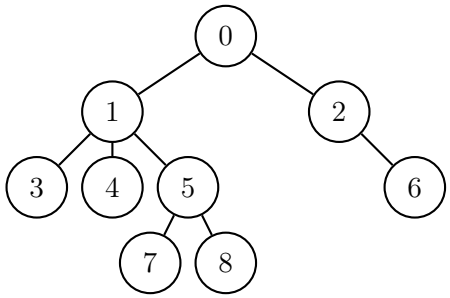
진화 2 (evolution2)

생명체의 진화 과정을 다루는 연구를 진행하고 있다. **최초 생명체**를 제외한 모든 생명체는 기존에 존재하던 생명체가 **진화**하여 새롭게 탄생한다. 이때, 기존에 존재하던 생명체를 **부모 생명체**, 새롭게 탄생한 생명체를 **자식 생명체**라고 정의한다.

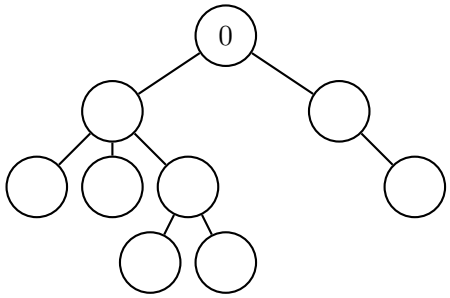
N개의 각 생명체에는 0 이상 $N-1$ 이하의 서로 다른 **탄생 번호**가 붙어 있다. 탄생 번호는 생명체가 탄생한 순서대로 부여된다. 이에 따라, 부모 생명체의 탄생 번호는 자식 생명체의 탄생 번호보다 작으며, 최초 생명체의 탄생 번호는 0이다.

생명체들이 진화를 통해 탄생하는 과정은 생명체를 정점으로, 진화 과정을 부모 생명체와 자식 생명체를 잇는 간선으로 나타내고, 최초 생명체를 루트로 한 트리 구조로 표현할 수 있다. 이러한 트리를 **진화 트리**라고 부르자.

예를 들어, 아래 그림은 탄생 번호가 0인 생명체가 진화해 탄생 번호가 1, 2인 생명체가 탄생하고, 탄생 번호가 1인 생명체가 진화해 탄생 번호가 3, 4, 5인 생명체가 탄생하고, 탄생 번호가 2인 생명체가 진화해 탄생 번호가 6인 생명체가 탄생하고, 탄생 번호가 5인 생명체가 진화해 탄생 번호가 7, 8인 생명체가 탄생하는 과정을 표현한 진화 트리이다. 아래 그림에서 트리의 각 정점에는 해당하는 생명체의 탄생 번호가 적혀 있다.

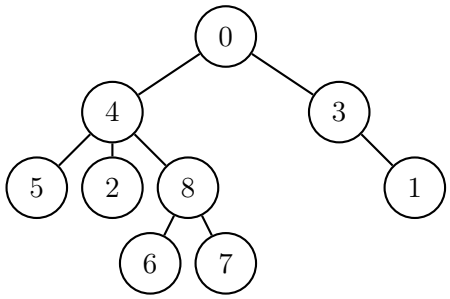


하지만 아뿔싸, 여러분의 소중한 진화 트리에 조영욱 코치가 커피를 쏟았다. 커피를 쏟은 이후, 진화 트리의 형태와 최초 생명체는 식별할 수 있으나, 최초 생명체를 제외한 각 생명체의 탄생 번호는 식별할 수 없게 되었다.



조영욱 코치는 급한 대로 각 생명체에 **임시 번호**를 매겼다. 최초 생명체의 임시 번호는 0으로 매겼고, 최초 생명체를 제외한 다른 $N-1$개의 생명체에는 각각 1 이상 $N-1$ 이하의 서로 다른 번호를 임의로 매겼다. 각 생명체의 임시 번호는 탄생 번호와 다를 수도, 같을 수도 있다. 또한, 탄생 번호와는 달리 부모 생명체의 임시 번호가 자식 생명체의 임시 번호보다 작다는 보장은 없다.

예를 들어, 아래 그림은 위에서 예시로 든 진화 트리와 같으나, 트리의 각 정점에는 해당하는 생명체의 임시 번호가 적혀 있다.



다행히도 진화 트리의 백업본이 조영욱 코치의 노트북에 저장되어 있었다. 하지만 백업본은 암호화되어 있고, 조영욱 코치는 백업본의 암호를 기억하지 못하고 있다. 대신, 조영욱 코치는 백업본의 백업 프로그램을 사용할 수 있다.

백업 프로그램에는 두 생명체의 **임시 번호**를 입력하면, 두 생명체 중 어느 생명체의 **탄생 번호**가 더 작은지 알려주는 기능이 있다. 이 기능을 너무 많이 사용한다면 조영욱 코치의 성능 낮은 노트북이 고장날 수도 있으므로, 여러분은 조영욱 코치를 도와 적은 횟수의 질문으로 진화 트리의 모든 생명체의 탄생 번호를 복구하는 코드를 작성해야 한다.

함수 목록 및 정의

여러분은 아래 함수를 구현해야 한다.

```
std::vector<int> recover(int N, std::vector<int> U, std::vector<int> V)
```

- 이 함수는 한 번의 실행에서 **1번 이상 호출된다.**
- U, V: 크기가 $N-1$인 정수 배열. 모든 정수 $0 \le i \le N-2$에 대해, 진화 트리에서 임시 번호가 $U[i]$인 생명체가 부모 생명체이고, 임시 번호가 $V[i]$인 생명체가 자식 생명체임을 뜻한다. 모든 $V[i]$ 는 서로 다르다.
- 여러분은 매 함수 호출마다 추후에 정의될 `compare` 함수를 0회 이상 호출하여 진화 트리에서의 각 생명체의 탄생 번호를 찾아서 크기 N의 `std::vector`에 담아서 반환해야 한다. 함수가 반환한 `std::vector`를 X라고 할 때, 모든 i $(0 \le i \le N-1)$에 대해, 임시 번호가 i인 생명체의 탄생 번호는 $X[i]$여야 한다.

프로그램은 아래 함수를 호출할 수 있다.

```
int compare(int a, int b)
```

- a와 b는 $0 \le a, b \le N-1$이고 $a \neq b$인 두 정수여야 한다.
- 임시 번호가 a인 생명체의 탄생 번호가 임시 번호가 b인 생명체의 탄생 번호보다 작다면 1, 그렇지 않다면 0을 반환한다.

제출하는 소스 코드의 어느 부분에서도 입출력 함수를 실행해서는 안 된다.

제약 조건

- $2 \le N \le 10\,000$
- 모든 $0 \le i \le N-2$에 대해:
  - $0 \le U[i] \le N-1$
  - $1 \le V[i] \le N-1$
  - $U[i] \ne V[i]$
- 주어지는 입력은 올바른 진화 트리를 구성한다.
- 한 번의 실행에서의 모든 `recover` 호출에 대해 N의 합은 $10\,000$ 이하이다.
- 이 문제에서 그레이더는 적응적이지 않다(NOT adaptive). 이것은 각 생명체의 탄생 번호가 그레이더의 수행 초기에 고정되어 `compare` 함수 호출에 따라 변하지 않음을 의미한다.

부분문제

1. (1점)
   - 세 개 이상의 생명체와 진화 트리 상에서 인접한 생명체가 존재하지 않고, 최초 생명체는 하나의 생명체와만 인접하다.
2. (7점)
   - 모든 $0 \le i \le N-2$에 대해 $U[i] = 0$
3. (12점)
   - 진화 트리의 각 생명체에 서로 다른 색 c $(0 \le c \le N-1)$를 적절하게 새로 매기면, 모든 $1 \le i \le N-1$에 대해 색이 i인 생명체의 부모 생명체의 색이 $\left\lfloor \frac{i-1}{2} \right\rfloor$이게 할 수 있는 방법이 존재하고, $N = 2^k - 1$을 만족하는 어떤 양의 정수 k가 존재한다. 즉, 주어지는 진화 트리는 완전 이진 트리이다.
4. (80점)
   - 추가적인 제약 조건이 없다.

각각의 `recover` 함수 호출마다 아래와 같은 방식으로 점수를 매긴다. 각 부분문제의 점수는, 그 부분문제 내의 테스트 케이스들에서의 `recover` 호출이 받은 점수 중 최솟값이다.

만약, 프로그램이 비정상적으로 종료했거나, `recover` 함수의 반환값이 올바르지 않은 경우, 0점을 받는다.

`compare` 함수를 통한 생명체 번호의 대소 관계에 대한 정보나 부분문제에 의한 제약 조건을 모르는 상태에서 $0 \le i \le N-1$에 대해 임시 번호가 i인 생명체의 탄생 번호가 $X[i]$인 것이 문제 조건상 가능한 $0, 1, \cdots, N-1$의 순열 X의 개수를 P라고 하자. 문제의 정답 또한 가능한 X에 해당하므로, P는 양의 정수이다. $Z = \lceil \log_2 P \rceil$라고 하고, C를 이번 `recover` 함수 호출에서 여러분의 코드가 `compare` 함수를 호출한 횟수라고 하자. 점수는 $K = \frac{C}{Z}$에 의해 결정된다. $Z = 0$이어서 K가 정의되지 않는 경우 $C = 0$이면 $K = 0$, $C > 0$이면 $K = 2025$로 정의한다.

- $K > 20$이면 0점을 받는다.
- $8 < K \le 20$이면 해당 부분문제 점수의 $(5 \times \frac{20-K}{12} + 5)$퍼센트를 받는다.

- $2.5 < K \leq 8$이면 해당 부분문제 점수의 $(50 \times \frac{8-K}{5.5} + 10)$퍼센트를 받는다.
- $1.5 < K \leq 2.5$이면 해당 부분문제 점수의 $(20 \times (2.5 - K) + 60)$퍼센트를 받는다.
- $1.4 < K \leq 1.5$이면 해당 부분문제 점수의 $(10 \times \frac{1.5-K}{0.1} + 80)$퍼센트를 받는다.
- $K \leq 1.4$이면 해당 부분문제 점수의 100퍼센트를 받는다.

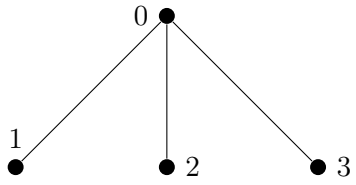
예제 1

$N = 4$, $U = [0, 0, 0]$, $V = [1, 2, 3]$인 경우를 생각해 보자.

그레이더는 다음과 같이 함수를 호출한다.

```
recover(4, [0, 0, 0], [1, 2, 3])
```

트리의 모양은 다음과 같다.



이 호출에서:

- 0번으로 임시 번호가 매겨진 생명체의 탄생 번호는 0 (다른 배정이 불가능하다),
- 1번으로 임시 번호가 매겨진 생명체의 탄생 번호는 2,
- 2번으로 임시 번호가 매겨진 생명체의 탄생 번호는 3,
- 3번으로 임시 번호가 매겨진 생명체의 탄생 번호는 1 이다.

아무 정보를 모르는 상태에서 문제의 답으로 가능한 순열은 $[0, 1, 2, 3]$, $[0, 1, 3, 2]$, $[0, 2, 1, 3]$, $[0, 2, 3, 1]$, $[0, 3, 1, 2]$, $[0, 3, 2, 1]$의 6개가 있으므로 $P = 6$, $Z = 3$이다.

`compare(1,2)`를 호출하면 $2 < 3$ 이기 때문에 1 이 반환된다.

`compare(2,3)`를 호출하면 $3 > 1$ 이기 때문에 0 이 반환된다.

`compare(1,3)`를 호출하면 $2 > 1$ 이기 때문에 0 이 반환된다.

`compare(0,3)`를 호출하면 $0 < 1$ 이기 때문에 1 이 반환된다.

4번의 호출을 통해 $[0, 2, 3, 1]$을 반환했다면 `recover` 함수의 반환값이 올바르다. 이 경우 `compare` 함수를 호출한 횟수가 4회이다. $K = \frac{C}{Z} = \frac{4}{3} \leq 1.4$이므로 해당 부분문제 점수의 100퍼센트를 받는다. 이 예시는 부분문제 2, 4의 조건을 만족한다.

Sample grader

Sample grader는 테스트 케이스의 개수 T를 입력 받는다. 그 이후 T회에 걸쳐 다음과 같은 정보를 입력받는다:

- Line 1: N
- Line 2: $PAR[1]\ PAR[2]\ \cdots\ PAR[N-1]$
- Line 3: $Y[0]\ Y[1]\ \cdots\ Y[N-1]$

$PAR[i]$는 진화 트리에서 탄생 번호가 i번인 생명체의 부모 생명체의 탄생 번호여야 한다. 문제 조건에 의해 $0 \leq PAR[i] < i$를 만족한다.

$Y[i]$는 진화 트리의 탄생 번호가 i번인 생명체에 매긴 임시 번호이다. 문제 조건에 의해 $Y[0] = 0$이고 Y는 $0, 1, \cdots, N-1$의 순열이어야 한다.

각 테스트 케이스마다 Sample grader는 주어진 진화 트리의 각 생명체의 탄생 번호와 임시 번호에 맞게 U와 V를 구성하여 `recover` 함수를 호출하고, 그 결과를 다음과 같이 출력한다.

- 만약 호출된 `compare(a,b)` 가 $0 \leq a, b \leq N-1$을 만족하지 않는 경우 한 줄에 `Wrong Answer [1]`를 출력한다.
- 만약 호출된 `compare(a,b)` 가 $a \neq b$을 만족하지 않는 경우 한 줄에 `Wrong Answer [2]`를 출력한다.
- `recover`함수가 반환한 배열의 크기가 N이 아닌 경우 한 줄에 `Wrong Answer [3]`를 출력한다.
- 그 외의 경우 첫 줄에 `compare` 함수의 총 호출 횟수 C를 $C : 4$의 형태로 출력하고 다음 줄에 `recover` 함수의 반환값 X의 원소를 순서대로 출력한다.

`Wrong Answer`를 출력할 시, Sample grader는 즉시 종료된다.

$0 \leq i \leq N-1$에 대해 $Y[X[i]] = i$를 만족하는 출력이 올바른 출력임과, Sample grader는 이를 확인하지 않음에 유의하라.

Sample grader는 실제 채점에서 사용하는 그레이더와 다를 수 있음에 유의하라.