

매직 시티

타슈켄트의 시장은 유명한 매직 시티 놀이 공원을 재설계하기를 원한다. 양의 정수 K 가 주어질 때, 당신은 다음과 같이 공원을 설계해야 한다:

- 놀이기구의 수 N 을 결정하고 놀이기구에 0부터 $N - 1$ 까지의 번호를 붙인다.
- 서로 다른 놀이기구의 쌍 사이에 양방향 보행로를 만든다. 놀이기구의 같은 쌍 사이에 하나 이상의 보행로가 허락된다. **보행로들을 사용해서 모든 놀이기구 쌍 사이를 이동할 수 없어도 괜찮다.**
- $0 \leq i < N$ 인 각 i 에 대해서, 놀이기구 i 에 타입 $T[i]$ 를 부여한다. 타입은 0부터 $2K - 1$ 까지 번호가 붙은 $2K$ 개의 타입이 있다. 모든 타입은 적어도 하나의 놀이기구에 부여되어야 한다. 다른 놀이기구들이 같은 타입을 가질 수 있다.

시장 조사 결과로 다음이 밝혀졌다:

1. 방문객들은 세 곳의 놀이기구를 방문하되, 같은 타입의 놀이기구를 연속해서 두 곳 방문하지 않는 것을 선호한다.
2. 방문객들은 많은 보행로의 끝점이 되는 놀이기구를 싫어한다.

모든 잠재적 방문객을 만족시키기 위해서, 시장은 당신의 설계에 두 가지 조건을 제시한다.

$0 \leq t_1, t_2, t_3 < 2K$ 에 대해서, $t_1 \neq t_2$ 이고 $t_2 \neq t_3$ 이면, **타입들의 순서 트리플** (t_1, t_2, t_3) 를 **흥미롭다**고 부른다. t_1 은 t_3 와 같을 수 있음에 주목하자. 따라서 총 $2K \cdot (2K - 1)^2$ 개의 흥미로운 트리플이 존재한다.

조건 1: 각 흥미로운 트리플 (t_1, t_2, t_3) 에 대해서, 다음을 만족하는 3개의 놀이기구 a_1, a_2, a_3 ($0 \leq a_1, a_2, a_3 < N$)가 존재해야 한다:

- a_1, a_2, a_3 의 타입은 각각 t_1, t_2, t_3 와 일치한다. 즉, $T[a_1] = t_1, T[a_2] = t_2, T[a_3] = t_3$ 이다.
- a_1 과 a_2 사이에 보행로가 존재한다.
- a_2 과 a_3 사이에 보행로가 존재한다.

a_1 과 a_3 사이에 보행로가 있는지는 상관없다. 또한 $t_1 = t_3$ 일 때, a_1 과 a_3 가 같을 수도 있음에 주목하자.

조건 2: 모든 놀이기구는 **많아야 K 개** 보행로의 끝점이 될 수 있다.

이 문제는 50개의 **부분 점수**가 주어지는 **output-only** 서브태스크로 구성된다. 각 서브태스크에는 특정한 K 값이 주어지고, 당신은 K 의 값에 따라 위 모든 조건을 만족하는 공원을 설계해야 한다. 당신의 점수는 당신의 답에 포함된 놀이기구 수에 따라 달라진다: 놀이기구가 적을 수록 더 높거나 같은 점수를 받을 수 있다.

Implementation Details

답을 제출하는 두 가지 방법이 있다. 당신은 각 서브 태스크에 대해 둘 중 하나의 방법을 사용할 수 있다.

- Procedure call
- Output file

procedure call을 통해 답을 제출하기 위해서는, 당신은 다음 함수를 구현해야 한다.

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

- K : 놀이기구 타입 수의 절반이자 각 놀이기구가 끝점이 될 수 있는 보행로의 최대 개수
- 이 함수는 각 서브태스크에 대해 정확히 한 번 호출된다.

이 함수는 놀이공원을 나타내는 순서쌍 (T, E) 를 반환해야 한다. M 을 공원에 있는 보행로의 수라고 하자.

- T : 놀이기구의 타입들을 나타내는 길이 N 의 배열
- E : 보행로들을 나타내는 길이 M 의 배열. 각 $0 \leq j < M$ 에 대해서, $E[j] = (U[j], V[j])$ 는 서로 다른 놀이기구 $U[j]$ 와 $V[j]$ 사이의 양방향 보행로를 나타낸다.

output file을 통해 답을 제출하기 위해서는, 다음 형식에 맞춰서 텍스트 파일을 만들고 제출해야 한다:

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

당신의 답이 **유효한** 것으로 간주되려면 다음 제약 조건을 만족해야 한다:

- $N \leq 2000$
- 각 $0 \leq i < N$ 에 대해, $0 \leq T[i] < 2K$, 그리고 모든 타입은 적어도 하나의 놀이기구에 배정되어야 한다.
- 각 $0 \leq j < M$ 에 대해, $0 \leq U[j], V[j] < N$ 이고 $U[j] \neq V[j]$.
- 조건 1, 조건 2를 충족해야 한다.

Constraints

- $1 \leq K \leq 50$

Scoring

1부터 50까지의 각 정수 K 마다 하나씩 50개의 서브태스크가 있다. 서브태스크 i ($1 \leq i \leq 50$)에 대해, K 의 값은 i 이다.

각 서브태스크는 아래 표에 따라 점수 S 와 놀이기구의 목표 개수 P 를 가진다.

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	$12 \cdot K$

각 서브태스크에 대해, 당신의 답이 유효한 놀이공원을 나타내지 않는다면, 당신의 점수는 0이 된다. (CMS에서 Output isn't correct로 나타난다).

그렇지 않다면, 당신의 점수는 다음과 같이 N, S, P 에 따라 계산된다.

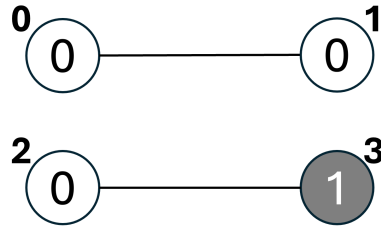
Condition	Score
$N \leq P$	S
$P < N \leq 2P$	$(0.4 + 0.3 \cdot \frac{2P-N}{P}) \cdot S$
$2P < N \leq 2000$	$(0.1 + 0.3 \cdot \frac{2P}{N}) \cdot S$

Example

다음 호출을 생각해보자:

```
construct(1)
```

이 예시에서는 $K = 1$ 이고, 따라서 $2K = 2$ 개의 놀이기구 타입이 존재한다. 아래 그림은 $N = 4$ 개 놀이기구와 $M = 2$ 개 보행로를 가진 유효한 답을 보여준다. 놀이기구 0, 1, 2는 타입 0이고, 놀이기구 3은 타입 1이다.



두 가지 흥미로운 트리플이 존재한다:

- 타입 트리플 $(0, 1, 0)$ 에 대해, $(a_1, a_2, a_3) = (2, 3, 2)$ 을 선택할 수 있다.
- 타입 트리플 $(1, 0, 1)$ 에 대해, $(a_1, a_2, a_3) = (3, 2, 3)$ 을 선택할 수 있다.

따라서 조건 1이 성립한다.

함수는 순서쌍 $([0, 0, 0, 1], [(0, 1), (2, 3)])$ 을 반환한다. 이 예제에 대해 제공된 답은 $K = 1$ 에 대해 최적이지 아닐 수 있다.

Sample Grader

Input format:

```
K
```

Output format:

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

샘플 그레이더의 출력 결과가 출력 파일에 요구되는 형식과 일치함에 주목하자.