

Magic City

El alcalde de Tashkent quiere rediseñar el famoso parque de atracciones Magic City. Se te da un número entero positivo K y tu tarea consiste en diseñar un parque de la siguiente manera:

- Elige un número entero N , el número de atracciones. Las atracciones están numeradas con números enteros de 0 a $N - 1$.
- Añade pasarelas bidireccionales, cada una entre un par de atracciones **distintas**. Se permite tener más de una pasarela entre un mismo par de atracciones. **No es necesario poder desplazarse entre todas las parejas de atracciones utilizando las pasarelas.**
- 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 asignarse a una atracción como mínimo. Diferentes atracciones pueden tener el mismo tipo.

Una 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 muchas pasarelas.

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

Llamamos a una terna 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 $t_2 \neq t_3$. Nótese que t_1 puede ser igual a t_3 . Por lo tanto, hay $2K \cdot (2K - 1)^2$ 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 coinciden con t_1, t_2, t_3 . Es decir, $T[a_1] = t_1$, $T[a_2] = t_2$ y $T[a_3] = t_3$.
- Hay una pasarela entre a_1 y a_2 .
- Hay una pasarela entre a_2 y a_3 .

Es irrelevante si existe o no una pasarela entre a_1 y a_3 . Además, ten en cuenta que cuando $t_1 = t_3$, las atracciones a_1 y a_3 pueden ser las mismas.

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

Esta tarea consta de 50 subtareas **output-only** con **puntuación parcial**. Cada subtarea corresponde a un valor específico de K , y debes diseñar un parque que satisfaga todas las condiciones anteriores para ese valor de K . Tu puntuación depende del número de atracciones en tu solución: menos atracciones siempre darán una puntuación mayor o igual.

Detalles de implementación

Existen dos métodos para enviar tu solución, y puedes utilizar cualquiera de ellos para cada subtarea:

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

Para enviar su 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 del número de tipos de atracciones, y también el número máximo de pasarelas de las que cada atracción puede ser un extremo.
- Esta función se llama exactamente una vez por cada subtarea.

La función debe devolver un par (T, E) que describa un parque de atracciones. Sea M el número de pasarelas en tu parque.

- T : un arreglo de longitud N que describe los tipos de atracciones.
- E : un 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 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]
```

Ten en cuenta que tu solución debe cumplir 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 asignarse al menos a una atracción.
- $0 \leq U[j], V[j] < N$ y $U[j] \neq V[j]$ para cada $0 \leq j < M$.
- Deben cumplirse las condiciones 1 y 2.

Restricciones

- $1 \leq K \leq 50$

Puntuación

Hay 50 subtareas, una para 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 una puntuación S y un número objetivo de atracciones P según la tabla que aparece a continuación.

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	$12 \cdot K$

Para cada subtarea, si tu solución no describe un parque de atracciones válido, la puntuación de tu solución será 0 (se informará como `Output isn't correct` en CMS).

De lo contrario, tu puntuación se calcula en función de N y de los parámetros S y P de la siguiente manera:

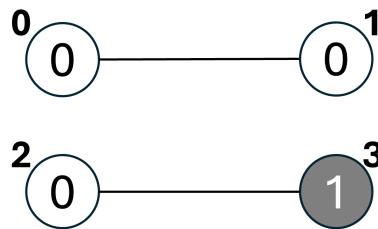
Condición	Puntuación
$N \leq P$	S
$P < N \leq 2P$	$(0.4 + 0.3 \cdot \frac{2P-N}{P}) \cdot S$
$2P < N \leq 2000$	$(0.1 + 0.3 \cdot \frac{2P}{N}) \cdot S$

Ejemplo

Considere la siguiente llamada:

```
construct(1)
```

En este ejemplo, $K = 1$, por lo que hay $2K = 2$ tipos de atracciones. La siguiente figura muestra 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 $(a_1, a_2, a_3) = (3, 2, 3)$.

Esto significa que se cumple la condición 1.

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

Calificador 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]
```

Nota que la salida del evaluador de prueba coincide con el formato requerido para el archivo de salida.