W

Primo Progetto ASD 2022/2023

AmongASD

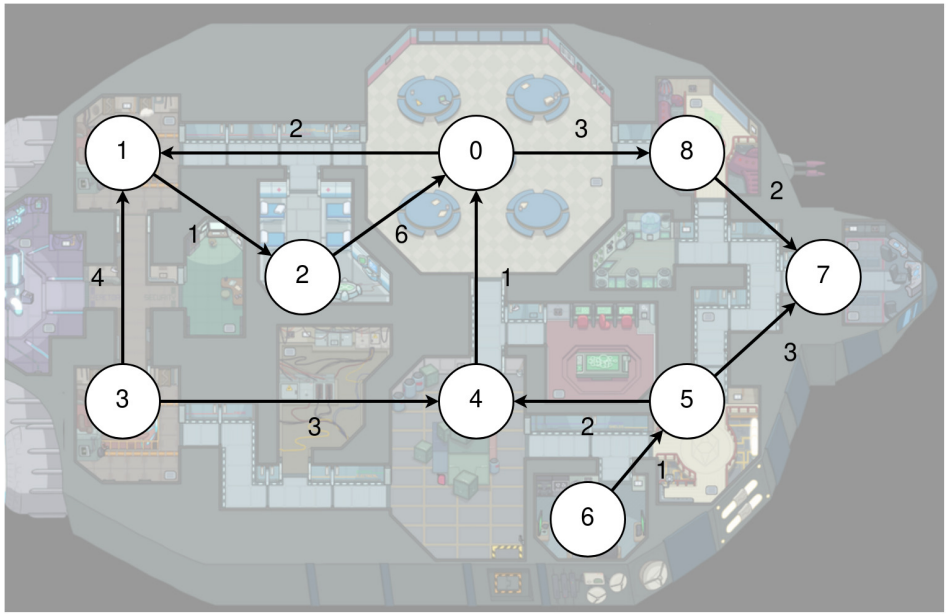
Il Prof. Montresus

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

L'Università non è altro che un grafo, dove varie stanze sono collegate tra loro da corridoi, che possono essere attraversati in un certo numero di unità di tempo.

Per motivi logistici, i corridoi possono essere percorsi solo in una direzione.



Voci di corridoio

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) ci sia nascosta la Bibbia™ che li può aiutare a superare l'esame.

L'IMPOSTORE!

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.

Percorrenze variabili (I)

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...

Percorrenze variabili (II)

... tramite il suo centro di comando.

E ora? Il vostro compito

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?

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 la stanza di partenza dell'impostore, degli studenti e 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 (parte obbligatoria)

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 nel migliore dei casi 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 (i corridoi vanno stampati nello stesso ordine in cui compaiono nell'input). Se $K = 0$, lasciare una riga vuota;

Output (parte facoltativa)

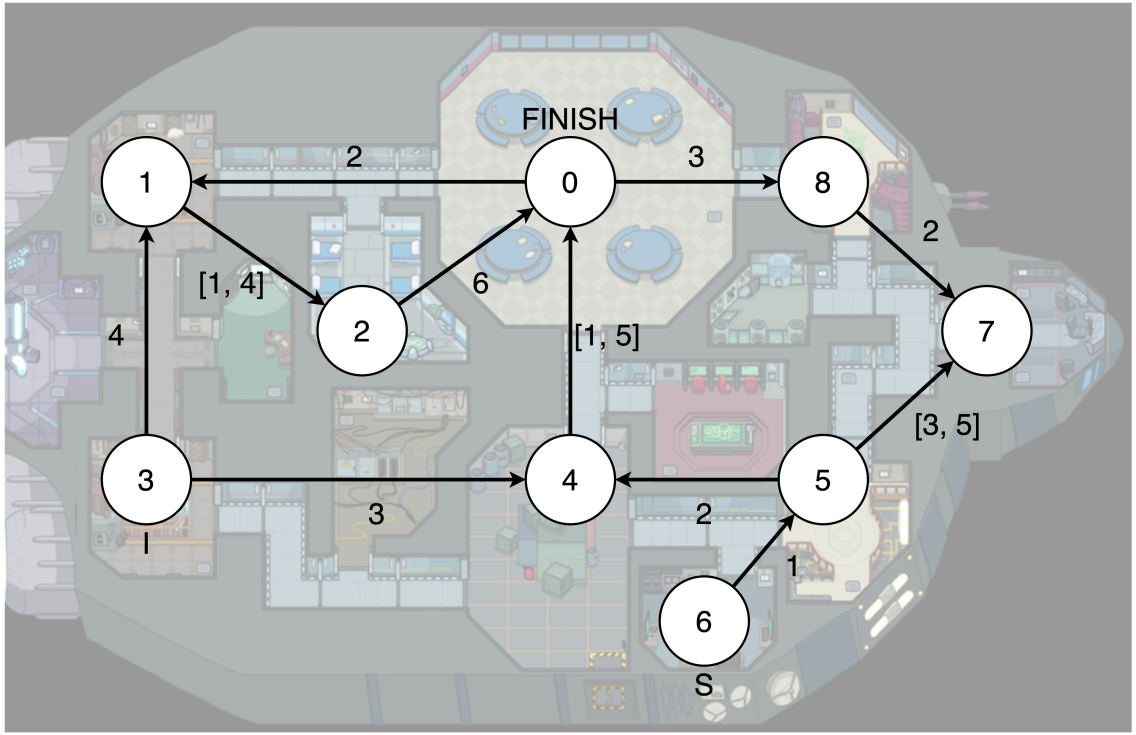
Opzionalmente, l'output può contenere anche le seguenti due righe:

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

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

Esempio

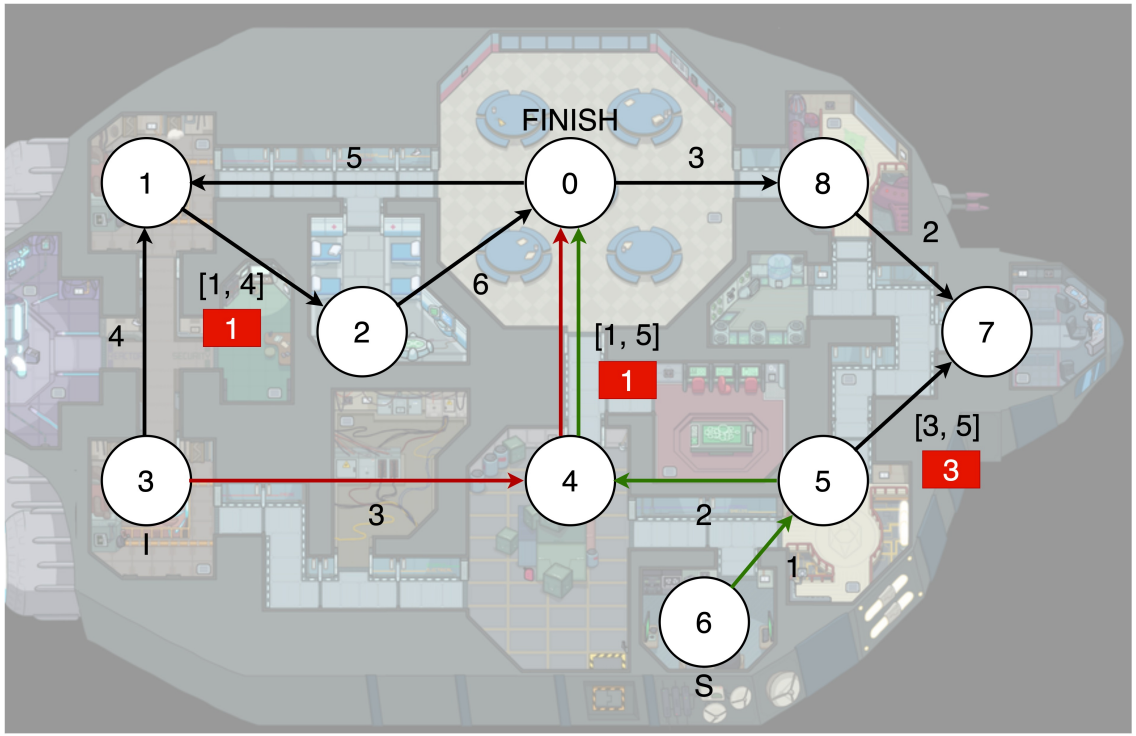
- 9 stanze e 11 corridoi (di cui 8 semplici e 3 con ventole)
- I corridoi semplici hanno un tempo di percorrenza fissato
- I corridoi con ventole hanno un range di tempi di percorrenza possibili
- Montresus si trova inizialmente nella stanza al nodo 3, gli studenti al nodo 6
- Il FabLab è al nodo 0



Esempio - Pareggio

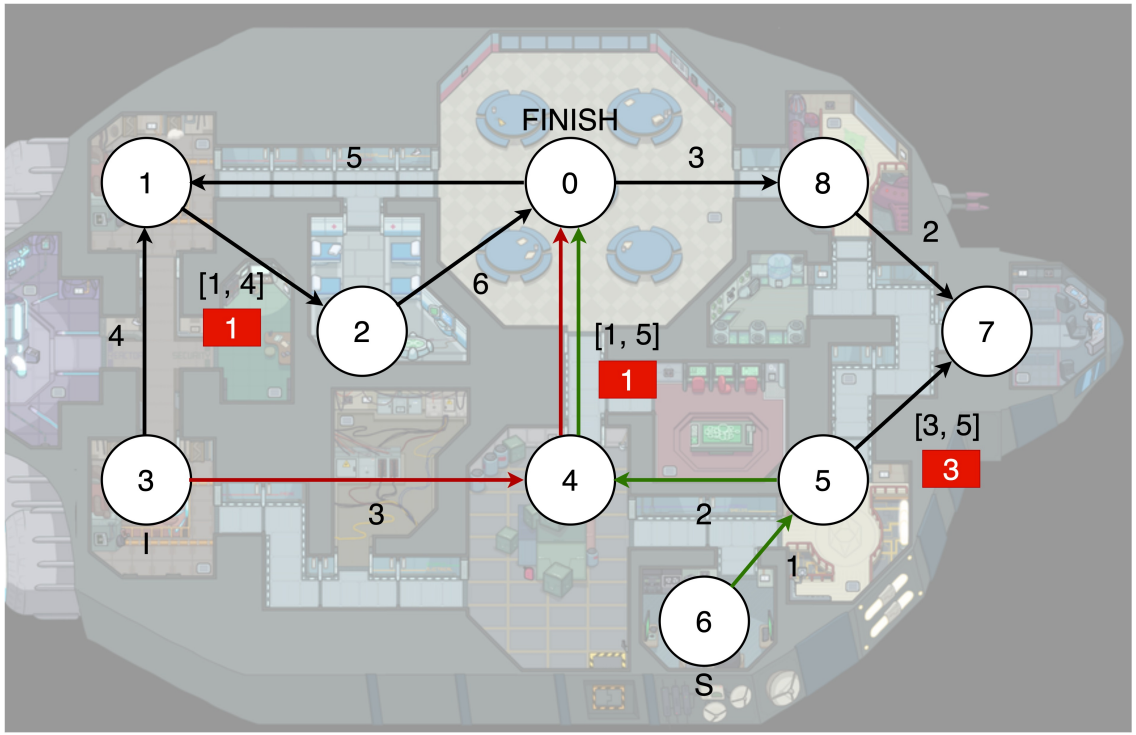
In questo caso, gli studenti possono al massimo pareggiare.

- 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.



Esempio - Pareggio

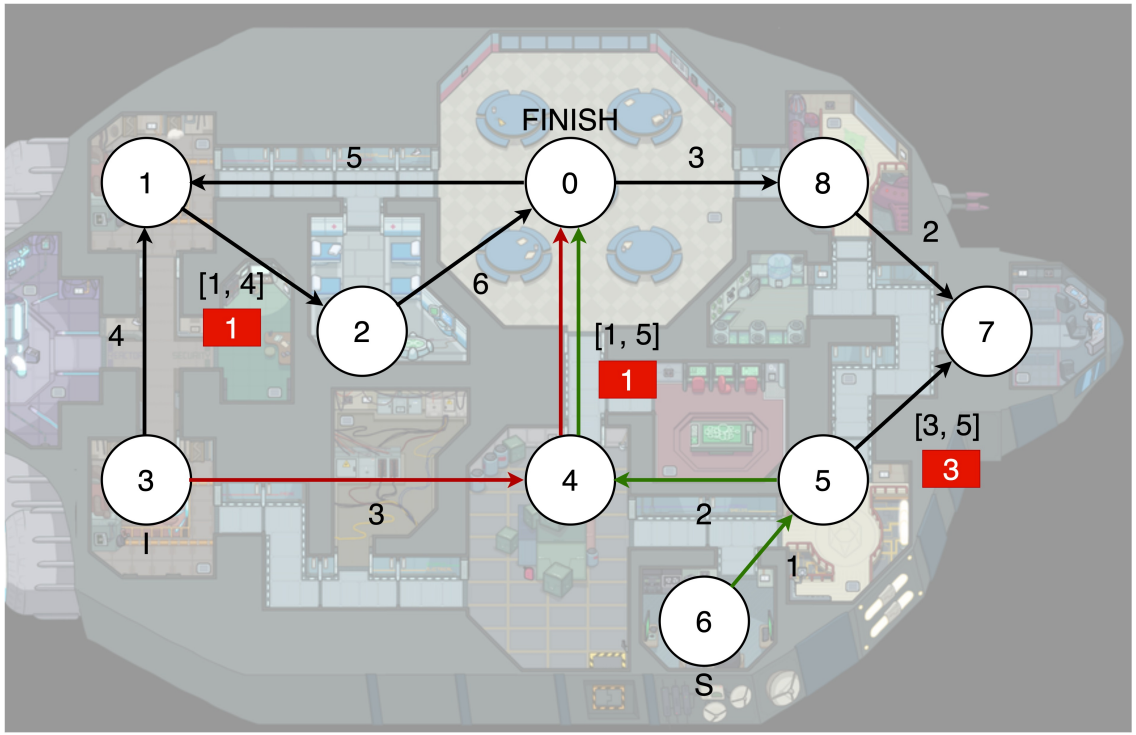
- Anche Montresus non può far meglio che raggiungere F in 4 unità di tempo (percorso: $[3, 4, 0]$), pareggiando.
- Montresus ha anche un percorso alternativo per arrivare ad F: 3, 1, 2, 0. Tuttavia, riesce a percorrerlo in 11 unità di tempo (notare che attraversa un corridoio manipolato in modo da essere percorribile il più velocemente possibile).



Esempio - Pareggio: output

In questo caso l'output valido sarebbe il seguente:

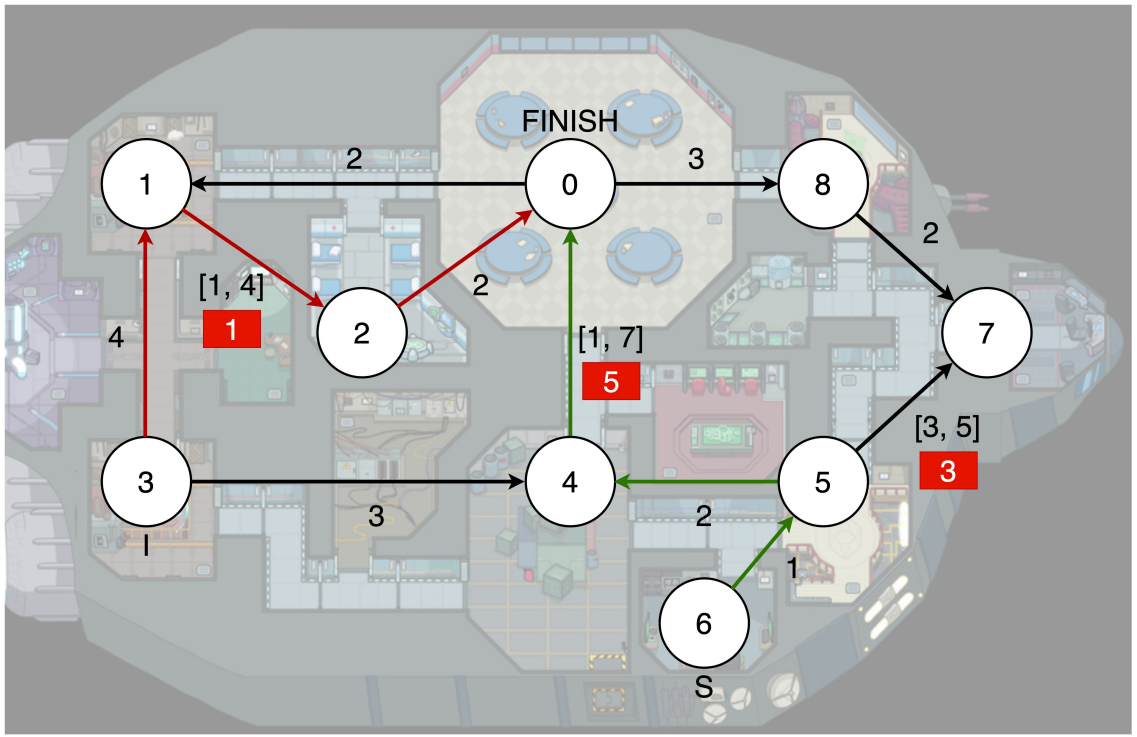
```
0
4 4
1 1 3
3
3 4 0
```



Esempio - Montresus vince

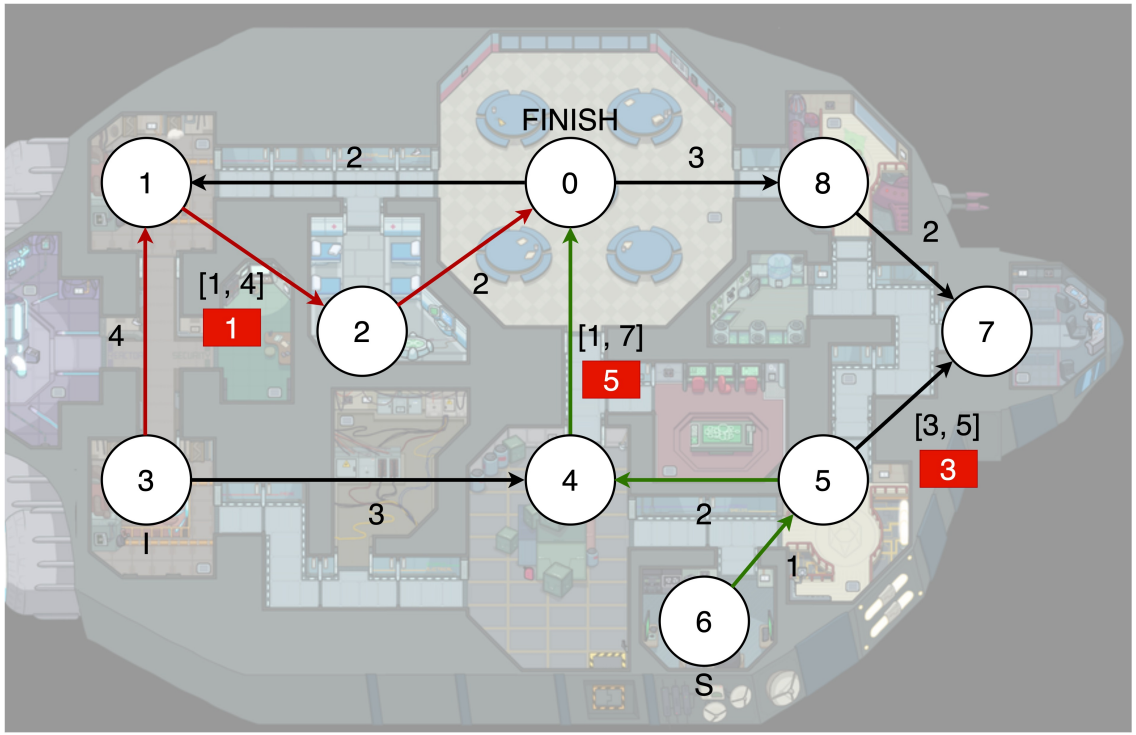
In questo caso, gli studenti perdono sempre.

- Il loro percorso più breve (ma 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.



Esempio - Montresus vince

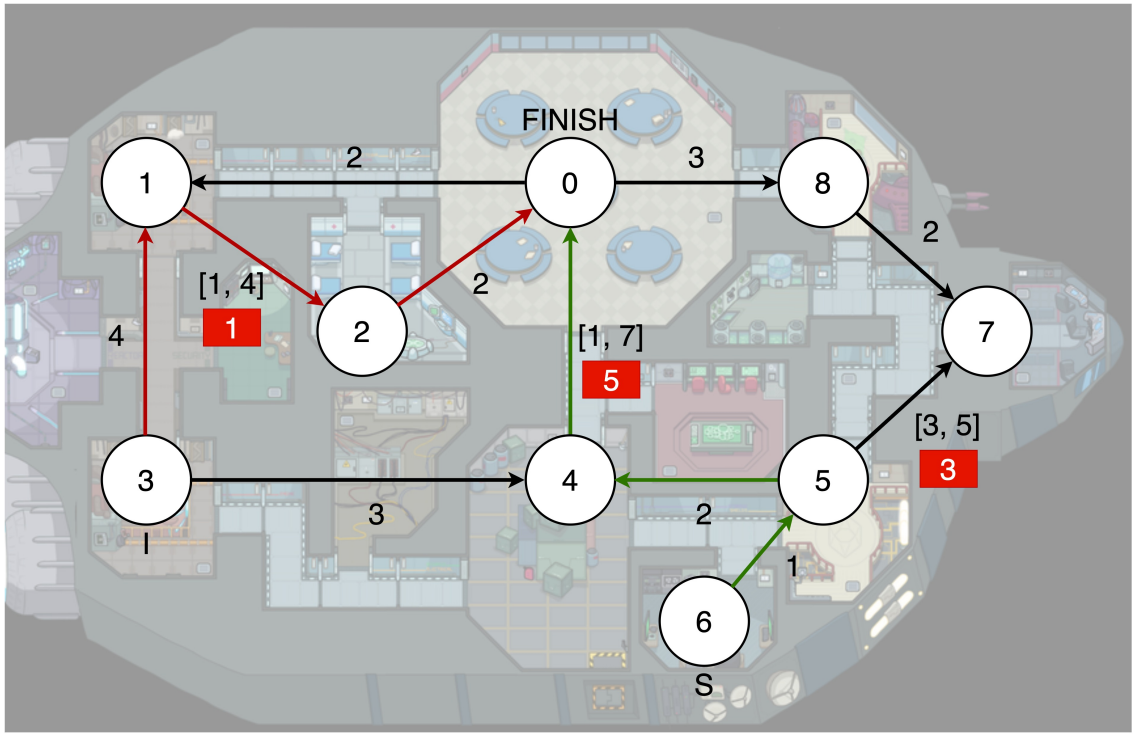
- L'impostore può manipolare i percorsi a propio favore, Montresus può sfruttare il percorso 3, 1, 2, 0 per arrivare ad *F* in 7 unità di tempo, arrivando quindi prima degli studenti.



Esempio - Montresus vince: output

In questo caso l'output valido sarebbe il seguente:

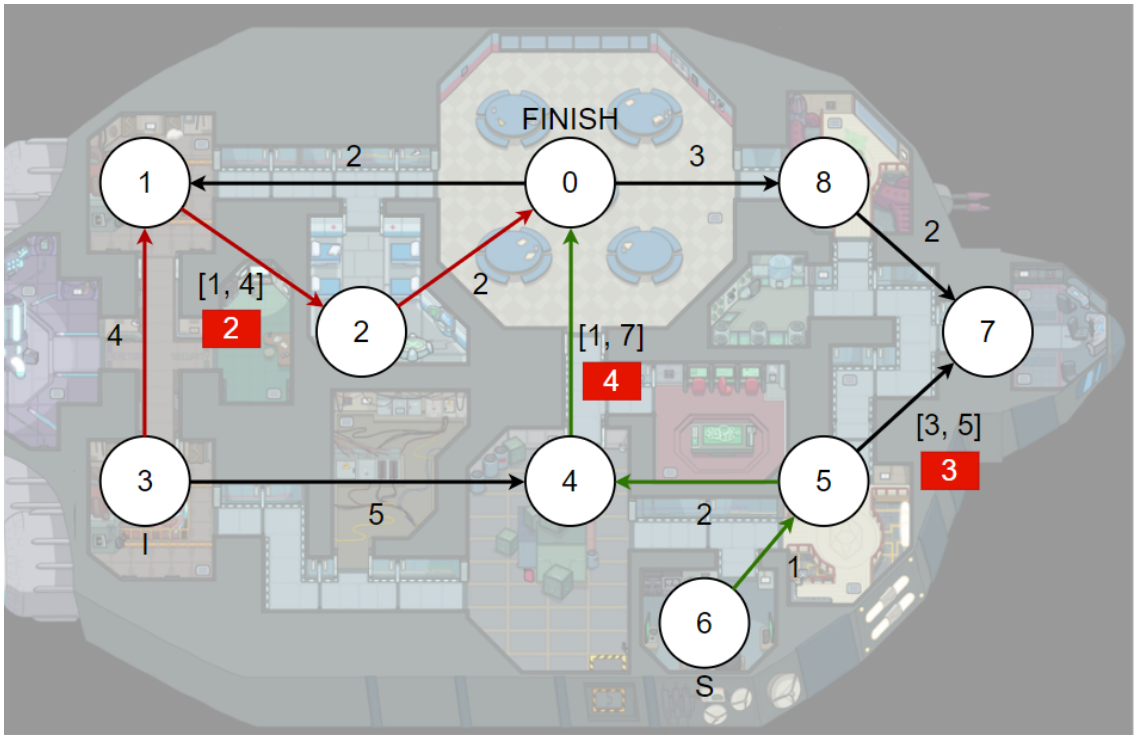
```
1
7 8
1 5 3
4
3 1 2 0
```



Esempio - Montresus vince: wrong answer

Se i pesi fossero stati assegnati secondo questo output

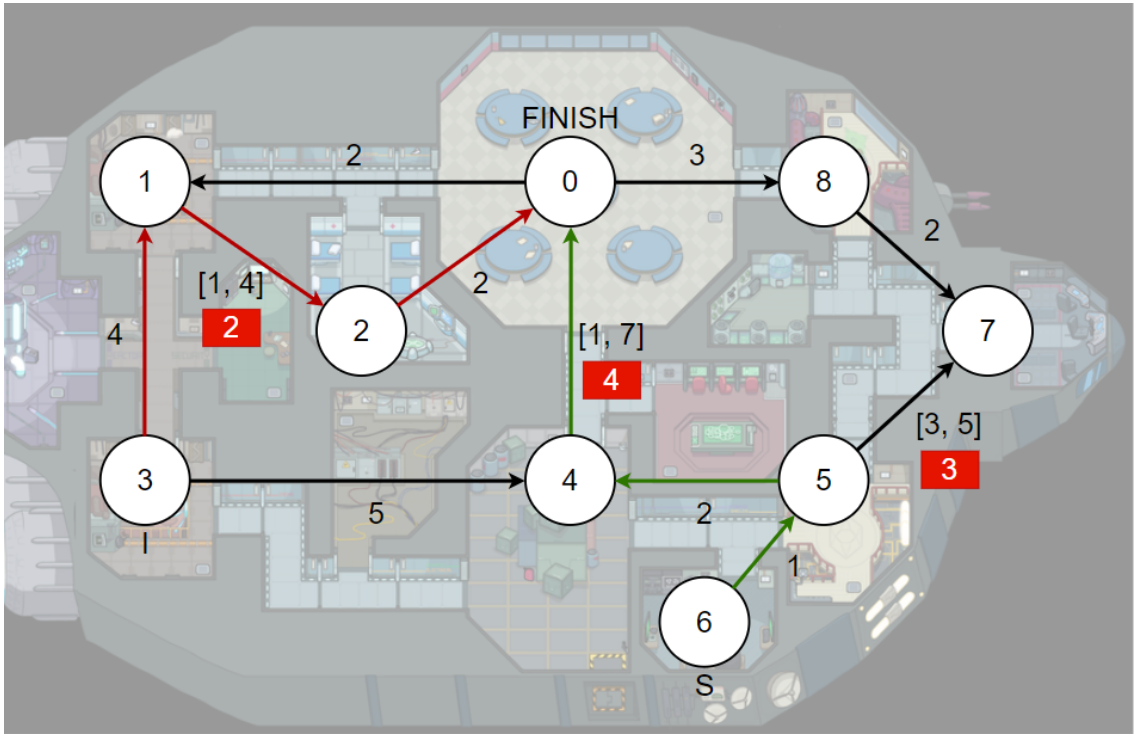
```
2
8 7
2 4 3
4
3 1 2 0
```



avrebbero vinto gli studenti.

Esempio - Montresus vince: wrong answer

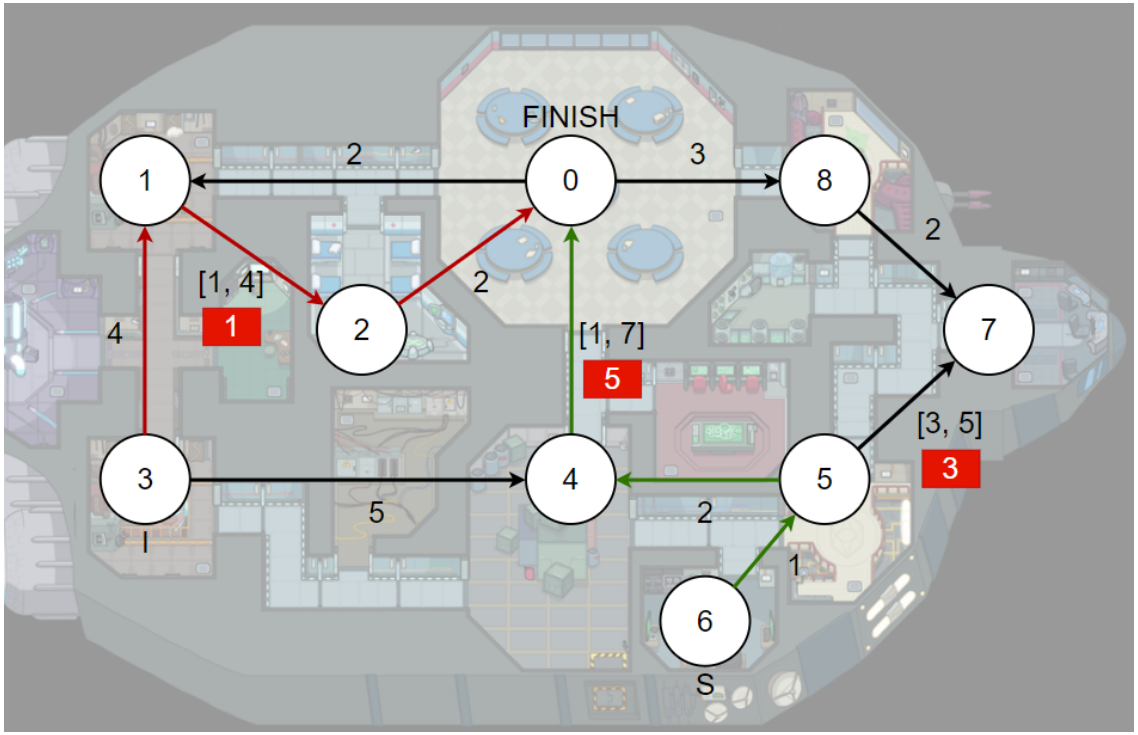
Tuttavia, esiste un'assegnazione di pesi che permette all'impostore di vincere, quindi l'output precedente non è corretto.



Esempio - Montresus vince: wrong answer

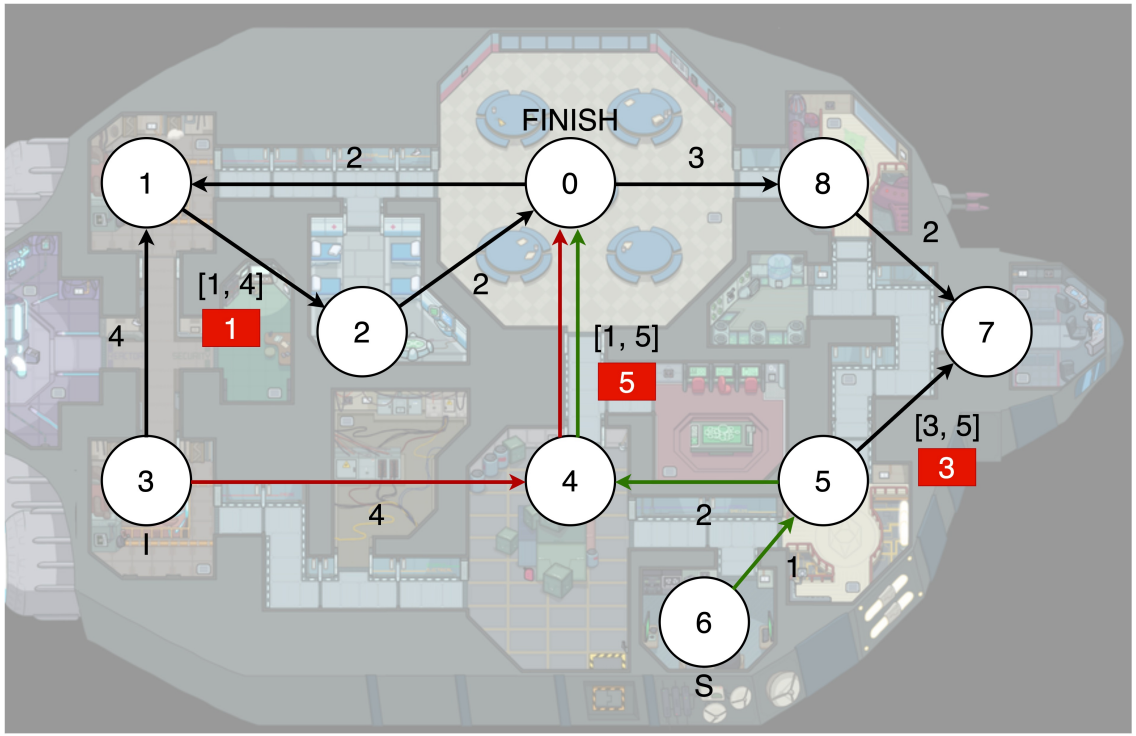
Questo è un esempio di output corretto

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



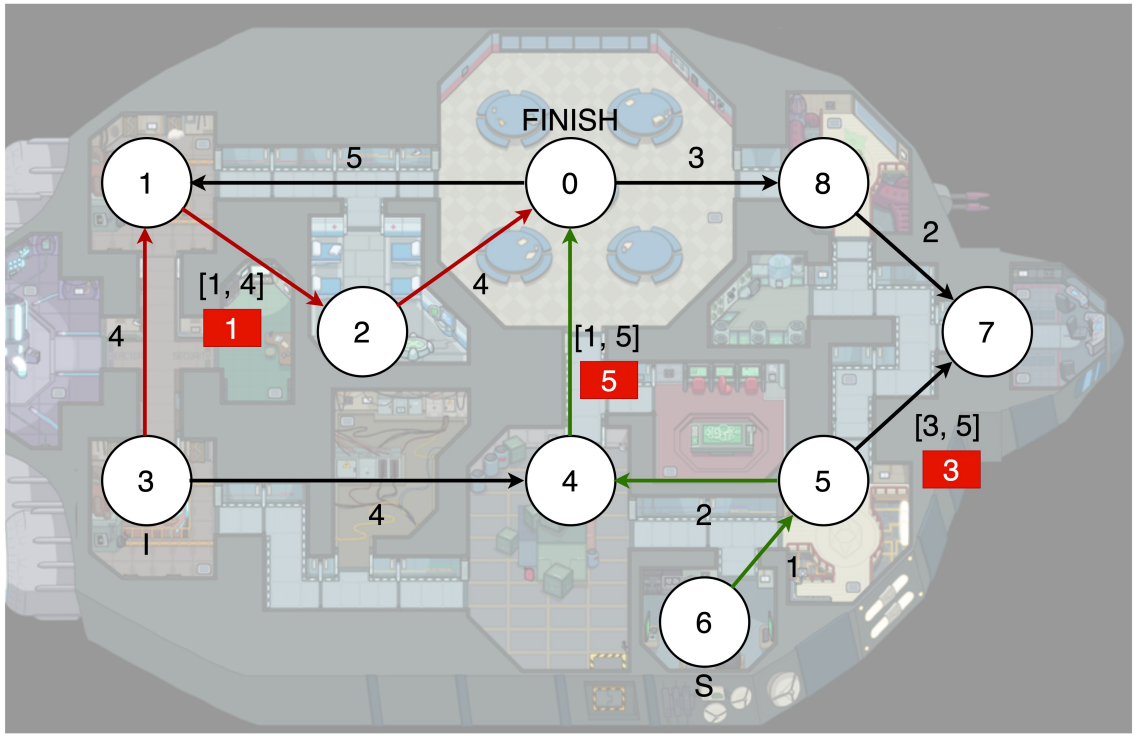
Esempio - Studenti vincono: wrong answer

- Con questa scelta di percorsi, gli studenti arriverebbero ad *F* prima di Montresus.
- Tuttavia, Montresus non starebbe seguendo il percorso ottimale per arrivare in *F*. Infatti, il percorso 3, 4, 0 viene percorso in 9 unità di tempo, ma esiste il percorso 3, 1, 2, 0 che lo è in 7.
- La soluzione riportata in figura risulta quindi errata.



Esempio - Studenti vincono: risposta esatta

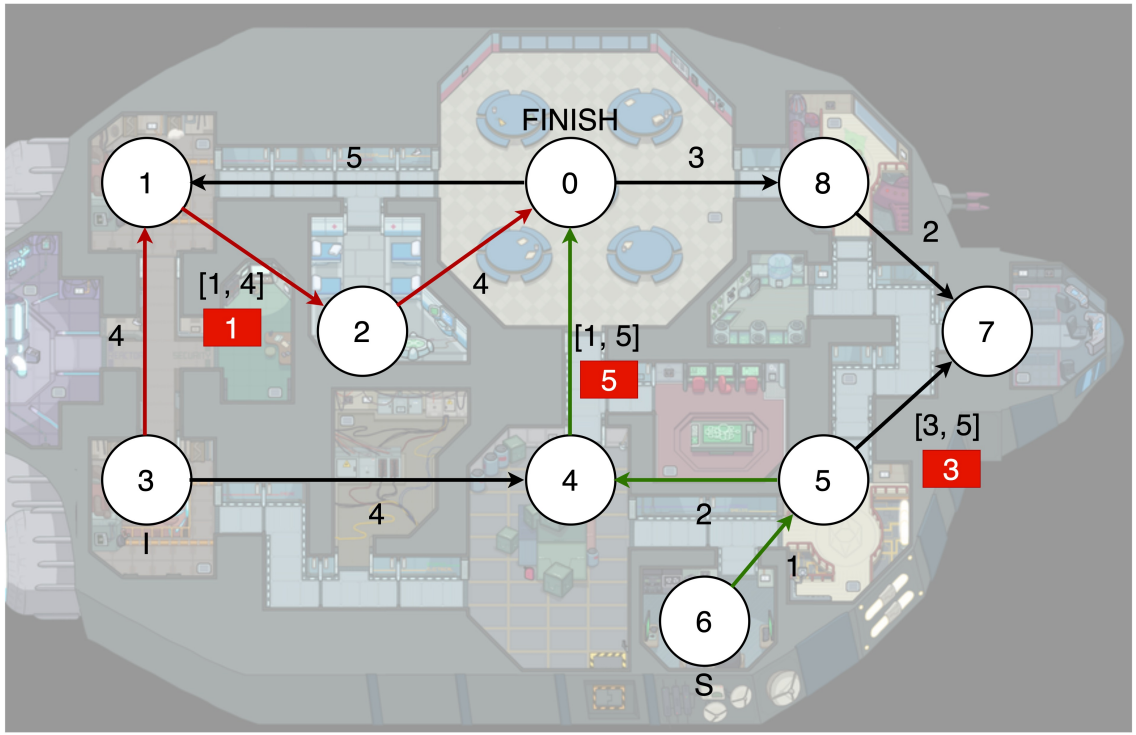
- Ora invece il percorso migliore che Montresus possa percorrere per arrivare in *F* impiega 9 unità di tempo.
- Gli studenti, nonostante debbano percorrere un corridoio manipolato, hanno a disposizione un percorso percorribile in 8 unità di tempo. Ovvero arrivano al Fablab sempre più velocemente di Montresus.



Esempio - Studenti vincono: output

In questo caso l'output valido sarebbe il seguente:

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



Note su Input

Assunzioni generali

- $1 \leq N \leq 10.000$
- $1 \leq M \leq 10.000$
- $0 \leq K \leq 1.000$
- $1 \leq T, T_{min}, T_{max} \leq 10.000$
- 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, $K \geq 2$.

Per la sufficienza si possono risolvere i 6 casi di difficoltà ★☆☆☆.

Limiti di tempo e memoria

I limiti di tempo e memoria sono:

- Tempo limite massimo: 1 secondo.
- Memoria massima: 8 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 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.

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 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.

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: mercoledì 21 dicembre 2022 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

- giovedì 15 dicembre 2022 dalle 13:30 alle 15:30 (A101);
- venerdì 16 dicembre 2022 dalle 16:30 alle 18:30 (online);
- sabato 17 dicembre 2022 dalle 11:00 alle 12:00 (online);
- lunedì 19 dicembre 2022 dalle 16:30 alle 18:30 (online);
- martedì 20 dicembre 2022 dalle 10.30 alle 12.30 (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.