

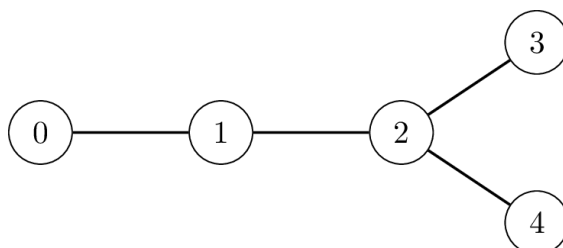
이주

국립 고고학 박물관은 볼리비아의 공룡 이주 패턴을 연구하고 있다. 고고학자들은 N 개의 서로 다른 장소에서 공룡 발자국을 발견하였다. 장소들은 오래된 것부터 시작해서 0부터 $N - 1$ 까지 번호가 붙어 있다. 즉, 장소 0이 가장 오래된 것이며 장소 $N - 1$ 이 가장 최근의 것이다.

각 장소의 공룡들은 (0번 장소를 제외하고) 다른 더 오래된 장소에서 직접 이주해 왔다. 즉, 모든 장소 i 에 대해 ($1 \leq i \leq N - 1$) 장소 i 의 공룡들은 어떤 더 오래된 하나의 장소 $P[i]$ 에서 ($P[i] < i$) 직접 이주해 왔다. 한 장소에서 다른 여러 장소로 직접 이주해 간 것은 가능하다.

고고학자들은 한 장소에서 다른 장소로 직접 이주해 간 관계를 **무방향 간선**으로 표현한다. 임의의 서로 다른 두 장소 x 와 y 에 대해 x 에서 y 로 하나 이상의 무방향 간선들을 이용하여 이동하는 것이 가능함을 알 수 있다. 두 장소 x 와 y 간의 **거리**는 x 에서 y 로 이동하며 거치는 최소 간선 개수로 정의된다.

예를 들어, 아래 그림은 $N = 5$ 개의 장소가 있고 $P[1] = 0$, $P[2] = 1$, $P[3] = 2$, $P[4] = 2$ 인 상황을 보여 준다. 장소 3에서 장소 4로 이동하는데 2개의 간선을 사용하면 가능하므로 두 장소 간의 거리는 2이다.



박물관은 장소들 중 그 거리가 가장 먼 쌍을 알아내려고 한다.

답이 유일하지 않음에 주의하라. 위의 예에서 장소 쌍 (0, 3)과 장소 쌍 (0, 4)는 모두 그 거리가 전체에서 가장 큰 3이다. 이러한 경우에 두 쌍 모두 정답으로 인정된다.

초기에 $P[i]$ 들의 값은 **알려지지 않았다**. 박물관은 연구팀을 각각의 장소에 보낸다. 보내는 순서는 장소 1부터 장소 번호 순서대로, 즉, 장소 1, 2, ..., $N - 1$ 의 순서이다. 각 장소 i ($1 \leq i \leq N - 1$)에 도착하면 연구팀은 아래 두 작업을 진행한다.

- 장소 i 로 공룡들이 직접 이주한 다른 장소의 번호, 즉, $P[i]$ 의 값을 알아낸다.
- 현재까지 수집된 정보들을 이용하여 박물관으로 **하나의** 메시지를 보낼 지 말지 결정한다.

메세지는 인공위성을 통해 전달되어 그 비용이 매우 비싸다. 그러한 이유로 각 메세지는 1 이상 20 000 이하의 정수 하나이다. 추가로, 연구팀이 전체 과정에서 보낼 수 있는 메세지는 **최대 50개**이다.

당신이 할 일은 아래 목표를 달성하는 전략을 구현하는 것이다.

- 연구팀은 장소들 중 어떤 곳에서 어떤 내용의 메시지를 보낼 지 결정한다.
- 박물관은 받은 메시지들만을 이용해서 어떤 쌍의 장소 간의 거리가 가장 먼지 파악할 수 있다. 박물관은 각 메시지가 어떤 장소에서 보내진 것인지 알 수 있다.

인공위성을 이용한 통신은 매우 비싸서, 당신의 해결책은 메시지들에 저장된 가장 큰 정수의 값과, 보낸 메시지의 총 개수에 따라 다른 점수를 받을 것이다.

Implementation Details

두 개의 함수를 구현해야 한다. 하나는 연구팀을 위한 것이고, 다른 하나는 박물관을 위한 것이다.

연구팀을 위한 함수는 아래와 같다.

```
int send_message(int N, int i, int Pi)
```

- N : 장소의 개수.
- i : 현재 방문 중인 장소의 번호.
- $P_i: P[i]$ 의 값.
- 이 함수는 각 테스트 케이스에 대해 $N - 1$ 번 호출된다. $i = 1, 2, \dots, N - 1$ 의 순서로 호출된다.

이 함수는 이 장소에서 연구팀의 결과를 알려주는 $S[i]$ 를 리턴해야 한다.

- $S[i] = 0$: 장소 i 에서 메시지를 보내지 않았다.
- $1 \leq S[i] \leq 20\,000$: 연구팀은 장소 i 에서 $S[i]$ 를 메시지로 전송했다.

박물관을 위한 함수는 아래와 같다.

```
std::pair<int,int> longest_path(std::vector<int> S)
```

- S : 길이 N 인 배열이고, 그 내용은 아래와 같다.
 - $S[0] = 0$.
 - 각 i ($1 \leq i \leq N - 1$)에 대해 $S[i]$ 는 `send_message(N, i, Pi)`가 리턴한 값이다.
- 이 함수는 각 테스트 케이스에 대해 정확히 한 번 호출된다..

이 함수는 거리가 가장 먼 장소의 쌍 (U, V) 를 리턴해야 한다.

실제 채점 과정에서 위 두 함수를 호출하는 프로그램은 **정확히 두 번** 실행된다.

- 첫번째 실행에서는,
 - `send_message`가 정확히 $N - 1$ 번 호출된다.
 - 당신이 작성한 프로그램은 연속한 함수 호출이 진행되는 동안 정보를 저장하고 활용하는 것이 가능하다.
 - 리턴된 값들(배열 S)이 채점 시스템에 저장된다.

- 그레이더는 **적응적**으로 행동할 수 있다. 즉, `send_message` 함수 호출의 Pi 인자의 값이 이전 호출의 행동에 따라 달라질 수 있다.
- 두번째 실행에서는:
 - `longest_path` 함수가 정확히 한 번 호출된다. `longest_path` 함수가 사용할 수 있는 정보는, 첫번째 실행에서 저장된 배열 S 뿐이다.

Constraints

- $N = 10\,000$
- $0 \leq P[i] < i$ ($1 \leq i \leq N - 1$).

Subtasks and Scoring

Subtask	Score	Additional Constraints
1	30	장소 0 과, 다른 어떤 장소 간의 거리가 가장 큰 거리이다.
2	70	추가적인 제한이 없다.

Z 는 배열 S 에 등장하는 가장 큰 정수의 값이고 M 은 전체 메세지의 개수라고 하자.

각 테스트 케이스에서 아래 조건들 중 하나라도 사실이면 점수는 0점이 된다. (CMS에서는 `Output isn't correct`로 결과가 나온다.):

- S 의 원소 중 하나라도 규칙을 어겼다.
- $Z > 20\,000$ 혹은 $M > 50$.
- `longest_path` 함수의 리턴 값이 정답이 아니다.

그렇지 않은 경우, subtask 1에 대한 점수는 아래와 같이 계산된다.

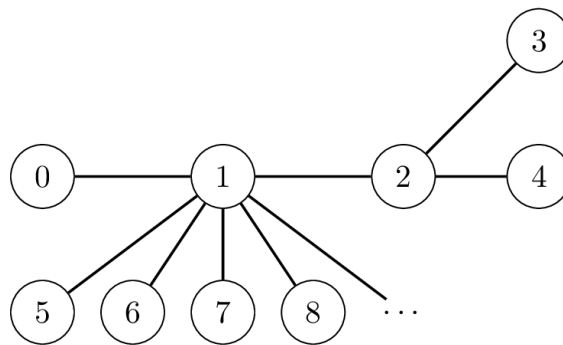
Condition	Score
$9\,998 \leq Z \leq 20\,000$	10
$102 \leq Z \leq 9997$	16
$5 \leq Z \leq 101$	23
$Z \leq 4$	30

Subtask 2에 대한 점수는 아래와 같이 계산된다.

Condition	Score
$5 \leq Z \leq 20\,000, M \leq 50$	$35 - 25 \log_{4000} \left(\frac{Z}{5} \right)$
$Z \leq 4$ and $32 \leq M \leq 50$	40
$Z \leq 4$ and $9 \leq M \leq 31$	$70 - 30 \log_4 \left(\frac{M}{8} \right)$
$Z \leq 4$ and $M \leq 8$	70

Example

$N = 10\,000$ 이고 $P[1] = 0, P[2] = 1, P[3] = 2, P[4] = 2$ 이며 $P[i] = 1$ ($i > 4$)인 경우를 생각하자.



연구팀의 작전이, `send_message` 호출에 의해 알려진 최대 거리가 더 커지게 만드는 장소 쌍 (U, V) 를 알게 될 때마다 메시지 $10 \cdot V + U$ 를 보내는 것이라고 하자.

초기에 연구팀이 알고 있는 최대 거리인 장소 쌍은 $(U, V) = (0, 0)$ 이다. 채점 서버에서 프로그램을 첫번째로 돌릴 때 아래와 같은 일이 일어난다.

함수 호출	(U, V)	리턴 값 ($S[i]$)
<code>send_message(10000, 1, 0)</code>	$(0, 1)$	10
<code>send_message(10000, 2, 1)</code>	$(0, 2)$	20
<code>send_message(10000, 3, 2)</code>	$(0, 3)$	30
<code>send_message(10000, 4, 2)</code>	$(0, 3)$	0

이후의 모든 호출에서 $P_i = 1$ 이라는 것에 주의하라. 즉, 이후에는 최대 거리를 만드는 장소 쌍이 바뀌지 않으므로 연구팀은 메시지를 더이상 보내지 않는다.

이제, 프로그램의 두번째 실행에서 다음 호출이 일어난다.

```
longest_path([0, 10, 20, 30, 0, ...])
```

박물관은 연구팀이 보낸 마지막 메시지인 $S[3] = 30$ 을 읽고 $(0, 3)$ 이 최대 거리를 만드는 장소 쌍임을 알 수 있다. 이 호출은 $(0, 3)$ 을 리턴한다.

이 방법으로 모든 경우에 정답을 찾지는 못함에 주의하라.

Sample Grader

Sample grader는 `send_message`와 `longest_path`를 한번의 실행에서 모두 호출한다. 이 동작은 채점 서버에서의 동작과 다르다.

입력 형식:

```
N
P[1] P[2] ... P[N-1]
```

출력 형식:

```
S[1] S[2] ... S[N-1]
U V
```

Sample grader를 사용하기 위해 지정하는 N 의 값은 임의의 값이 가능함에 주의하라.