광석 채굴

시간 제한	1초
메모리 제한	512 MB
사용 가능 언어	C, C++
C++ 허용 STL	전체

C++의 경우 main 함수 내의 시작 지점에 다음 내용을 추가하여 cin, cout 의 입출력 속도를 개선할 수 있다.

ios_base::sync_with_stdio(false);

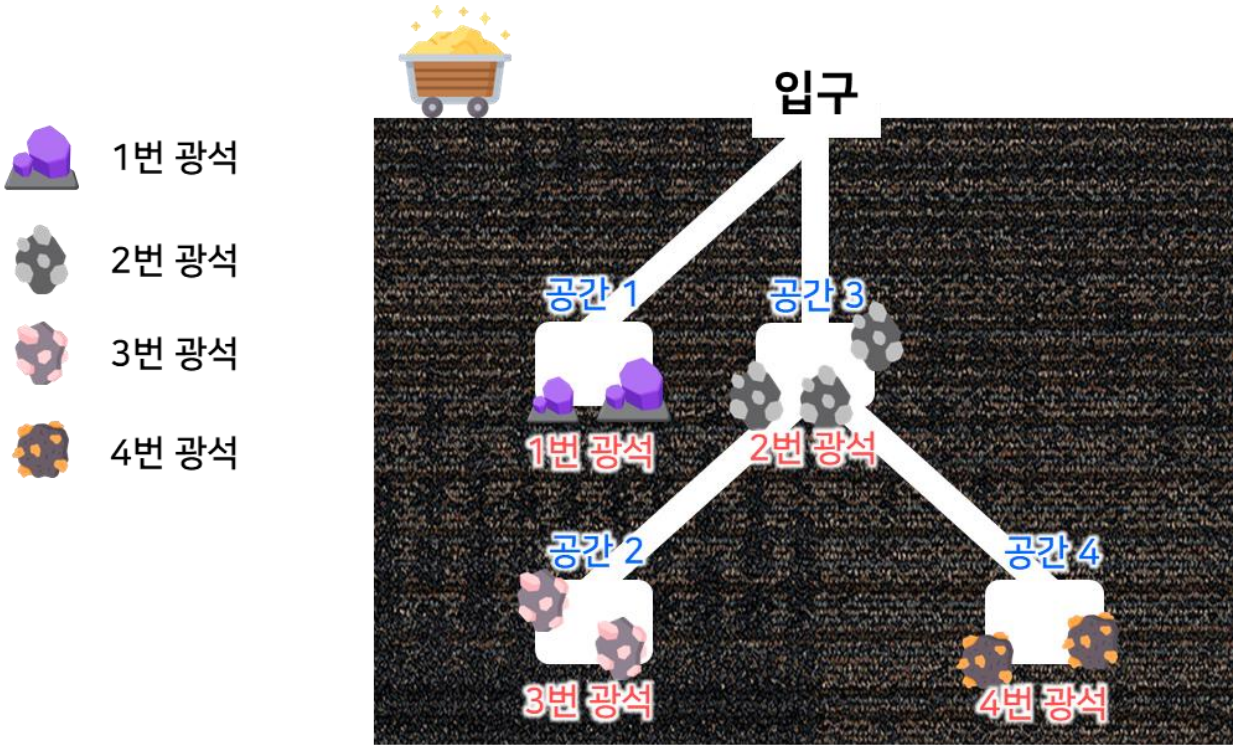
cin.tie(nullptr);

cout.tie(nullptr);

단, 위 내용을 추가할 경우 cin, cout 만 사용해야 하며, scanf, printf 등 C 입출력과 혼용해서 사용하면 안 된다.

C++ std::endl의 경우 출력 속도가 느리므로, cout<<endl; 대신 **cout<<"₩n";**을 사용하는 것을 권장한다.

비룡이는 인하 광산에 매장된 광석을 채굴하려고 한다. 인하 광산의 입구는 하나이며 광석이 매장된 모든 공간은 입구로부터 연결되어 있다. 광산 입구의 깊이는 0이고, 광산 내부의 모든 공간은 트리 구조로 연결되어 있으며 깊이가 1 단위로 구분되어 있다. 인하 광산의 각 공간은 서로 다른 광석이 존재한다. 인하 광산에 매장되어 있는 여러 광석 중에서 비룡이가 원하는 광석들이 매장되어 있는 공간들의 깊이를 출력해보자.



예를 들어, 인하 광산의 형태가 그림과 같이 공간 1, 2, 3, 4로 구분되어 있고 각 공간에 광석이 하나씩 매장되어 있을 때 비룡이가 원하는 광석이 1번, 3번 광석일 경우, 입구로부터 깊이가 1만큼 떨어진 공간 1에 1번 광석이 있고, 깊이가 2만큼 떨어진 공간 2에 3번 광석이 있으므로 1번, 3번 광석이 매장되어 있는 깊이는 각각 1, 2이다.

입력

첫 번째 줄에는 테스트 케이스 수 T $(1 \le T \le 10)$가 주어진다. 이후 각 테스트 케이스의 정보가 다음과 같이 주어진다.

- 첫 번째 줄에 광산에 있는 내부 공간의 수 n $(1 \le n \le 100{,}000)$과 비룡이가 원하는 광석의 수 k $(1 \le k \le n)$가 공백으로 구분되어 주어진다. n개의 공간은 공간 1부터 n까지 순서대로 번호가 부여되어 있으며, 각 공간마다 하나의 광석이 매장되어 있다.
- 두 번째 줄에는 공간 1부터 n까지 각 공간에 매장된 n개의 광석 번호 w_i $(1 \le w_i \le n,\ 1 \le i \le n)$가 공백으로 구분되어 주어진다. 즉, i번째로 주어지는 광석의 번호 w_i는 공간 i에 w_i번 광석이 매장되어 있음을 의미한다.
- 세 번째 줄에 비룡이가 원하는 k개의 광석 번호가 공백으로 구분되어 주어진다. 각 번호는 중복되게 주어지지 않으며, 존재하는 광석의 번호로만 주어진다.
- 네 번째 줄부터 n개의 줄에 걸쳐, 인하 광산 내부 공간의 번호 a, b가 공백으로 구분되어 주어진다. 이는 공간 a와 공간 b가 서로 이어져 있음을 의미하며, 이때 공간 b의 깊이는 공간 a의 깊이보다 1만큼 더 깊다. 단, 중복된 입력은 주어지지 않으며, 광산 입구의 깊이는 0이고 공간 번호는 0으로 주어진다.

출력

각 테스트 케이스마다 비룡이가 원하는 k개의 광석에 대하여 각 광석이 매장된 깊이를 공백으로 구분하여 한 줄에 출력한다.

예제 입출력

예제 입력	예제 출력
2 4 2 1 3 2 4 1 3 0 1 0 3 3 2 3 4 9 3 2 3 7 4 6 8 9 1 5 1 7 2 0 3 2 9 6 7 4 8 2 1 0 6 4 2 9 5 0 4	1 2 2 1 3