

Magic City

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

- Elije un número entero N , el número de atracciones. Las atracciones son etiquetadas con números enteros del 0 al $N - 1$.
- Añade senderos bidireccionales, cada uno entre un par de atracciones **distintas**. Está permitido tener más de un sendero entre el mismo par de atracciones. **No es necesario poder desplazarse entre cada par de atracciones utilizando los senderos.**
- Para cada i tal que $0 \leq i < N$, asigne a la atracción i un tipo $T[i]$. Hay $2K$ tipos disponibles del 0 al $2K - 1$. 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.
2. A los visitantes no les gustan las atracciones que sean extremo de muchos senderos.

Con el fin de satisfacer **todos** los potenciales visitantes, el alcalde impuso dos condiciones en tu diseño.

Llamamos una tripleta ordenada 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 $t_2 \neq t_3$. Note que t_1 podría ser igual a t_3 . Por lo tanto, hay $2K \cdot (2K - 1)^2$ tripletas interesantes.

Condición 1: Para cada tripleta interesante (t_1, t_2, t_3) , debe existir tres atracciones a_1, a_2, a_3 (con $0 \leq a_1, a_2, a_3 < N$) tal 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 un sendero entre a_1 y a_2 .
- Hay un sendero entre a_2 y a_3 .

Es irrelevante si existe o no un sendero entre a_1 y a_3 . Además, note que cuando $t_1 = t_3$, las atracciones a_1 y a_3 podrían ser las mismas.

Condición 2: Cada atracción puede ser el extremo de **a lo mucho con** K senderos.

Esta tarea consiste de 50 **output-only (sólo salida)** subtareas 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 . Tú puntuación depende del número de atracciones en tu solución: un número menor de atracciones produce una puntuación igual o más alta.

Detalles de Implementación

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

- **Procedure call (Llamada a función)**
- **Output file (Archivo de salida)**

Para enviar tu solución por medio de **Procedure call (Llamada a función)**, debes implementar la siguiente función.

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

(**construct** significa "construir")

- K : la mitad del número de tipos de atracciones, y a su vez el máximo número permitido de atracciones que pueden ser extremos de senderos.
- Esta función se llama exactamente una vez para cada subtask.

La función debe retornar una pareja (T, E) describiendo un parque de diversiones. Sea M el número de senderos en tu parque.

- T : un arreglo de tamaño N que describe los tipos de atracciones.
- E : un arreglo de tamaño M que representa los senderos. Para cada $0 \leq j < M$, $E[j] = (U[j], V[j])$ representa un sendero bidireccional entre atracciones distintas $U[j]$ y $V[j]$.

Para enviar tu solución **output file (archivo de salida)**, crea and 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 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 debe ser asignado a al menos una atracción.

- $0 \leq U[j], V[j] < N$ y $U[j] \neq V[j]$ para cada $0 \leq j < M$.
- Condiciones 1 y 2 deben cumplirse.

Restricciones

- $1 \leq K \leq 50$

Puntuación

Hay 50 subtareas, una para cada 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 la cantidad de atracciones P según la tabla de abajo.

Subtarea	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 diversiones valido, entonces tu puntuación será 0 (reportada como `Output isn't correct` en CMS).

De lo contrario, tu puntuación será calculada con base en N y los parametros S y P como se muestra:

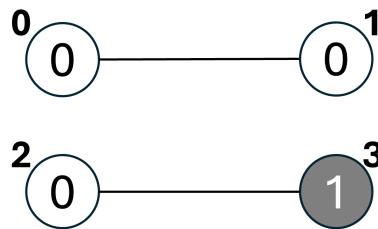
Condición	Puntuación
$N \leq P$	S
$P < N \leq 2P$	$(0.4 + 0.3 \cdot \frac{2P-N}{P}) \cdot S$
$2P < N \leq 2000$	$(0.1 + 0.3 \cdot \frac{2P}{N}) \cdot S$

Ejemplo

Considere la siguiente llamada:

```
construct(1)
```

En este ejemplo, $K = 1$, entonces hay $2K = 2$ tipos de atracciones. La figura de abajo muestra una solución valida con $N = 4$ atracciones y $M = 2$ senderos. Atracciones 0, 1, 2 son un tipo de 0, y la atracción 3 es de tipo 1.



Hay dos tripletas interesantes :

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

Esto significa que la Condición 1 es satisfecha.

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

Sample Grader (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]
```

Note que la salida del evaluador de ejemplos coincide con el formato requerido para el archivo de salida.