

La città incantata

Il sindaco di Tashkent vuole costruire il parco dei divertimenti "La città incantata".

Dato un intero K devi progettare un parco come segue:

- Scegli un intero N , il numero di attrazioni, e numera le attrazioni da 0 a $N - 1$.
- Aggiungi percorsi bidirezionali tra coppie di attrazioni **distinte**. È possibile aggiungere più percorsi tra la stessa coppia di attrazioni. Nota che **non è richiesto che sia possibile raggiungere ogni attrazione da ogni altra usando i percorsi**.
- A ogni attrazione i assegna un tipo $T[i]$. I tipi sono $2K$, numerati da 0 a $2K - 1$. Ci deve essere almeno un'attrazione per ogni tipo e potrebbero esserci più attrazioni dello stesso tipo.

Analisi di mercato hanno rivelato che:

1. ai visitatori piace visitare tre attrazioni senza che due consecutive siano dello stesso tipo;
2. ai visitatori non piace se da una singola attrazione partono troppi percorsi.

Chiamiamo una tripla ordinata di **tipi** (t_1, t_2, t_3) (con $0 \leq t_1, t_2, t_3 < 2K$) **interessante** se $t_1 \neq t_2$ e $t_2 \neq t_3$. Nota che potrebbe essere che $t_1 = t_3$, e quindi ci sono $2K \cdot (2K - 1)^2$ triple interessanti distinte.

Per soddisfare ogni potenziale visitatore, il sindaco impone due condizioni per il tuo progetto:

Condizione 1: per ogni tripla interessante t_1, t_2, t_3 , devono esistere tre attrazioni a_1, a_2 e a_3 (con $0 \leq a_1, a_2, a_3 < N$) tali che:

- i tipi di a_1, a_2 e a_3 siano rispettivamente t_1, t_2 e t_3 : $T[a_1] = t_1, T[a_2] = t_2$ e $T[a_3] = t_3$.
- ci sia un percorso tra a_1 e a_2 e un percorso tra a_2 e a_3 .

Non importa se ci sia o meno un percorso tra a_1 e a_3 . Nota inoltre che se $t_1 = t_3$ allora è possibile che $a_1 = a_3$.

Condizione 2: da ogni attrazione possono partire **al più** K percorsi.

Il problema consiste di 50 subtask **output-only** e a **punteggio parziale**. Ogni subtask corrisponde a un valore specifico di K , per cui dovrai progettare un parco che soddisfi le due condizioni precedenti per quel valore di K . Il tuo punteggio dipende dal numero di attrazioni: una soluzione con meno attrazioni varrà almeno tanti punti quanti una con più attrazioni.

Implementazione

Ci sono due modi per sottoporre la tua soluzione, e puoi scegliere indipendentemente il modo per ciascun subtask:

- **chiamata a funzione;**
- **file di output.**

Per sottoporre una soluzione tramite **sorgente**, devi implementare la seguente funzione:

```
std::pair<std::vector<int>, std::vector<std::pair<int, int>>>>  
construct(int K)
```

- K : metà del numero di tipi di attrazioni, nonché il massimo numero di percorsi che può partire da una singola attrazione.
- La funzione viene chiamata esattamente una volta per subtask.

La funzione deve restituire una coppia (T, E) che descriva il parco. Detto M il numero di percorsi:

- T : un array lungo N contenente i tipi delle attrazioni.
- E : un array lungo M che descriva i percorsi: per ogni $0 \leq j < M$, $E[j] = (U[j], V[j])$ rappresenta un percorso tra $U[j]$ e $V[j]$.

Per sottoporre una soluzione tramite **file di output**, crea e sottoponi un file di testo nel seguente formato:

```
N M  
T[0] T[1] ... T[N-1]  
U[0] V[0]  
U[1] V[1]  
...  
U[M-1] V[M-1]
```

Nota che una soluzione deve soddisfare quanto segue per essere **valida**:

- $N \leq 2000$;
- $0 \leq T[i] < 2K$ per ogni $0 \leq i < N$, e deve esserci almeno un'attrazione per tipo;
- $0 \leq U[j], V[j] < N$ e $U[j] \neq V[j]$ per ogni $0 \leq j < M$;
- Le condizioni 1 e 2 devono essere soddisfatte.

Assunzioni

- $1 \leq K \leq 50$.

Punteggio

Ci sono 50 subtask, uno per ogni intero K tra 1 e 50. Nel subtask $1 \leq i \leq 50$, vale che $K = i$.

Ogni subtask ha un punteggio S e un numero obiettivo P di attrazioni, secondo la tabella seguente:

Subtask	S	P
1	1	2
2	8	12
3	9	24
4	9	40
5	9	50
6 - 10	4	$12 \cdot K$
11 - 12	3	$12 \cdot K$
13 - 50	1	$12 \cdot K$

In ogni subtask, se la tua soluzione non descrive un parco valido, otterrai 0 punti e il verdetto `Output isn't correct` su CMS. Altrimenti, il tuo punteggio viene calcolato in base di N e dei parametri S e P come segue:

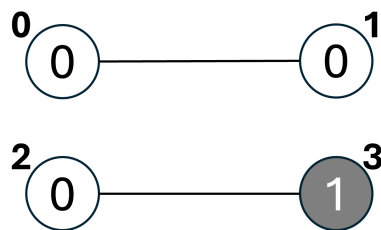
Condizione	Punteggio
$N \leq P$	S
$P < N \leq 2P$	$(0.4 + 0.3 \cdot \frac{2P-N}{P}) \cdot S$
$2P < N \leq 2000$	$(0.1 + 0.3 \cdot \frac{2P}{N}) \cdot S$

Esempio

Considera la chiamata:

```
construct(1)
```

Qui $K = 1$ e ci sono pertanto $2K = 2$ tipi di attrazioni. La figura illustra una soluzione valida con $N = 4$ attrazioni e $M = 2$ percorsi. Le attrazioni 0, 1 e 2 sono di tipo 0, e l'attrazione 3 è di tipo 1.



Ci sono due triple interessanti:

- per la tripla $(0, 1, 0)$, possiamo scegliere $(a_1, a_2, a_3) = (2, 3, 2)$;
- per la tripla $(1, 0, 1)$, possiamo scegliere $(a_1, a_2, a_3) = (3, 2, 3)$.

Pertanto la condizione 1 è soddisfatta.

La procedura può restituire la coppia $([0, 0, 0, 1], [(0, 1), (2, 3)])$. Nota che questa potrebbe non essere la soluzione ottimale per $K = 1$.

Grader di esempio

Formato di input:

```
K
```

Formato di output:

```
N M
T[0] T[1] ... T[N-1]
U[0] V[0]
U[1] V[1]
...
U[M-1] V[M-1]
```

Nota che il formato di output del grader di esempio coincide con il formato per la sottoposizione tramite file di output.