Il crescione perfetto (`cesena` [72 punti])

Questo problema è preso dalla giornata dell'Informatica - Cesena 2024 (problema `cesena`)

Si consideri un grafo non orientato con:

- $N \in \mathbb{N}$: numero di nodi, indicizzati da 0 a $N-1$
- $M \in \mathbb{N}$: numero di archi
- $K \in \mathbb{N}$: numero di nodi target $\{s_0, s_1,, s_{K-1}\} \subseteq \{0, ..., N-1\}$
- Un array $A \in \{0, ..., N-1\}^M$ e un array $B \in \{0, ..., N-1\}^M$ tali che per ogni $i \in \{0, ..., M-1\}$, l'arco $\{A_i, B_i\}$ collega due nodi distinti del grafo
- Un array $T \in \{1, ..., K\}^M$ tale che l'arco $\{A_i, B_i\}$ viene rimosso non appena sono stati accesi esattamente T_i nodi target distinti

Obiettivo

Devi determinare il massimo numero $R \in \mathbb{N}$ di nodi target che è possibile accendere applicando il seguente processo fino a quando i nodi target attualmente raggiungibili dal nodo 0 sono tutti già accesi:

> nel grafo corrente si percorre un cammino dal nodo 0 ad un qualche nodo target non ancora acceso e lo si accende. Fatto ciò si ritorna al nodo 0 e si rimuovono dal grafo corrente tutti quegli archi $\{A_i, B_i\}$ per cui T_i eguaglia il numero attuale di nodi già accesi

Input

L'input va letto da `stdin` nel seguente formato:

- Una riga con tre interi N, M, K
- Una riga con K interi $S_0, ..., S_{\{K-1\}}$
- M righe contenenti tre interi A_i, B_i, T_i per ogni $i \in [0, M)$

Output

Per ciascuna istanza, prima di leggere l'istanza successiva, scrivi su `stdout` il tuo output così strutturato:

[**goal 1 (opt_val)**]: la prima riga contiene un intero: il massimo numero di nodi target distinti che è possibile visitare.

[**goal 2 (opt_target_seq)**]: la seconda riga riporta gli `opt_val` nodi target che accenderesti, nell'ordine di accensione.

[**goal 3 (opt_paths)**]: le seguenti `opt_val` righe riportano, nello stesso ordine, gli `opt_val` cammini utilizzati. Ciascun cammino è specificato fornendo la sequenza di nodi che visita, partendo dal nodo 0 e terminando col nodo che accende.

Assunzioni

- $1 \leq K \leq N \leq 10^5$
- $1 \leq M \leq 2 \cdot 10^5$
- $0 \leq s_i < N$, tutti distinti
- $0 \leq A_i, B_i < N$, $A_i \neq B_i$
- $1 \leq T_i \leq K$
- Il grafo iniziale è connesso e non contiene archi paralleli
- Gli estremi di ciascun arco sono nodi distinti

Esempio di Input/Output

```
<start in>
2
5 6 3
1 4 2
0 3 3
3 4 2
2 3 1
1 4 1
0 4 1
4 2 2
<more>
```

```
7 10 5
5 4 0 6 3
5 2 1
1 6 2
1 5 1
6 2 2
3 5 2
3 6 1
0 2 3
1 3 2
0 4 1
1 0 1

 <end>
```

```
<start out>
2
1 2
0 3 4 1
0 3 4 2
3
4 3 0
0 4
0 2 6 1 3
0

 <end>
```

Spiegazione:

Nel primo caso d'esempio la mappa della città è la seguente:

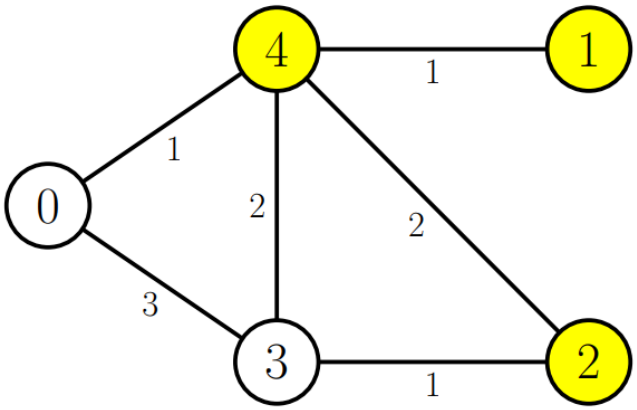

Figure 1: Mappa della citta coi chioschi che vendono piadine.

ed è possibile mangiarsi prima una piadina al chiosco nella piazza 1 raggiungibile col cammino 0-3-4-1 e poi, benchè appesantiti, andare a mangiarsi la piadina all'incrocio 2 raggiungibile col cammino 0-3-4-2.

Subtask

Il tempo limite per istanza è di 1 secondo.

1. **[0 pts← 2 istanze da 0 + 0 + 0 punti] `esempi_testo`:** I due esempi del testo
2. **[12 pts← 4 istanze da 1 + 1 + 1 punti] `tiny`:** $N \leq 10, M \leq 20, K \leq 10$
3. **[12 pts← 4 istanze da 1 + 1 + 1 punti] `small`:** $N \leq 20, M \leq 30, K \leq 20$
4. **[12 pts← 4 istanze da 1 + 1 + 1 punti] `path`:** $N \leq 3000, M = N - 1, \{A_i, B_i\} = \{i, i+1\}$ per $i \in [0, M)$ (il grafo è un cammino)
5. **[12 pts← 4 istanze da 1 + 1 + 1 punti] `static`:** $N \leq 3000, M \leq 7000, T_i = K$ (gli archi non scadranno mai)
6. **[12 pts← 4 istanze da 1 + 1 + 1 punti] `medium`:** $N \leq 3000, M \leq 7000$
7. **[12 pts← 4 istanze da 1 + 1 + 1 punti] `big`:** Nessuna limitazione specifica

In generale, quando si richiede la valutazione di un subtask vengono valutati anche i subtask che li precedono, ma si evita di avventurarsi in subtask successivi fuori dalla portata del tuo programma

che potrebbe andare in crash o comportare tempi lunghi per ottenere la valutazione completa della sottomissione. Ad esempio, chiamando[1, 2] :

```
rtal -s <URL> connect -a size=small cesena -- <MY_SOLUTION>
```

vengono valutati, nell'ordine, i subtask:

`esempi_testo`, `tiny`, `small`.

Il valore di default per l'argomento `size` è `big` che include tutti i testcase.

Nel template di riga di comando quì sopra `<MY_SOLUTION>` è una qualsiasi scrittura che ove immessa anche da sola al prompt della CLI comporti l'avvio del solver da tè realizzato. Solo alcuni esempi:

- `./a.out` per un compilato da `C/c++`, eventualmente seguito dagli argomenti che prevede
- `./my_solution.py arg1 arg2 ...` se il tuo file `my_solution.py` col codice python ha i permessi di esecuzione e inizia con la riga di shebang
- `python my_solution.py` o `python3 my_solution.py` per far eseguire il tuo script da un interprete python

Se vuoi che una tua sottomissione venga conteggiata ai fini di un esame o homework devi allegare il sorgente della tua soluzione con `-fsource=<FILENAME>` (ad esempio `-fsource=my_solution.py` subito a valle del comando `connect`. Inoltre devi aver precedentemente affettuato il login tramite credenziali GIA (lancia prima `rtal -s <URL> login` e segui le istruzioni per impostare il sign-on).

Il comando `rtal` prevede diversi parametri, consigliamo di esplorarne e sperimentarne le potenzialità ed opzioni d'uso. Un tutorial all'uso di `rtal` è esposto alla pagina:

`https://github.com/romeorizzi/AlgoritmiUniVR/tree/main/strumenti`

[1] `<URL>` server esame: `wss://ta.di.univr.it/esame`

[2] `<URL>` server esercitazioni e simula-prove: `wss://ta.di.univr.it/algo2025`