

Magic City

Der Bürgermeister von Taschkent möchte den berühmten Vergnügungspark Magic City neu gestalten. Gegeben ist eine positive ganze Zahl K . Deine Aufgabe ist es, einen Park wie folgt zu entwerfen:

- Wähle eine ganze Zahl N , die Anzahl der Attraktionen. Die Attraktionen sind mit den ganzen Zahlen von 0 bis $N - 1$ beschriftet.
- Füge bidirektionale Gehwege zwischen jeweils zwei **unterschiedlichen** Attraktionen hinzu. Es ist erlaubt, mehr als einen Gehweg zwischen einem Paar von Attraktionen zu haben. **Es ist nicht nötig, dass man von jeder Attraktion über die Gehwege zu jeder anderen gelangen kann.**
- Für jedes i mit $0 \leq i < N$ weise der Attraktion i einen Typ $T[i]$ zu. Es gibt $2K$ Typen, nummeriert von 0 bis $2K - 1$. Jeder Typ muss mindestens einer Attraktion zugeordnet werden. Verschiedene Attraktionen können denselben Typ haben.

Bei Marktforschungen hat sich Folgendes ergeben:

1. Jeder Besucher bevorzugt es, genau drei Attraktionen zu besuchen, ohne zwei Attraktionen des gleichen Typs direkt hintereinander zu besuchen.
2. Besucher mögen Attraktionen nicht, die Endpunkt von vielen Gehwegen sind.

Um **alle** potenziellen Besucher zufriedenzustellen, stellt der Bürgermeister daher zwei Bedingungen an deinen Entwurf.

Ein geordnetes Tripel von **Typen** (t_1, t_2, t_3) für $0 \leq t_1, t_2, t_3 < 2K$ **hat Drachenaure**, wenn $t_1 \neq t_2$ und $t_2 \neq t_3$. Beachte, dass t_1 gleich t_3 sein könnte. Somit gibt es $2K \cdot (2K - 1)^2$ Tripel mit Drachenaure.

Bedingung 1: Für jedes Tripel mit Drachenaure (t_1, t_2, t_3) müssen drei Attraktionen a_1, a_2, a_3 (mit $0 \leq a_1, a_2, a_3 < N$) existieren, sodass:

- Die Typen von a_1, a_2, a_3 entsprechen t_1, t_2, t_3 . Das heißt, $T[a_1] = t_1$, $T[a_2] = t_2$ und $T[a_3] = t_3$.
- Zwischen a_1 und a_2 gibt es einen Gehweg.
- Zwischen a_2 und a_3 gibt es einen Gehweg.

Es ist egal, ob zwischen a_1 und a_3 ein Gehweg existiert oder nicht. Beachte außerdem, dass es sich bei a_1 und a_3 um dieselbe Attraktion handeln kann, wenn $t_1 = t_3$.

Bedingung 2: Jede Attraktion kann Endpunkt von **höchstens** K Gehwegen sein.

Diese Aufgabe besteht aus 50 **Output-Only**-Teilaufgaben mit **Teilbewertung**. Jede Teilaufgabe entspricht einem bestimmten Wert von K , und du musst einen Park entwerfen, der alle oben genannten Bedingungen für diesen Wert von K erfüllt. Deine Punktezahl hängt von der Anzahl der Attraktionen in deiner Lösung ab: weniger Attraktionen führen zu einer höheren oder gleichen Punktezahl.

Implementierungsdetails

Es gibt zwei Möglichkeiten, deine Lösung einzureichen, und du kannst für jede Teilaufgabe aussuchen, welche du verwendest:

- **Funktionsaufruf**
- **Ausgabedatei**

Um deine Lösung über einen **Funktionsaufruf** einzureichen, musst du die folgende Funktion implementieren:

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

- K : die Hälfte der Anzahl der Attraktionstypen sowie die maximal zulässige Anzahl von Gehwegen, deren Endpunkt bei einer einzelnen Attraktion liegen darf.
- Diese Funktion wird für jede Teilaufgabe genau einmal aufgerufen.

Die Funktion muss ein Paar (T, E) zurückgeben, das einen Vergnügungspark beschreibt. Sei M die Anzahl der Gehwege in deinem Park.

- T : ein Array der Länge N , das die Typen der Attraktionen beschreibt.
- E : ein Array der Länge M , das die Gehwege beschreibt. Für jedes $0 \leq j < M$ repräsentiert $E[j] = (U[j], V[j])$ einen bidirektionalen Gehweg zwischen der Attraktion $U[j]$ und der Attraktion $V[j]$ ($U[j] \neq V[j]$).

Um deine Lösung über eine **Ausgabedatei** einzureichen, erstelle eine Textdatei im folgenden Format und sende sie ein:

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

Beachte, dass eine **gültige** Lösung die folgenden Bedingungen erfüllen muss:

- $N \leq 2000$
- $0 \leq T[i] < 2K$ für jedes $0 \leq i < N$, und jeder Typ muss mindestens einer Attraktion zugewiesen sein.
- $0 \leq U[j], V[j] < N$ und $U[j] \neq V[j]$ für jedes $0 \leq j < M$.
- Die Bedingungen 1 und 2 müssen erfüllt sein.

Beschränkungen

- $1 \leq K \leq 50$

Punktevergabe

Es gibt 50 Teilaufgaben, eine für jede ganze Zahl K von 1 bis 50. Für die Teilaufgabe i ($1 \leq i \leq 50$) ist der Wert von K gleich i .

Jede Teilaufgabe hat eine Punktezahl S und eine gewünschte Anzahl von Attraktionen P gemäß der untenstehenden Tabelle.

Teilaufgabe	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	$12 \cdot K$

Wenn deine Lösung für eine der Teilaufgaben keinen gültigen Vergnügungspark beschreibt, wird sie mit 0 Punkten bewertet (und im CMS als `Output isn't correct` gemeldet).

Andernfalls wird deine Punktezahl anhand von N und den Parametern S und P wie folgt berechnet:

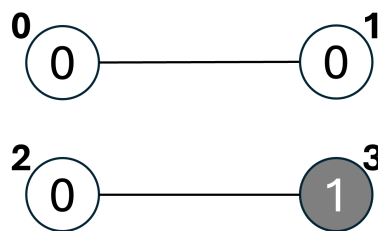
Bedingung	Bewertung
$N \leq P$	S
$P < N \leq 2P$	$(0.4 + 0.3 \cdot \frac{2P-N}{P}) \cdot S$
$2P < N \leq 2000$	$(0.1 + 0.3 \cdot \frac{2P}{N}) \cdot S$

Beispiel

Betrachte den folgenden Anruf:

```
construct(1)
```

In diesem Beispiel ist $K = 1$, daher gibt es $2K = 2$ Typen von Attraktionen. Die untenstehende Abbildung zeigt eine gültige Lösung mit $N = 4$ Attraktionen und $M = 2$ Gehwegen. Die Attraktionen 0, 1, 2 sind vom Typ 0 (ein chinesisches 龍窩 -- ein Drachennest ohne Dracheneier), Attraktion 3 ist vom Typ 1 (eine einzigartige „cho'kkalab o'tiradigan hojatxona“).



Es gibt zwei Tripel mit Drachenauren:

- Für das Tripel $(0, 1, 0)$ können wir $(a_1, a_2, a_3) = (2, 3, 2)$ wählen.
- Für das Tripel $(1, 0, 1)$ können wir $(a_1, a_2, a_3) = (3, 2, 3)$ wählen.

Das bedeutet, dass Bedingung 1 erfüllt ist.

Die Funktion kann das Paar $([0, 0, 0, 1], [(0, 1), (2, 3)])$ zurückgeben. Beachte, dass die für dieses Beispiel angegebene Lösung für $K = 1$ möglicherweise nicht optimal ist.

Beispielgrader

Eingabeformat:

```
K
```

Ausgabeformat:

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

Beachte, dass die Ausgabe des Beispielgraders dem erforderlichen Format für die Ausgabedatei entspricht.