

Magic City

El alcalde de Tashkent desea rediseñar el famoso parque de diversiones Magic City. Dado un entero positivo K , tu tarea es diseñar un parque de la siguiente manera:

- Elegir un entero N , la cantidad de atracciones. Las atracciones se numeran con enteros desde 0 hasta $N - 1$.
- Agregar pasarelas bidireccionales, cada una entre dos atracciones **distintas**. Se permite tener más de una pasarela entre el mismo par de atracciones. **No es necesario que se pueda viajar entre todo par de atracciones utilizando las pasarelas.**
- Para cada i tal que $0 \leq i < N$, asignarle a la atracción i el tipo $T[i]$. Hay $2K$ tipos numerados desde 0 hasta $2K - 1$. Cada tipo se debe asignar a por lo menos una atracción. Atracciones diferentes pueden ser del mismo tipo.

Un estudio de mercado reveló lo siguiente:

1. Cada visitante prefiere visitar tres atracciones sin visitar dos atracciones seguidas del mismo tipo.
2. A los visitantes no les gustan las atracciones que están en un extremo de muchas pasarelas.

Con el fin de satisfacer a **todos** los potenciales visitantes, el alcalde le impone dos condiciones a tu diseño.

Decimos que una terna ordenada de **tipos** (t_1, t_2, t_3) para $0 \leq t_1, t_2, t_3 < 2K$ es **interesante** si $t_1 \neq t_2$ y $t_2 \neq t_3$. Notar que t_1 podría ser igual que t_3 . Por lo tanto, existen $2K \cdot (2K - 1)^2$ ternas interesantes.

Condición 1: Para cada terna 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 sean t_1, t_2, t_3 . Es decir, $T[a_1] = t_1$, $T[a_2] = t_2$, y $T[a_3] = t_3$.
- Existe una pasarela entre a_1 y a_2 .
- Existe una pasarela entre a_2 y a_3 .

Es irrelevante si hay o no hay una pasarela entre a_1 y a_3 . Notar también que cuando $t_1 = t_3$, las atracciones a_1 y a_3 pueden ser la misma.

Condición 2: Cada atracción puede estar en un extremo de **a lo sumo** K pasarelas.

Este problema consiste de 50 subtareas **output-only** con **puntaje parcial**. Cada subtarea se corresponde con un valor específico de K , y debes diseñar un parque que satisfaga todas las

condiciones anteriores para ese valor de K . Tu puntaje depende de la cantidad de atracciones en tu solución: menos atracciones resultan en un puntaje mayor o igual.

Detalles de implementación

Hay dos formas de enviar tu solución, y puedes utilizar una o la otra para cada subtarea:

- **Llamada a función**
- **Archivo de salida**

Para enviar tu solución mediante una **llamada a función**, debes implementar la siguiente función:

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

- K : la mitad de la cantidad de tipos de atracciones, y también la máxima cantidad de pasarelas en un extremo de las cuales una atracción puede estar.
- Esta función se llama exactamente una vez en cada subtarea.

La función debe retornar un par (T, E) que describa un parque de diversiones. Sea M la cantidad de pasarelas en tu parque.

- T : arreglo de longitud N que describe los tipos de las atracciones.
- E : arreglo de longitud M que describe las pasarelas. Para cada $0 \leq j < M$, $E[j] = (U[j], V[j])$ representa una pasarela bidireccional entre las atracciones distintas $U[j]$ 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]
```

Notar que tu 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 se debe asignar a por lo 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 se deben cumplir.

Restricciones

- $1 \leq K \leq 50$

Puntuación

Hay 50 subtareas, una por cada entero K desde 1 hasta 50. Para la subtarea i ($1 \leq i \leq 50$), el valor de K es i .

Cada subtarea tiene un puntaje S y un número de atracciones objetivo P de acuerdo a la siguiente tabla.

Subtareas	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	$12 \cdot K$

En cada subtarea, si tu solución no describe un parque de diversiones válido, el puntaje será 0 (reportado como `Output isn't correct` en CMS).

De lo contrario, tu puntaje se calcula en base a N y a los parámetros S y P de la siguiente manera:

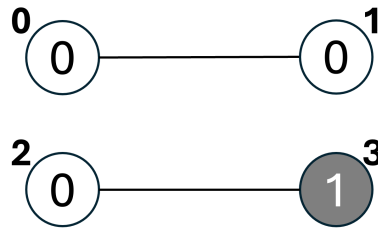
Condición	Puntaje
$N \leq P$	S
$P < N \leq 2P$	$(0.4 + 0.3 \cdot \frac{2P-N}{P}) \cdot S$
$2P < N \leq 2000$	$(0.1 + 0.3 \cdot \frac{2P}{N}) \cdot S$

Ejemplo

Considere la siguiente llamada:

```
construct(1)
```

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



Hay dos ternas interesantes:

- Para la terna de tipos $(0, 1, 0)$, podemos elegir $(a_1, a_2, a_3) = (2, 3, 2)$.
- Para la terna de tipos $(1, 0, 1)$, podemos elegir $(a_1, a_2, a_3) = (3, 2, 3)$.

Por lo tanto la Condición 1 se satisface.

La función puede retornar el par $([0, 0, 0, 1], [(0, 1), (2, 3)])$. Notar que la solución provista en este ejemplo podría no ser óptima para $K = 1$.

Evaluador local

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]
```

Notar que la salida del evaluador local se corresponde con el formato del archivo de salida.