

Magic City

O presidente da Câmara de Tashkent quer reformular o famoso parque de diversões Magic City. Dado um número inteiro positivo K , a tua tarefa é projetar um parque da seguinte forma:

- Escolhe um número inteiro N , o número de atrações. As atrações são identificadas por números inteiros de 0 a $N - 1$.
- Adiciona caminhos bidirecionais, cada um ligando um par de atrações **distintas**. É permitido existir mais do que um caminho entre o mesmo par de atrações. **Não é obrigatório poder deslocar-se entre todos os pares de atrações utilizando os caminhos.**
- Para cada i tal que $0 \leq i < N$, atribui à atração i um tipo $T[i]$. Existem $2K$ tipos numerados de 0 a $2K - 1$. Cada tipo deve ser atribuído a pelo menos uma atração. Atrações diferentes podem ter o mesmo tipo.

Um pesquisa de mercado revelou que:

1. Cada visitante prefere visitar três atrações, sem visitar consecutivamente duas atrações do mesmo tipo.
2. Os visitantes não gostam de atrações que sejam extremidades de muitos caminhos.

De modo a satisfazer **todos** os potenciais visitantes, o presidente da Câmara impõe duas condições ao teu projeto.

Dizemos que uma tripla ordenada de **tipos** (t_1, t_2, t_3) para $0 \leq t_1, t_2, t_3 < 2K$ é **interessante** se $t_1 \neq t_2$ e $t_2 \neq t_3$. Nota que t_1 pode ser igual a t_3 . Assim, existem $2K \cdot (2K - 1)^2$ triplas interessantes.

Condição 1: Para cada tripla interessante (t_1, t_2, t_3) , devem existir três atrações a_1, a_2, a_3 (com $0 \leq a_1, a_2, a_3 < N$) tais que:

- Os tipos de a_1, a_2, a_3 correspondem t_1, t_2, t_3 . Ou seja, $T[a_1] = t_1$, $T[a_2] = t_2$ e $T[a_3] = t_3$.
- Existe um caminho entre a_1 e a_2 .
- Existe um caminho entre a_2 e a_3 .

É irrelevante se existe ou não um caminho entre a_1 e a_3 . Nota ainda que quando $t_1 = t_3$, as atrações a_1 e a_3 podem ser as mesmas.

Condição 2: Cada atração pode ser uma extremidade de **no máximo** K caminhos.

Este problema é constituído por 50 subtarefas de tipo **output-only**, com **pontuação parcial**. Cada subtarefa corresponde a um valor específico de K , e deves conceber um parque que satisfaça todas as condições acima referidas para esse valor de K . A tua pontuação depende do número de atrações na tua solução: menos atrações resultam numa pontuação mais elevada ou igual.

Detalhes da Implementação

Existem dois métodos para submeteres a tua solução, e podes utilizar qualquer um deles para cada subtarefa:

- **Chamada de uma função**
- **Ficheiro de output**

Para submeteres a tua solução através de uma **chamada de função**, deves implementar a seguinte função.

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

- K : metade do número de tipos de atrações, e também o número máximo de caminhos dos quais uma atração pode ser uma extremidade.
- Este procedimento é chamado exatamente uma vez para cada subtarefa.

O procedimento deve devolver um par (T, E) descrevendo um parque de diversões. Seja M o número de caminhos no teu parque.

- T : um array de tamanho N que descreve os tipos de atrações.
- E : um array de tamanho M que descreve os caminhos. Para cada $0 \leq j < M$, $E[j] = (U[j], V[j])$ representa uma caminho bidireccional entre duas atrações distintas $U[j]$ e $V[j]$.

Para submeteres a tua solução através de um **ficheiro de output**, cria e envia um ficheiro de texto no seguinte formato:

```
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 a tua solução deve satisfazer as seguintes restrições para ser considerada **válida**:

- $N \leq 2000$
- $0 \leq T[i] < 2K$ para cada $0 \leq i < N$ e cada tipo deve ser atribuído a pelo menos uma atração.
- $0 \leq U[j], V[j] < N$ e $U[j] \neq V[j]$ para cada $0 \leq j < M$.
- As condições 1 e 2 devem ser cumpridas.

Restrições

- $1 \leq K \leq 50$

Pontuação

Existem 50 subtarefas, uma para cada número inteiro K de 1 a 50. Para a subtarefa i ($1 \leq i \leq 50$), o valor de K é i .

Cada subtarefa tem uma pontuação S e um número alvo de atrações P de acordo com a tabela abaixo.

Subtarefas	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 subtarefa, se a tua solução não descrever um parque de diversões válido, a pontuação da tua solução será 0 (reportada como `Output isn't correct` no CMS).

Caso contrário, a tua pontuação será calculada com base em N e nos parâmetros S e P da seguinte forma:

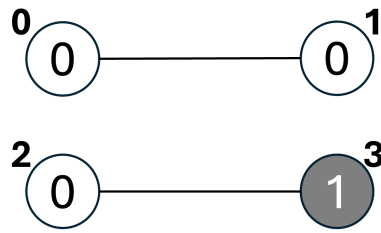
Condição	Pontos
$N \leq P$	S
$P < N \leq 2P$	$(0.4 + 0.3 \cdot \frac{2P-N}{P}) \cdot S$
$2P < N \leq 2000$	$(0.1 + 0.3 \cdot \frac{2P}{N}) \cdot S$

Exemplo

Considera a seguinte chamada:

```
construct(1)
```

Neste exemplo, $K = 1$, pelo que existem $2K = 2$ tipos de atrações. A figura abaixo mostra uma solução válida com $N = 4$ atrações e $M = 2$ caminhos. As atrações 0, 1, 2 são do tipo 0 e a atração 3 é do tipo 1.



Existem duas triplas interessantes:

- Para a tripla de tipo $(0, 1, 0)$, podemos escolher $(a_1, a_2, a_3) = (2, 3, 2)$.
- Para a tripla de tipo $(1, 0, 1)$, podemos escolher $(a_1, a_2, a_3) = (3, 2, 3)$.

Isto significa que a Condição 1 foi satisfeita.

O procedimento pode devolver o par $([0, 0, 0, 1], [(0, 1), (2, 3)])$. Nota que a solução fornecida para este exemplo pode não ser a ideal para $K = 1$.

Avaliador de Exemplo

Formato de Input:

K

Formato de Output:

```

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 o output do avaliador de exemplo corresponde ao formato exigido para o ficheiro de output.