

Ciudad Mágica

El alcalde de Tashkent quiere rediseñar el famoso parque de diversiones Ciudad Mágica. Te dan un número entero positivo K y tu tarea es diseñar un parque de la siguiente manera.

- Escoge un número entero N , el número de atracciones. Las atracciones están etiquetadas con los números enteros de 0 a $N - 1$.
- Agrega caminos bidireccionales, cada uno entre un par de **distintas** atracciones. Está permitido tener más de un camino entre el mismo par de atracciones. **No se requiere ser capaz de viajar entre cada par de atracciones usando los caminos.**
- Para cada i tal que $0 \leq i < N$, asigna a la atracción i un tipo $T[i]$. Hay $2K$ tipos numerados de 0 a $2K - 1$. Cada tipo debe ser asignado a al menos una atracción. Atracciones diferentes pueden tener el mismo tipo.

La investigación de mercado reveló que:

1. Cada visitante prefiere visitar tres atracciones sin visitar dos atracciones del mismo tipo consecutivamente.
2. A los visitantes no les gustan las atracciones que son el extremo de muchos caminos.

Para satisfacer a **todos** los potenciales visitantes el alcalde impone dos condiciones en tu diseño.

Llamamos a un triple ordenado de **tipos** (t_1, t_2, t_3) para $0 \leq t_1, t_2, t_3 < 2K$ **interesante** si $t_1 \neq t_2$ y $t_2 \neq t_3$. Nota que t_1 podría ser igual a t_3 . En consecuencia, hay $2K \cdot (2K - 1)^2$ triples interesantes.

Condición 1: Para cada triple interesante (t_1, t_2, t_3) , deben existir tres atracciones a_1, a_2, a_3 (con $0 \leq a_1, a_2, a_3 < N$) tales que:

- Los tipos de a_1, a_2, a_3 coinciden con t_1, t_2, t_3 . Esto es, $T[a_1] = t_1$, $T[a_2] = t_2$, y $T[a_3] = t_3$.
- Hay un camino entre a_1 y a_2 .
- Hay un camino entre a_2 y a_3 .

Es irrelevante si existe o no un camino entre a_1 y a_3 . También nota que cuando $t_1 = t_3$, las atracciones a_1 y a_3 pueden ser la misma.

Condición 2: Cada atracción puede ser el extremo de **como máximo** K caminos.

Esta tarea consiste en 50 subtareas de **solo-salida** con **puntuación parcial**. Cada subtarea corresponde a un valor específico de K , y debes diseñar un parque que satisfaga todas las condiciones de arriba para ese valor de K . Tu puntuación depende del número de atracciones en tu solución - un menor número de atracciones resultará en una puntuación mayor o igual.

Detalles de Implementación

Hay dos métodos para enviar tu solución, y puedes usar cualquiera de ellos para cada subtask:

- **Llamada a procedimiento**
- **Archivo de salida**

Para enviar tu solución mediante una **llamada a procedimiento**, debes implementar el siguiente procedimiento.

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

- K : la mitad del número de tipos de atracción, y también el máximo número de caminos de los que cada atracción puede ser un extremo.
- Este procedimiento es llamado exactamente una vez por cada subtask.

Este procedimiento debe retornar un par (T, E) describiendo un parque de diversiones. Sea M el número de caminos en tu parque.

- T : un arreglo de largo N describiendo el tipo de atracciones.
- E : un arreglo de largo M describiendo los caminos. Para cada $0 \leq j < M$, $E[j] = (U[j], V[j])$ representa un camino bidireccional entre distintas atracciones $U[j]$ y $V[j]$.

Para enviar tu solución mediante un **archivo de salida**, crea y envía un archivo de texto con el siguiente formato:

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

Note que su solución debe satisfacer las siguientes restricciones para ser considerada **válida**:

- $N \leq 2000$
- $0 \leq T[i] < 2K$ para cada $0 \leq i < N$ y cada tipo debe ser asignado a al menos una atracción.
- $0 \leq U[j], V[j] < N$ y $U[j] \neq V[j]$ para cada $0 \leq j < M$.
- las condiciones 1 y 2 deben ser cumplidas.

Restricciones

- $1 \leq K \leq 50$

Puntuación

Hay 50 subtareas, una para cada número entero K de 1 a 50. Para la subtarea i ($1 \leq i \leq 50$), el valor de K es i .

Cada subtarea tiene una puntuación S y un número de atracciones objetivo P según la tabla que aparece a continuación.

Subtasks	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	$12 \cdot K$

Para cada subtarea, si su solución no describe un parque de diversiones válido, entonces la puntuación de tu solución será 0 (reportada como `Output isn't correct` en el CMS).

En otro caso, su puntuación será calculada basado en N y los parámetros S y P como sigue:

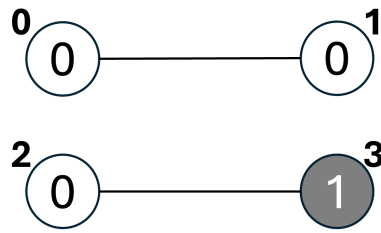
Condición	Puntuación
$N \leq P$	S
$P < N \leq 2P$	$(0.4 + 0.3 \cdot \frac{2P-N}{P}) \cdot S$
$2P < N \leq 2000$	$(0.1 + 0.3 \cdot \frac{2P}{N}) \cdot S$

Ejemplo

Considere la siguiente llamada.

```
construct(1)
```

En este ejemplo, $K = 1$, por lo que hay $2K = 2$ tipos de atracciones. La figura a continuación muestra una solución válida con $N = 4$ atracciones y $M = 2$ caminos. Las atracciones 0, 1, 2 son de tipo 0, y la atracción 3 es de tipo 1.



Hay dos triples interesantes:

- Para el tipo triple $(0, 1, 0)$, podemos escoger $(a_1, a_2, a_3) = (2, 3, 2)$.
- Para el tipo triple $(1, 0, 1)$, podemos escoger $(a_1, a_2, a_3) = (3, 2, 3)$.

Esto significa que la Condición 1 es satisfecha.

El procedimiento puede retornar un par $([0, 0, 0, 1], [(0, 1), (2, 3)])$. Tenga en cuenta que la solución proporcionada para este ejemplo podría no ser óptima para $K = 1$.

Evaluador de ejemplo

Formato de Entrada:

K

Formato de Salida:

```

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

```

Tenga en cuenta que la salida del evaluador de ejemplo coincide con el formato requerido para el archivo de salida.