

Kota Ajaib

Walikota Tashkent ingin merancang taman hiburan Kota Ajaib yang terkenal. Anda diberikan sebuah bilangan bulat positif K , dan tugas Anda adalah merancang sebuah taman sebagai berikut:

- Pilih suatu bilangan bulat N yang menyatakan banyaknya atraksi. Setiap atraksi dinomori dengan bilangan bulat dari 0 hingga $N - 1$.
- Tambahkan jalan dua arah antara beberapa pasang atraksi **berbeda**. Boleh saja terdapat lebih dari satu jalan yang menghubungkan sepasang atraksi. **Tidak diperlukan bahwa antara setiap pasang atraksi, terdapat jalur yang menghubungkan keduanya menggunakan jalan-jalan yang telah dibangun.**
- Untuk setiap i sehingga $0 \leq i < N$, tentukan jenis atraksi $T[i]$ pada atraksi i . Ada $2K$ jenis atraksi, dinomori dari 0 hingga $2K - 1$. Setiap jenis harus digunakan oleh setidaknya satu buah atraksi. Atraksi yang berbeda boleh memiliki jenis yang sama.

Riset pasar menunjukkan bahwa:

1. Setiap pengunjung cenderung ingin mengunjungi tiga atraksi, tanpa mengunjungi dua atraksi sejenis sama secara berurutan.
2. Pengunjung tidak suka atraksi yang terhubung dengan terlalu banyak jalan.

Untuk memuaskan **semua** calon pengunjung, sang walikota menetapkan dua syarat pada rancangan Anda.

Kita sebut sebuah tripel terurut dari **jenis-jenis** (t_1, t_2, t_3) untuk $0 \leq t_1, t_2, t_3 < 2K$ **menarik** jika $t_1 \neq t_2$ dan $t_2 \neq t_3$. Perhatikan bahwa t_1 mungkin sama dengan t_3 . Jadi, ada $2K \cdot (2K - 1)^2$ tripel menarik.

Syarat 1: Untuk setiap tripel menarik (t_1, t_2, t_3) , harus ada tiga atraksi a_1, a_2, a_3 (dengan $0 \leq a_1, a_2, a_3 < N$) sedemikian hingga:

- Jenis-jenis dari a_1, a_2, a_3 sesuai dengan t_1, t_2, t_3 . Artinya, $T[a_1] = t_1$, $T[a_2] = t_2$, dan $T[a_3] = t_3$.
- Ada jalan yang menghubungkan a_1 dan a_2 .
- Ada jalan yang menghubungkan a_2 dan a_3 .

Ada atau tidak adanya jalan antara a_1 dan a_3 tidak penting. Perhatikan juga bahwa ketika $t_1 = t_3$, atraksi a_1 dan a_3 bisa saja sama.

Syarat 2: Setiap atraksi terhubung ke **paling banyak** K jalan.

Soal ini terdiri dari 50 subsoal **output-only** dengan **penilaian parsial**. Setiap subsoal memiliki nilai K tertentu, dan Anda harus merancang taman yang memenuhi semua syarat di atas untuk nilai K tersebut. Nilai Anda bergantung pada banyaknya atraksi pada solusi Anda—solusi dengan lebih sedikit atraksi memperoleh nilai yang sama atau lebih tinggi.

Detail Implementasi

Ada dua cara untuk mengumpulkan solusi Anda, dan Anda boleh menggunakan yang mana saja untuk setiap subsoal:

- **Pemanggilan prosedur**
- **Berkas keluaran**

Untuk mengumpulkan solusi Anda menggunakan **pemanggilan prosedur**, Anda harus mengimplementasikan prosedur berikut:

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

- K : setengah dari banyaknya jenis atraksi, dan juga banyaknya jalan maksimum yang boleh terhubung ke setiap atraksi.
- Prosedur ini dipanggil tepat sekali untuk setiap subsoal.

Prosedur ini harus mengembalikan sebuah pasangan (T, E) yang menunjukkan sebuah taman. Misalkan M adalah banyaknya jalan di taman.

- T : array sepanjang N yang menyatakan jenis dari semua atraksi.
- E : array sepanjang M yang menyatakan jalan-jalan. Untuk setiap $0 \leq j < M$, $E[j] = (U[j], V[j])$ menyatakan sebuah jalan dua arah antara atraksi berbeda $U[j]$ dan $V[j]$.

Untuk mengumpulkan solusi melalui **berkas keluaran**, buat dan kumpulkan sebuah berkas teks dalam format berikut:

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

Perhatikan bahwa solusi Anda harus memenuhi batasan-batasan berikut agar dianggap **valid**:

- $N \leq 2000$
- $0 \leq T[i] < 2K$ untuk setiap $0 \leq i < N$, dan setiap jenis harus ditetapkan pada setidaknya satu atraksi.
- $0 \leq U[j], V[j] < N$ dan $U[j] \neq V[j]$ untuk setiap $0 \leq j < M$.
- Syarat 1 dan 2 harus terpenuhi.

Batasan

- $1 \leq K \leq 50$

Penilaian

Ada 50 subsoal, satu untuk setiap bilangan bulat K dari 1 hingga 50. Untuk subsoal i ($1 \leq i \leq 50$), nilai dari K adalah i .

Setiap subsoal memiliki nilai S dan target banyaknya atraksi P sesuai dengan tabel berikut.

Subsoal	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	$12 \cdot K$

Untuk setiap subsoal, apabila solusi Anda bukan merupakan taman hiburan yang valid, maka nilai solusi Anda adalah 0 (dilaporkan sebagai `Output isn't correct` pada CMS).

Selebihnya, nilai Anda akan dihitung berdasarkan N dan parameter-parameter S dan P sebagai berikut:

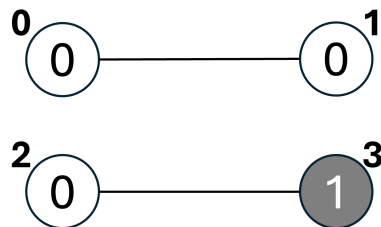
Syarat	Nilai
$N \leq P$	S
$P < N \leq 2P$	$(0.4 + 0.3 \cdot \frac{2P-N}{P}) \cdot S$
$2P < N \leq 2000$	$(0.1 + 0.3 \cdot \frac{2P}{N}) \cdot S$

Contoh

Perhatikan pemanggilan berikut.

```
construct(1)
```

Pada contoh ini, $K = 1$, sehingga ada $2K = 2$ jenis atraksi. Gambar di bawah ini menunjukkan solusi yang valid dengan $N = 4$ atraksi dan $M = 2$ jalan. Atraksi 0, 1, 2 berjenis 0, dan atraksi 3 berjenis 1.



Ada dua tripel menarik:

- Untuk tripel $(0, 1, 0)$, kita dapat memilih $(a_1, a_2, a_3) = (2, 3, 2)$.
- Untuk tripel $(1, 0, 1)$, kita dapat memilih $(a_1, a_2, a_3) = (3, 2, 3)$.

Ini berarti Syarat 1 terpenuhi.

Prosedur ini dapat mengembalikan pasangan $([0, 0, 0, 1], [(0, 1), (2, 3)])$. Perhatikan bahwa solusi yang diberikan pada contoh ini mungkin saja tidak optimal untuk $K = 1$.

Contoh Grader

Format Masukan:

```
K
```

Format Keluaran:

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

Perhatikan bahwa keluaran dari contoh *grader* sesuai dengan format yang diperlukan untuk berkas keluaran.