다리 보수 공사 (roadwork)

KOI 도시는 도시를 동서로 가로지르는 큰 강을 중심으로 형성되어 있다. 강의 북쪽과 남쪽에는 각각 N 개의 마을이 위치해 있다. 북쪽의 마을들은 하류에서부터 순서대로 $A_1, A_2, \ldots, A_N$ 과 같이 식별되며, 남쪽의 마을들은 하류에서부터 순서대로 $B_1, B_2, \ldots, B_N$ 과 같이 식별된다. 고로, KOI 도시에는 총 $2N$개의 마을이 존재한다.

KOI 도시의 사람들은 원래 뗏목을 타고 다니며 교류하였으나, 근대화가 진행되면서 강을 가로지르는 다리를 건설하게 되었다. KOI 도시는 하류에서부터 발전하였기 때문에, 맨 처음 건설된 다리는 마을 A_1 과 마을 B_1을 이었다. KOI 도시의 사람들은 이렇게 건설된 첫 다리를 0**번 다리** 라고 부른다. 이후, KOI 도시는 추가적으로 1번 다리, 2번 다리, …, $2N-2$번 다리를 순서대로 건설하여, 총 $2N-1$개의 다리를 건설하였다.

0번 다리를 지은 이후, 모든 다리는 직전에 지은 다리와 인접한 위치에 건설되었다. 구체적으로, 모든 $0 \le i \le 2N-3$ 에 대해서, 만약 i번 다리가 A_x 마을과 B_y 마을을 잇는다면, $i+1$번 다리는 A_x 마을과 B_{y+1} 마을을 잇거나, A_{x+1} 마을과 B_y 마을을 잇는다. 두 경우 중 어떤 것이 결정되었는지는 길이 $2N-2$의 문자열 S에 기록되어 있다. 만약 $S[i] =$ `'A'` 일 경우 $i+1$번 다리는 A_x 마을과 B_{y+1} 마을을 이으며, $S[i] =$`'B'` 일 경우 $i+1$번 다리는 A_{x+1} 마을과 B_y 마을을 잇는다. S 에는 정확히 $N-1$개의 `'A'`와 $N-1$개의 `'B'` 가 등장한다. 이에 따라 다음과 같은 사실이 성립함을 증명할 수 있다:

- 존재하지 않는 마을을 잇는 다리는 등장하지 않는다.
- 임의의 서로 다른 두 마을을 다리만을 통해서 항상 오갈 수 있다.
- $2N-2$번 다리는 A_N 마을과 B_N 마을을 잇는다.

KOI 도시는 다리들을 보수하는 공사를 진행하려고 한다. 보수 공사는 $2N-1$개 다리 중 몇 개의 다리를 선택해서 진행한다. 공사는 소음을 유발하기 때문에, 어떠한 마을에 대해 이 마을을 잇는 2개 이상의 다리가 동시에 공사의 대상이 되는 일은 피하려고 한다. KOI 도시는, 이 조건을 만족하면서 최대한 많은 다리에 공사를 진행하려고 한다. 또한, 향후 예상하지 못한 문제가 생길 수 있으니, 조건을 만족하면서 최대한 많은 다리에 공사를 진행할 수 있는 경우의 수를 10^9+7로 나눈 나머지를 계산하고자 한다. 두 공사가 다르다는 것은, 공사의 대상이 되는 다리의 집합이 다르다는 것으로 정의한다.

당신은 KOI 도시를 도와 이 두 값을 모두 계산하여야 한다. 하지만, 다리의 최대 개수만을 계산하였을 때도 부분 점수를 얻을 수 있다.

함수 목록 및 정의

여러분은 아래 함수를 구현해야 한다.

```
array<int, 2> roadwork(string S)
```

- S: 크기가 $2N-2$인 문자열.
- 이 함수는 한 테스트 케이스에서 T**번 호출될 수 있다**.
- 보수 공사를 진행할 수 있는 다리의 최대 개수가 p이고, p 개의 다리를 공사하는 경우의 수를 10^9+7로 나눈 나머지가 q일 때, 함수는 $[p, q]$ 를 반환해야 한다. 이 때 $0 \le q \le 10^9+6$ 을 만족하여야 한다.

제출하는 소스 코드의 어느 부분에서도 입출력 함수를 실행해서는 안 된다.

제약 조건

- $1 \le T \le 10$
- $2 \le N \le 2^{21}$
- S 에는 정확히 $N-1$개의 `'A'`와 $N-1$개의 `'B'` 가 등장한다.

부분문제

1. (4점) $N \le 2^1$
2. (5점) $N \le 2^3$
3. (6점) $N \le 2^5$
4. (7점) $N \le 2^7$
5. (8점) $N \le 2^9$
6. (9점) $N \le 2^{11}$
7. (10점) $N \le 2^{13}$
8. (11점) $N \le 2^{15}$
9. (12점) $N \le 2^{17}$
10. (13점) $N \le 2^{19}$
11. (15점) 추가적인 제약 조건이 없다.

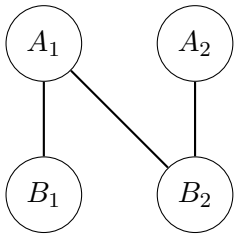
각 부분문제에서 다리의 최대 개수만을 올바르게 반환하였을 경우, 40% 의 부분 점수를 얻는다.

예제 1

$N = 2$, $S =$"AB" 인 경우를 생각해 보자. 그레이더는 다음과 같이 함수를 호출한다.

```
roadwork("AB")
```

아래 그림은 KOI 도시의 구조를 나타낸다.



0번 다리와 2번 다리를 공사할 경우 최대 2개의 다리에 공사를 진행할 수 있다. 2개의 다리에 공사를 진행하는 방법은 이것이 유일하다. 함수는 $[2, 1]$ 을 반환해야 한다. 만약 $[2, -1], [2, 0]$ 등을 반환할 경우, 다리의 최대 개수만을 올바르게 반환한 것으로 간주된다.

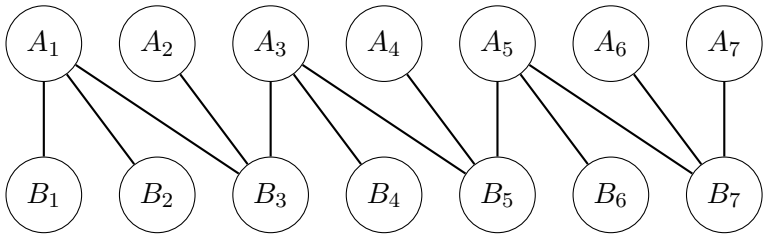
1개의 다리에 공사를 진행하는 방법이 3가지 있지만, 이 경우는 최대한 많은 다리에 공사를 진행하지 않기에 세지 않는다.

예제 2

$N = 7$, $S =$"AABBAABBAABB" 인 경우를 생각해 보자. 그레이더는 다음과 같이 함수를 호출한다.

```
roadwork("AABBAABBAABB")
```

아래 그림은 KOI 도시의 구조를 나타낸다.



최대 6개의 다리에 공사를 진행할 수 있다. 그러한 경우의 수는 총 4가지이다. 함수는 $[6, 4]$ 를 반환해야 한다.

Sample grader

Sample grader는 아래와 같은 형식으로 입력을 받는다.

- Line 1: T
- Line $2 + i$ $(0 \le i \le T - 1)$: N S

$i + 1$번째 테스트 케이스에서 `roadwork`가 반환한 배열을 $C[i]$라고 하자. Sample grader는 다음을 출력한다.

- Line $1 + j$ $(0 \le j \le T - 1)$: $C[j][0]$ $C[j][1]$

Sample grader는 실제 채점에서 사용하는 그레이더와 다를 수 있음에 유의하라.