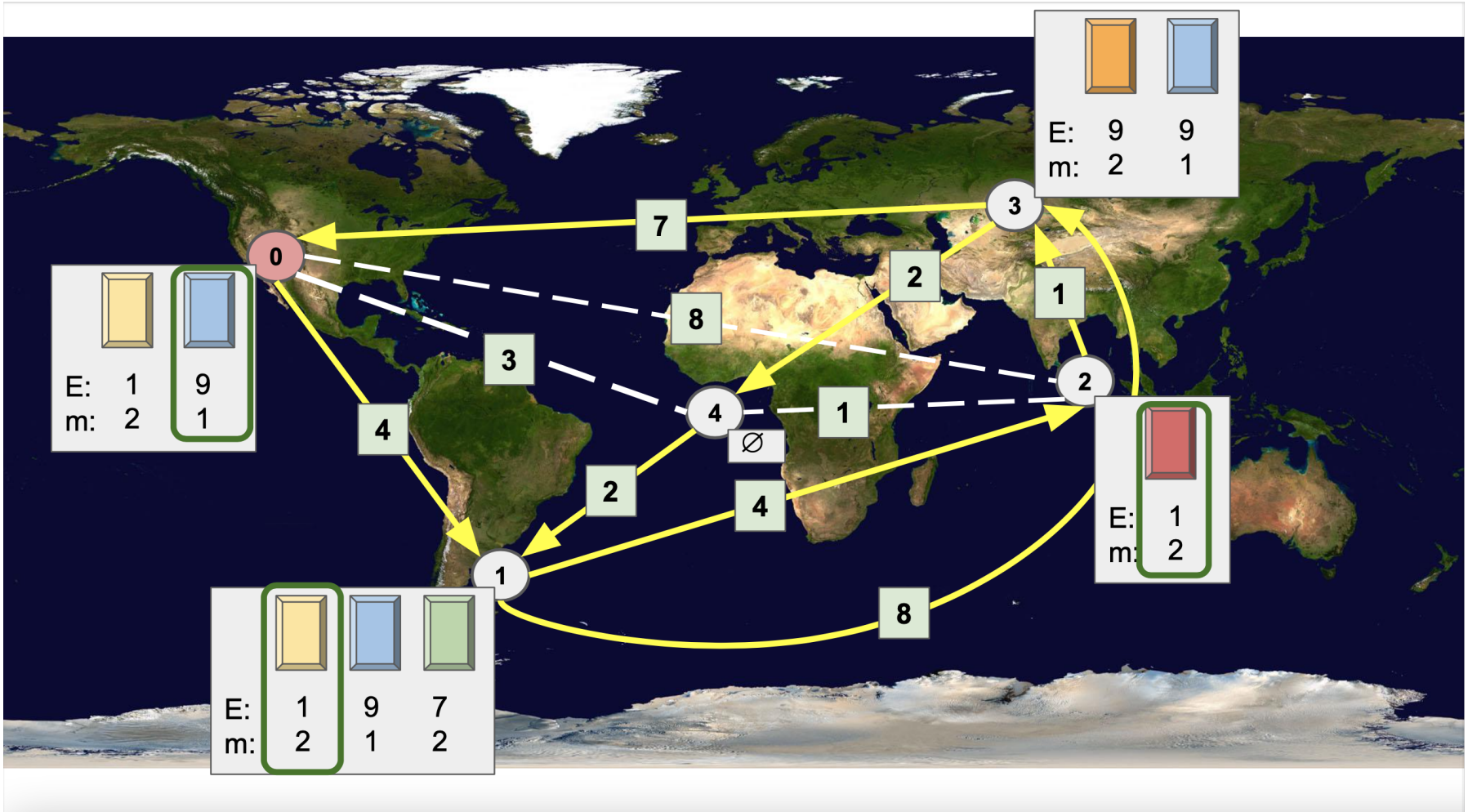
```
3.76 11.0 7.24
1 0 -1 2 -1
0 1 2 3 4 0
***
```

Esempio di soluzione errata I



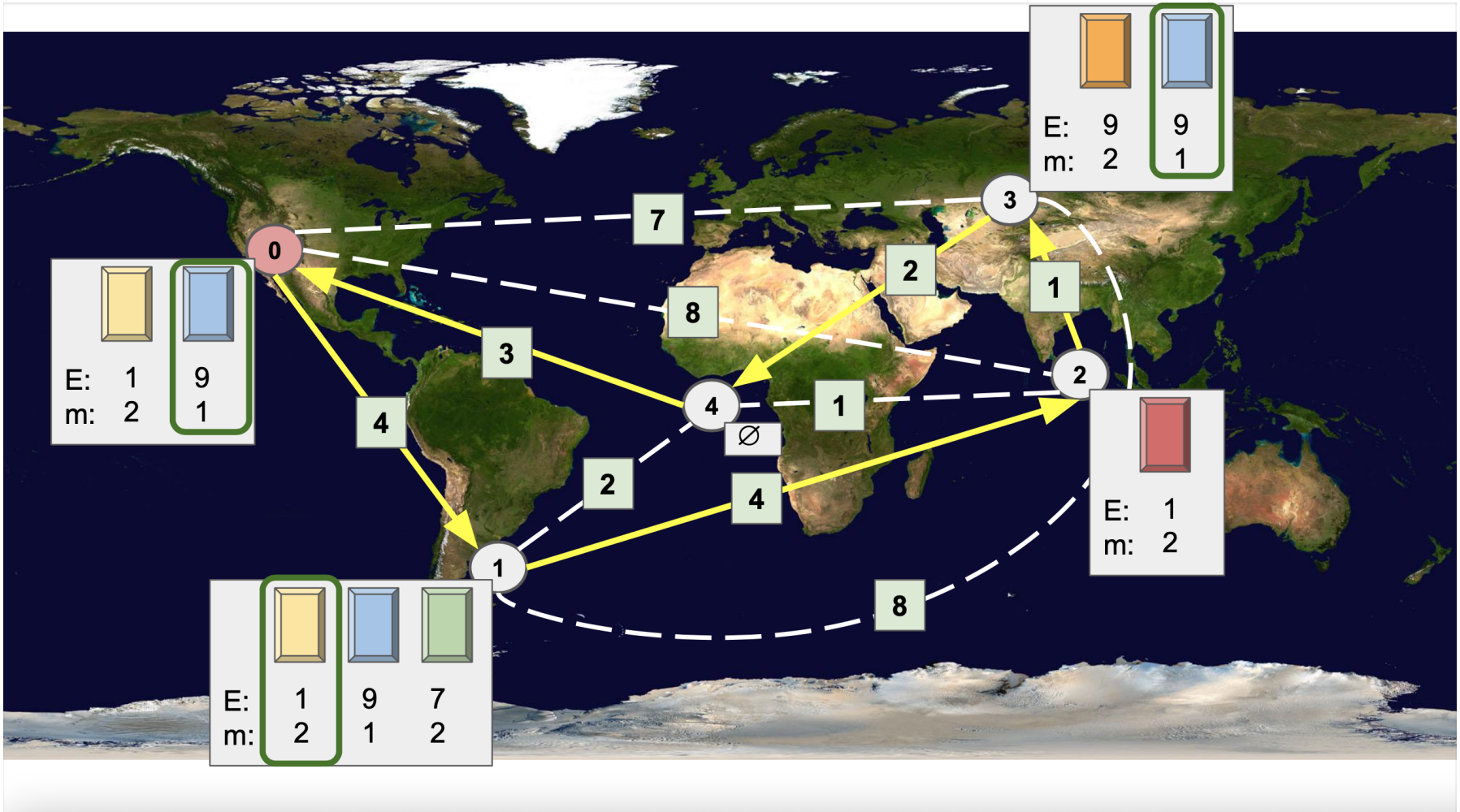
Questa soluzione non è valida perché il giro termina nella città 4, invece di tornare alla base situata in città 0.

Esempio di soluzione errata II



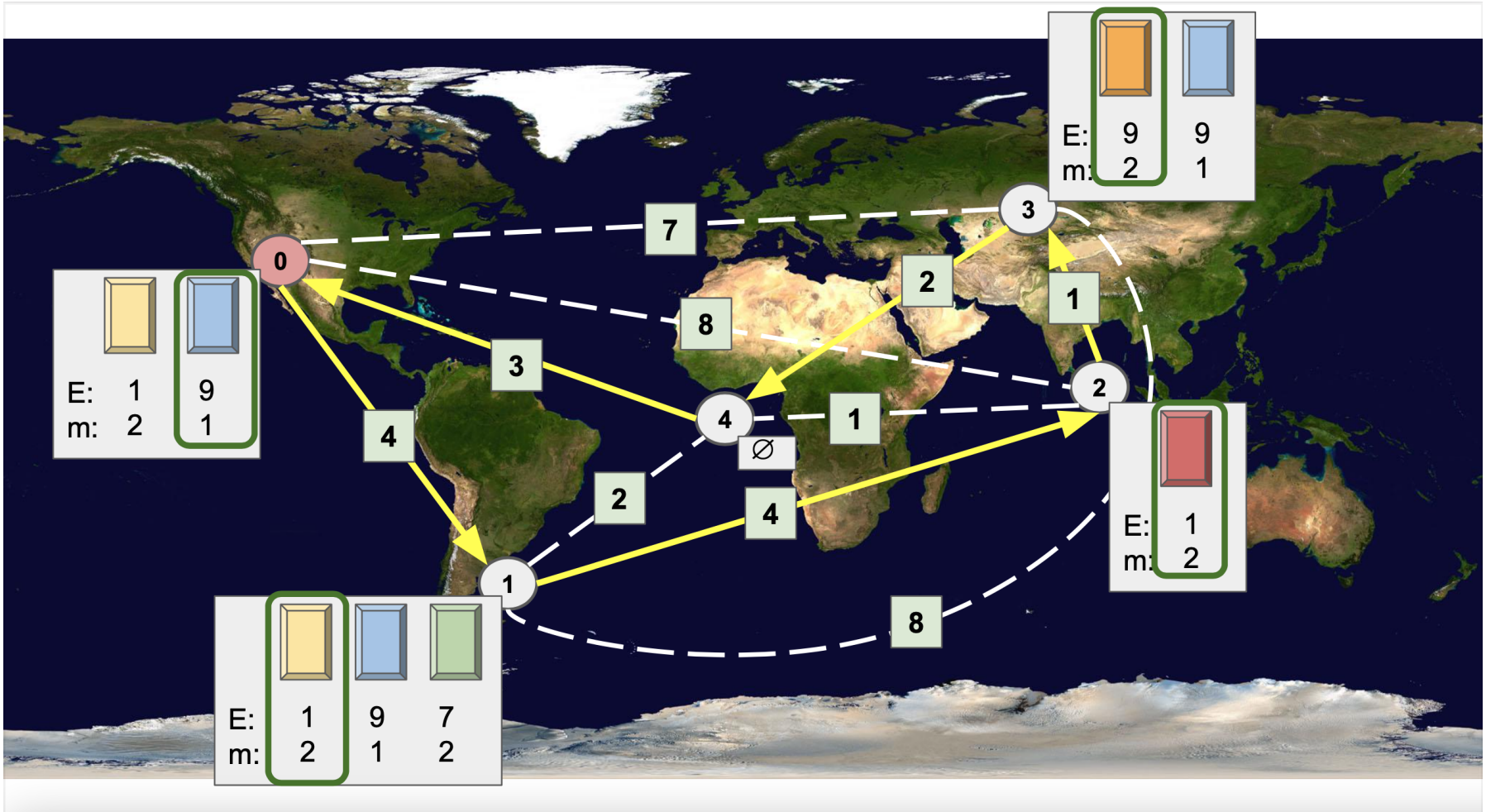
In questo caso il giro proposto è: $0, 1, 2, 3, 4, 1, 3, 0$. Questa soluzione non è valida perché il giro passa due volte dalle città 1 e 3.

Esempio di soluzione errata III



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

Esempio di soluzione errata IV



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).

Note su Input

Assunzioni generali

- $1 \leq N \leq 2.000$, $0 \leq S < N$
- $0 \leq M \leq 10.000$
- $0 \leq C \leq 10.000.000$
- $0 \leq R \leq 5.000$
- $0 \leq v_{min} \leq v_{max} \leq 1.000$
- $1 \leq m_i \leq 100.000$, $1 \leq e_i < 100.000$
- 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 di tempo e memoria

I limiti di tempo e memoria sono:

- Tempo limite massimo: 5 secondi.
- Memoria massima: 64 MB.

$\Rightarrow$ Limite di 40 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 non riporta gli output.

Output

Se scrivete una soluzione esponenziale:

- Importate `endgame.h` (scaricabile da judge).
- Man mano che migliorate la soluzione, scrivetela in output terminando la riga con `***`.
- La libreria arresterà il programma prima del timeout.

```
... include delle librerie di sistema ...
#include "endgame.h"
int main() {
  ...
}
```

Note:

- Il `main` va sempre dichiarato come `int main()` o `int main(void)`.
- Il correttore considererà l'**ultima soluzione** terminata da `***` quindi, anche se non stampate soluzioni multiple, terminate l'output con `***`.

Note sull'output

Quando si stampano i primi tre valori E, G e T, occorre stamparli in notazione scientifica con una precisione di 10 cifre decimali. Si può fare nel seguente modo, tramite la libreria `iomanip`:

```
#include <iomanip>
...
void print_solution() {
  ofstream out("output.txt");
  ...
  out << scientific << setprecision(10) << E << " ";
  out << scientific << setprecision(10) << G << " ";
  out << scientific << setprecision(10) << T << endl;
  ...
}
```

Compilare in locale

Per testare le vostre soluzioni in locale (supponiamo che il vostro file si chiami `endgame.cpp`):

- Scaricate `grader.cpp`
- Il comando di compilazione è il seguente

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

I file `endgame.cpp`, `grader.cpp` e `endgame.h` devono essere nella stessa cartella.

Per sistemi Mac OS X e Windows vedere la nota nel testo.

Nota

Per questo esercizio è necessario usare il `C++`, non è possibile usare il `C`.

Punteggio

Ogni caso di test vale 5 punti. Il punteggio massimo è di 100 punti.

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

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:

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

Punteggio

⇒ La **sufficienza è posta a 50 punti**.

- ✗ se una qualsiasi informazione data in output non è corretta entro i limiti di errore, si ottengono 0 punti.
- ✗ se una vostra soluzione dovesse dare un punteggio negativo, verranno assegnati 0 punti per quel caso di test.

Nota

`maxT` è il tempo risultante da una soluzione che segue città in modo casuale, raccogliendo pietre in modo casuale. Non è un vero e proprio upper bound, ma una soglia per valutare la vostra soluzione.

L'energia finale non deve essere per forza positiva

L'energia finale E data in output non deve essere per forza positiva affinché il punteggio sia positivo.

Test

C'è un limite di **40 sottoposizioni** per gruppo.

Dataset di esempio

Attenzione! 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.

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 50).

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

Consegna

Consegna: martedì 1 giugno 2021 ore 18:00

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

Suggerimenti

Cominciate subito a lavorare al progetto per presentarvi al prossimo ricevimento (giovedì 27 maggio) con tutte le domande che vorrete fare. In ogni caso, sappiate che:

- potete venire a ricevimento
- 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

Dont's

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)

Ricevimento

Date e orari

- giovedì 27 maggio 2021 dalle 11:30 alle 13:30 (lab);
- venerdì 28 maggio 2021 dalle 17:00 alle 18:00;
- sabato 29 maggio 2021 dalle 11:00 alle 12:00;
- lunedì 31 maggio 2020 dalle 17:00 alle 18:00;
- martedì 1 giugno 2021 dalle 13:30 alle 15:30 (lab);

⇒ I ricevimenti si svolgeranno su Discord, quando avrete bisogno di un aiuto scrivetelo sul canale del laboratorio.

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