**문제 설명**

스마트폰 전화 키패드의 각 칸에 다음과 같이 숫자들이 적혀 있습니다.



이 전화 키패드에서 왼손과 오른손의 엄지손가락만을 이용해서 숫자만을 입력하려고 합니다.  
맨 처음 왼손 엄지손가락은 \* 키패드에 오른손 엄지손가락은 # 키패드 위치에서 시작하며, 엄지손가락을 사용하는 규칙은 다음과 같습니다.

1. 엄지손가락은 상하좌우 4가지 방향으로만 이동할 수 있으며 키패드 이동 한 칸은 거리로 1에 해당합니다.
2. 왼쪽 열의 3개의 숫자 1, 4, 7을 입력할 때는 왼손 엄지손가락을 사용합니다.
3. 오른쪽 열의 3개의 숫자 3, 6, 9를 입력할 때는 오른손 엄지손가락을 사용합니다.
4. 가운데 열의 4개의 숫자 2, 5, 8, 0을 입력할 때는 두 엄지손가락의 현재 키패드의 위치에서 더 가까운 엄지손가락을 사용합니다.  
   4-1. 만약 두 엄지손가락의 거리가 같다면, 오른손잡이는 오른손 엄지손가락, 왼손잡이는 왼손 엄지손가락을 사용합니다.

순서대로 누를 번호가 담긴 배열 numbers, 왼손잡이인지 오른손잡이인 지를 나타내는 문자열 hand가 매개변수로 주어질 때, 각 번호를 누른 엄지손가락이 왼손인 지 오른손인 지를 나타내는 연속된 문자열 형태로 return 하도록 solution 함수를 완성해주세요.

**[제한사항]**

* numbers 배열의 크기는 1 이상 1,000 이하입니다.
* numbers 배열 원소의 값은 0 이상 9 이하인 정수입니다.
* hand는 "left" 또는 "right" 입니다.
  + "left"는 왼손잡이, "right"는 오른손잡이를 의미합니다.
* 왼손 엄지손가락을 사용한 경우는 L, 오른손 엄지손가락을 사용한 경우는 R을 순서대로 이어붙여 문자열 형태로 return 해주세요.

**입출력 예**

| **numbers** | **hand** | **result** |
| --- | --- | --- |
| [1, 3, 4, 5, 8, 2, 1, 4, 5, 9, 5] | "right" | "LRLLLRLLRRL" |
| [7, 0, 8, 2, 8, 3, 1, 5, 7, 6, 2] | "left" | "LRLLRRLLLRR" |
| [1, 2, 3, 4, 5, 6, 7, 8, 9, 0] | "right" | "LLRLLRLLRL" |

**입출력 예에 대한 설명**

**입출력 예 #1**

순서대로 눌러야 할 번호가 [1, 3, 4, 5, 8, 2, 1, 4, 5, 9, 5]이고, 오른손잡이입니다.

| **왼손 위치** | **오른손 위치** | **눌러야 할 숫자** | **사용한 손** | **설명** |
| --- | --- | --- | --- | --- |
| \* | # | 1 | L | 1은 왼손으로 누릅니다. |
| 1 | # | 3 | R | 3은 오른손으로 누릅니다. |
| 1 | 3 | 4 | L | 4는 왼손으로 누릅니다. |
| 4 | 3 | 5 | L | 왼손 거리는 1, 오른손 거리는 2이므로 왼손으로 5를 누릅니다. |
| 5 | 3 | 8 | L | 왼손 거리는 1, 오른손 거리는 3이므로 왼손으로 8을 누릅니다. |
| 8 | 3 | 2 | R | 왼손 거리는 2, 오른손 거리는 1이므로 오른손으로 2를 누릅니다. |
| 8 | 2 | 1 | L | 1은 왼손으로 누릅니다. |
| 1 | 2 | 4 | L | 4는 왼손으로 누릅니다. |
| 4 | 2 | 5 | R | 왼손 거리와 오른손 거리가 1로 같으므로, 오른손으로 5를 누릅니다. |
| 4 | 5 | 9 | R | 9는 오른손으로 누릅니다. |
| 4 | 9 | 5 | L | 왼손 거리는 1, 오른손 거리는 2이므로 왼손으로 5를 누릅니다. |
| 5 | 9 | - | - |  |

따라서 "LRLLLRLLRRL"를 return 합니다.

**입출력 예 #2**

왼손잡이가 [7, 0, 8, 2, 8, 3, 1, 5, 7, 6, 2]를 순서대로 누르면 사용한 손은 "LRLLRRLLLRR"이 됩니다.

**입출력 예 #3**

오른손잡이가 [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]를 순서대로 누르면 사용한 손은 "LLRLLRLLRL"이 됩니다.

**문제 설명**

IT 벤처 회사를 운영하고 있는 라이언은 매년 사내 해커톤 대회를 개최하여 우승자에게 상금을 지급하고 있습니다.  
이번 대회에서는 우승자에게 지급되는 상금을 이전 대회와는 다르게 다음과 같은 방식으로 결정하려고 합니다.  
해커톤 대회에 참가하는 모든 참가자들에게는 숫자들과 3가지의 연산문자(+, -, \*) 만으로 이루어진 연산 수식이 전달되며, 참가자의 미션은 전달받은 수식에 포함된 연산자의 우선순위를 자유롭게 재정의하여 만들 수 있는 가장 큰 숫자를 제출하는 것입니다.  
단, 연산자의 우선순위를 새로 정의할 때, 같은 순위의 연산자는 없어야 합니다. 즉, + > - > \* 또는 - > \* > + 등과 같이 연산자 우선순위를 정의할 수 있으나 +,\* > - 또는 \* > +,-처럼 2개 이상의 연산자가 동일한 순위를 가지도록 연산자 우선순위를 정의할 수는 없습니다. 수식에 포함된 연산자가 2개라면 정의할 수 있는 연산자 우선순위 조합은 2! = 2가지이며, 연산자가 3개라면 3! = 6가지 조합이 가능합니다.  
만약 계산된 결과가 음수라면 해당 숫자의 절댓값으로 변환하여 제출하며 제출한 숫자가 가장 큰 참가자를 우승자로 선정하며, 우승자가 제출한 숫자를 우승상금으로 지급하게 됩니다.

예를 들어, 참가자 중 네오가 아래와 같은 수식을 전달받았다고 가정합니다.

"100-200\*300-500+20"

일반적으로 수학 및 전산학에서 약속된 연산자 우선순위에 따르면 더하기와 빼기는 서로 동등하며 곱하기는 더하기, 빼기에 비해 우선순위가 높아 \* > +,- 로 우선순위가 정의되어 있습니다.  
대회 규칙에 따라 + > - > \* 또는 - > \* > + 등과 같이 연산자 우선순위를 정의할 수 있으나 +,\* > - 또는 \* > +,- 처럼 2개 이상의 연산자가 동일한 순위를 가지도록 연산자 우선순위를 정의할 수는 없습니다.  
수식에 연산자가 3개 주어졌으므로 가능한 연산자 우선순위 조합은 3! = 6가지이며, 그 중 + > - > \* 로 연산자 우선순위를 정한다면 결괏값은 22,000원이 됩니다.  
반면에 \* > + > - 로 연산자 우선순위를 정한다면 수식의 결괏값은 -60,420 이지만, 규칙에 따라 우승 시 상금은 절댓값인 60,420원이 됩니다.

참가자에게 주어진 연산 수식이 담긴 문자열 expression이 매개변수로 주어질 때, 우승 시 받을 수 있는 가장 큰 상금 금액을 return 하도록 solution 함수를 완성해주세요.

**[제한사항]**

* expression은 길이가 3 이상 100 이하인 문자열입니다.
* expression은 공백문자, 괄호문자 없이 오로지 숫자와 3가지의 연산자(+, -, \*) 만으로 이루어진 올바른 중위표기법(연산의 두 대상 사이에 연산기호를 사용하는 방식)으로 표현된 연산식입니다. 잘못된 연산식은 입력으로 주어지지 않습니다.
  + 즉, "402+-561\*"처럼 잘못된 수식은 올바른 중위표기법이 아니므로 주어지지 않습니다.
* expression의 피연산자(operand)는 0 이상 999 이하의 숫자입니다.
  + 즉, "100-2145\*458+12"처럼 999를 초과하는 피연산자가 포함된 수식은 입력으로 주어지지 않습니다.
  + "-56+100"처럼 피연산자가 음수인 수식도 입력으로 주어지지 않습니다.
* expression은 적어도 1개 이상의 연산자를 포함하고 있습니다.
* 연산자 우선순위를 어떻게 적용하더라도, expression의 중간 계산값과 최종 결괏값은 절댓값이 263 - 1 이하가 되도록 입력이 주어집니다.
* 같은 연산자끼리는 앞에 있는 것의 우선순위가 더 높습니다.

**입출력 예**

| **expression** | **result** |
| --- | --- |
| "100-200\*300-500+20" | 60420 |
| "50\*6-3\*2" | 300 |

**입출력 예에 대한 설명**

**입출력 예 #1**  
\* > + > - 로 연산자 우선순위를 정했을 때, 가장 큰 절댓값을 얻을 수 있습니다.  
연산 순서는 아래와 같습니다.  
100-200\*300-500+20  
= 100-(200\*300)-500+20  
= 100-60000-(500+20)  
= (100-60000)-520  
= (-59900-520)  
= -60420  
따라서, 우승 시 받을 수 있는 상금은 |-60420| = 60420 입니다.

**입출력 예 #2**  
- > \* 로 연산자 우선순위를 정했을 때, 가장 큰 절댓값을 얻을 수 있습니다.  
연산 순서는 아래와 같습니다.(expression에서 + 연산자는 나타나지 않았으므로, 고려할 필요가 없습니다.)  
50\*6-3\*2  
= 50\*(6-3)\*2  
= (50\*3)\*2  
= 150\*2  
= 300  
따라서, 우승 시 받을 수 있는 상금은 300 입니다.

**문제 설명**

**[본 문제는 정확성과 효율성 테스트 각각 점수가 있는 문제입니다.]**

개발자 출신으로 세계 최고의 갑부가 된 어피치는 스트레스를 받을 때면 이를 풀기 위해 오프라인 매장에 쇼핑을 하러 가곤 합니다.  
어피치는 쇼핑을 할 때면 매장 진열대의 특정 범위의 물건들을 모두 싹쓸이 구매하는 습관이 있습니다.  
어느 날 스트레스를 풀기 위해 보석 매장에 쇼핑을 하러 간 어피치는 이전처럼 진열대의 특정 범위의 보석을 모두 구매하되 특별히 아래 목적을 달성하고 싶었습니다.  
진열된 모든 종류의 보석을 적어도 1개 이상 포함하는 가장 짧은 구간을 찾아서 구매

예를 들어 아래 진열대는 4종류의 보석(RUBY, DIA, EMERALD, SAPPHIRE) 8개가 진열된 예시입니다.

| **진열대 번호** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 보석 이름 | DIA | RUBY | **RUBY** | **DIA** | **DIA** | **EMERALD** | **SAPPHIRE** | DIA |

진열대의 3번부터 7번까지 5개의 보석을 구매하면 모든 종류의 보석을 적어도 하나 이상씩 포함하게 됩니다.

진열대의 3, 4, 6, 7번의 보석만 구매하는 것은 중간에 특정 구간(5번)이 빠지게 되므로 어피치의 쇼핑 습관에 맞지 않습니다.

진열대 번호 순서대로 보석들의 이름이 저장된 배열 gems가 매개변수로 주어집니다. 이때 모든 보석을 하나 이상 포함하는 가장 짧은 구간을 찾아서 return 하도록 solution 함수를 완성해주세요.  
가장 짧은 구간의 시작 진열대 번호와 끝 진열대 번호를 차례대로 배열에 담아서 return 하도록 하며, 만약 가장 짧은 구간이 여러 개라면 시작 진열대 번호가 가장 작은 구간을 return 합니다.

**[제한사항]**

* gems 배열의 크기는 1 이상 100,000 이하입니다.
  + gems 배열의 각 원소는 진열대에 나열된 보석을 나타냅니다.
  + gems 배열에는 1번 진열대부터 진열대 번호 순서대로 보석이름이 차례대로 저장되어 있습니다.
  + gems 배열의 각 원소는 길이가 1 이상 10 이하인 알파벳 대문자로만 구성된 문자열입니다.

**입출력 예**

| **gems** | **result** |
| --- | --- |
| ["DIA", "RUBY", "RUBY", "DIA", "DIA", "EMERALD", "SAPPHIRE", "DIA"] | [3, 7] |
| ["AA", "AB", "AC", "AA", "AC"] | [1, 3] |
| ["XYZ", "XYZ", "XYZ"] | [1, 1] |
| ["ZZZ", "YYY", "NNNN", "YYY", "BBB"] | [1, 5] |

**입출력 예에 대한 설명**

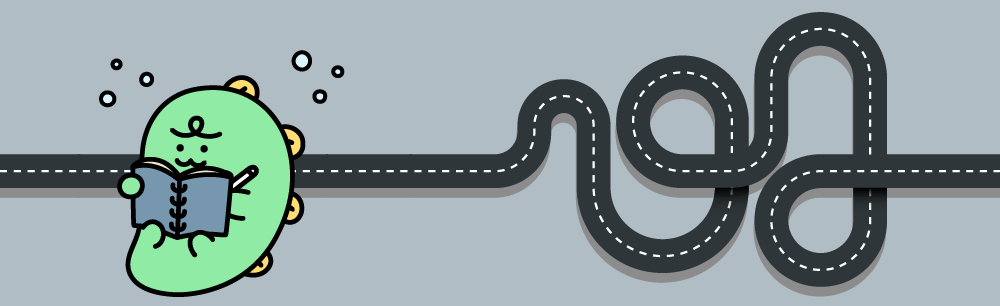
**입출력 예 #1**  
문제 예시와 같습니다.

**입출력 예 #2**  
3종류의 보석(AA, AB, AC)을 모두 포함하는 가장 짧은 구간은 [1, 3], [2, 4]가 있습니다.  
시작 진열대 번호가 더 작은 [1, 3]을 return 해주어야 합니다.

**입출력 예 #3**  
1종류의 보석(XYZ)을 포함하는 가장 짧은 구간은 [1, 1], [2, 2], [3, 3]이 있습니다.  
시작 진열대 번호가 가장 작은 [1, 1]을 return 해주어야 합니다.

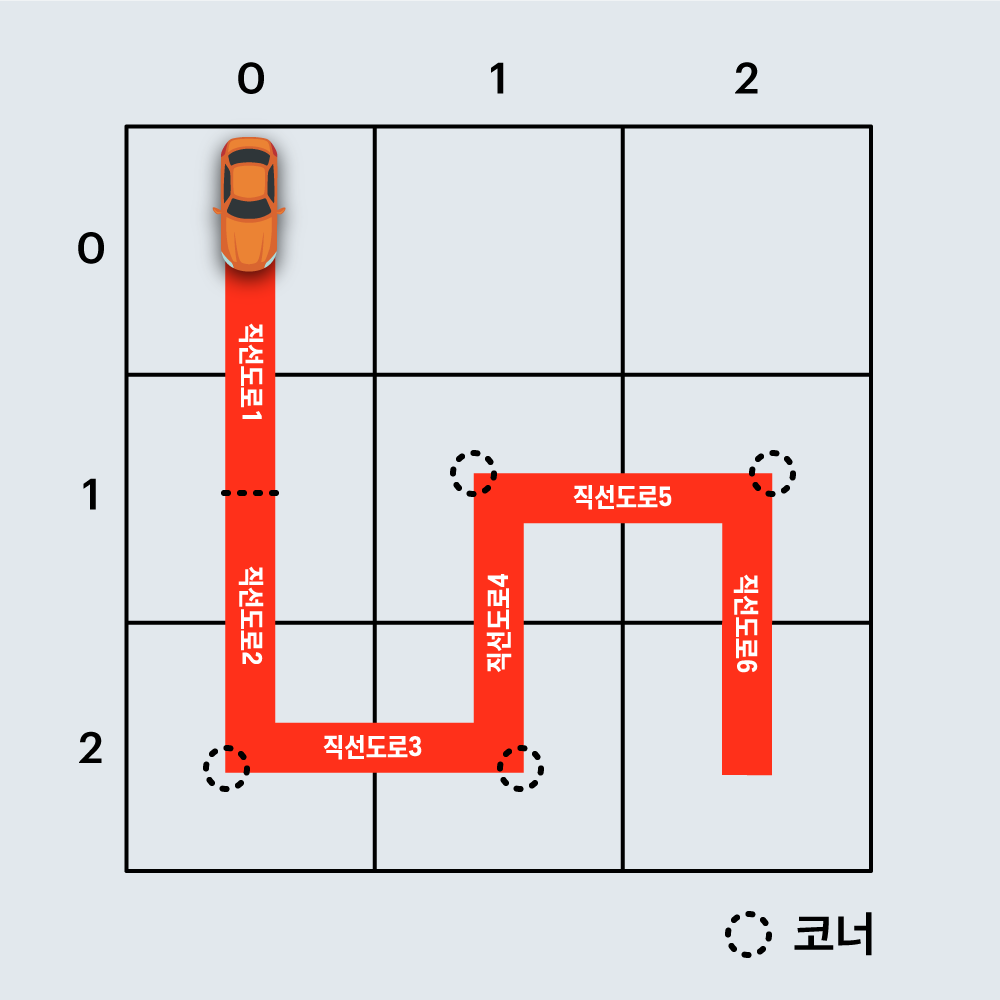
**입출력 예 #4**  
4종류의 보석(ZZZ, YYY, NNNN, BBB)을 모두 포함하는 구간은 [1, 5]가 유일합니다.  
그러므로 [1, 5]를 return 해주어야 합니다.

**문제 설명**

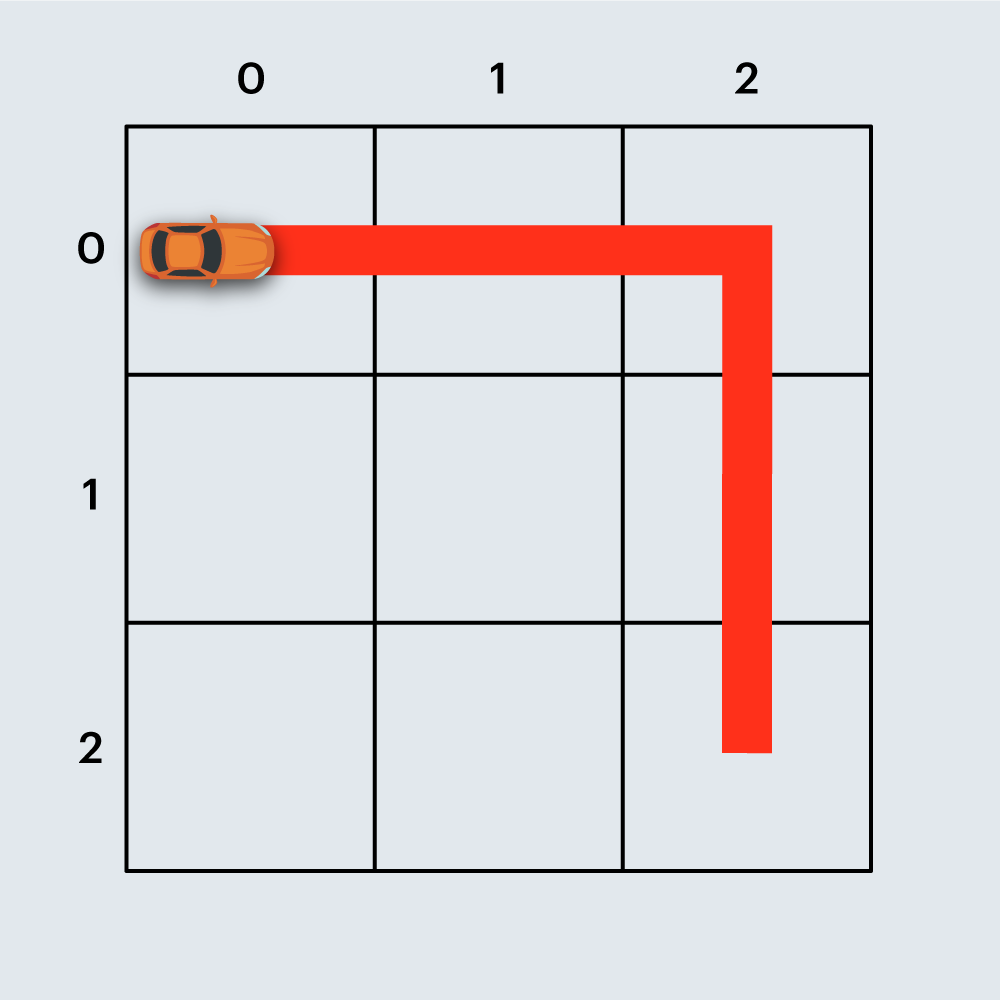


건설회사의 설계사인 죠르디는 고객사로부터 자동차 경주로 건설에 필요한 견적을 의뢰받았습니다.  
제공된 경주로 설계 도면에 따르면 경주로 부지는 N x N 크기의 정사각형 격자 형태이며 각 격자는 1 x 1 크기입니다.  
설계 도면에는 각 격자의 칸은 0 또는 1 로 채워져 있으며, 0은 칸이 비어 있음을 1은 해당 칸이 벽으로 채워져 있음을 나타냅니다.  
경주로의 출발점은 (0, 0) 칸(좌측 상단)이며, 도착점은 (N-1, N-1) 칸(우측 하단)입니다. 죠르디는 출발점인 (0, 0) 칸에서 출발한 자동차가 도착점인 (N-1, N-1) 칸까지 무사히 도달할 수 있게 중간에 끊기지 않도록 경주로를 건설해야 합니다.  
경주로는 상, 하, 좌, 우로 인접한 두 빈 칸을 연결하여 건설할 수 있으며, 벽이 있는 칸에는 경주로를 건설할 수 없습니다.  
이때, 인접한 두 빈 칸을 상하 또는 좌우로 연결한 경주로를 직선 도로 라고 합니다.  
또한 두 직선 도로가 서로 직각으로 만나는 지점을 코너 라고 부릅니다.  
건설 비용을 계산해 보니 직선 도로 하나를 만들 때는 100원이 소요되며, 코너를 하나 만들 때는 500원이 추가로 듭니다.  
죠르디는 견적서 작성을 위해 경주로를 건설하는 데 필요한 최소 비용을 계산해야 합니다.

예를 들어, 아래 그림은 직선 도로 6개와 코너 4개로 구성된 임의의 경주로 예시이며, 건설 비용은 6 x 100 + 4 x 500 = 2600원 입니다.



또 다른 예로, 아래 그림은 직선 도로 4개와 코너 1개로 구성된 경주로이며, 건설 비용은 4 x 100 + 1 x 500 = 900원 입니다.



도면의 상태(0은 비어 있음, 1은 벽)을 나타내는 2차원 배열 board가 매개변수로 주어질 때, 경주로를 건설하는데 필요한 최소 비용을 return 하도록 solution 함수를 완성해주세요.

**[제한사항]**

* board는 2차원 정사각 배열로 배열의 크기는 3 이상 25 이하입니다.
* board 배열의 각 원소의 값은 0 또는 1 입니다.
  + 도면의 가장 왼쪽 상단 좌표는 (0, 0)이며, 가장 우측 하단 좌표는 (N-1, N-1) 입니다.
  + 원소의 값 0은 칸이 비어 있어 도로 연결이 가능함을 1은 칸이 벽으로 채워져 있어 도로 연결이 불가능함을 나타냅니다.
* board는 항상 출발점에서 도착점까지 경주로를 건설할 수 있는 형태로 주어집니다.
* 출발점과 도착점 칸의 원소의 값은 항상 0으로 주어집니다.

**입출력 예**

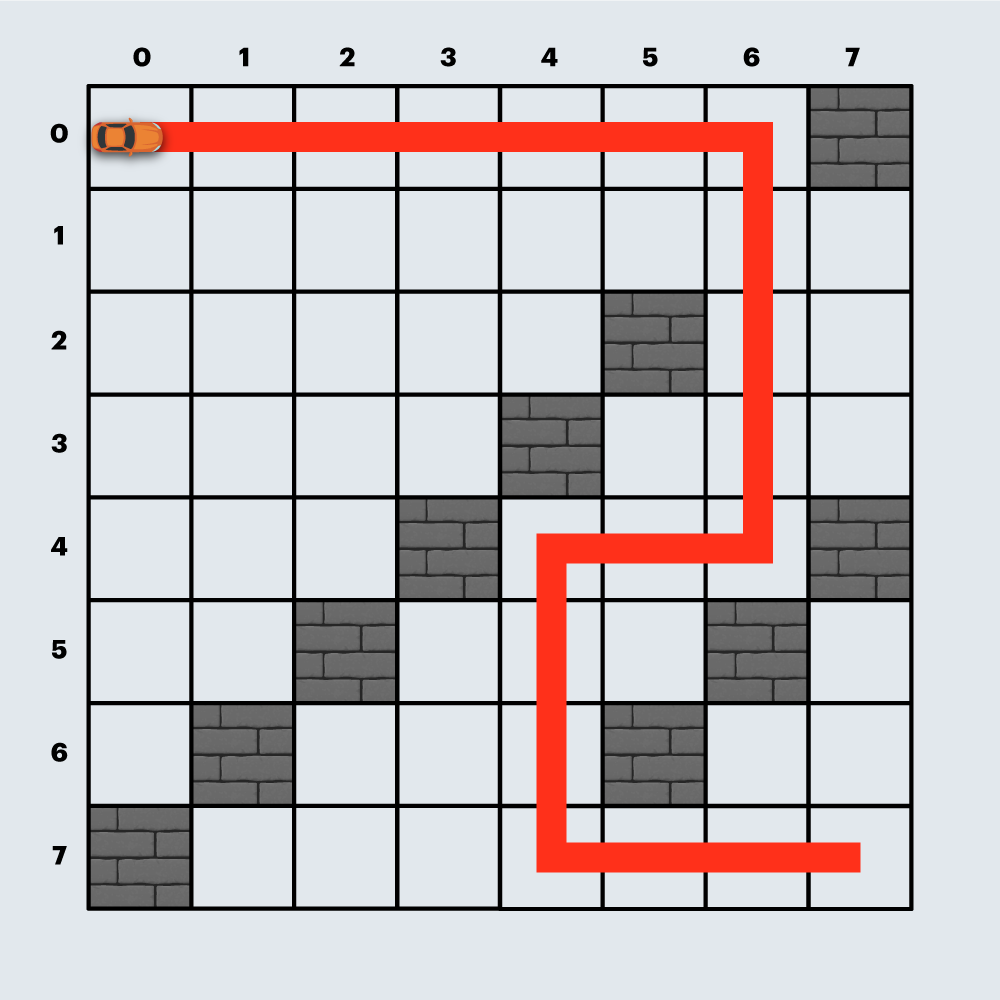
| **board** | **result** |
| --- | --- |
| [[0,0,0],[0,0,0],[0,0,0]] | 900 |
| [[0,0,0,0,0,0,0,1],[0,0,0,0,0,0,0,0],[0,0,0,0,0,1,0,0],[0,0,0,0,1,0,0,0],[0,0,0,1,0,0,0,1],[0,0,1,0,0,0,1,0],[0,1,0,0,0,1,0,0],[1,0,0,0,0,0,0,0]] | 3800 |
| [[0,0,1,0],[0,0,0,0],[0,1,0,1],[1,0,0,0]] | 2100 |
| [[0,0,0,0,0,0],[0,1,1,1,1,0],[0,0,1,0,0,0],[1,0,0,1,0,1],[0,1,0,0,0,1],[0,0,0,0,0,0]] | 3200 |

**입출력 예에 대한 설명**

**입출력 예 #1**

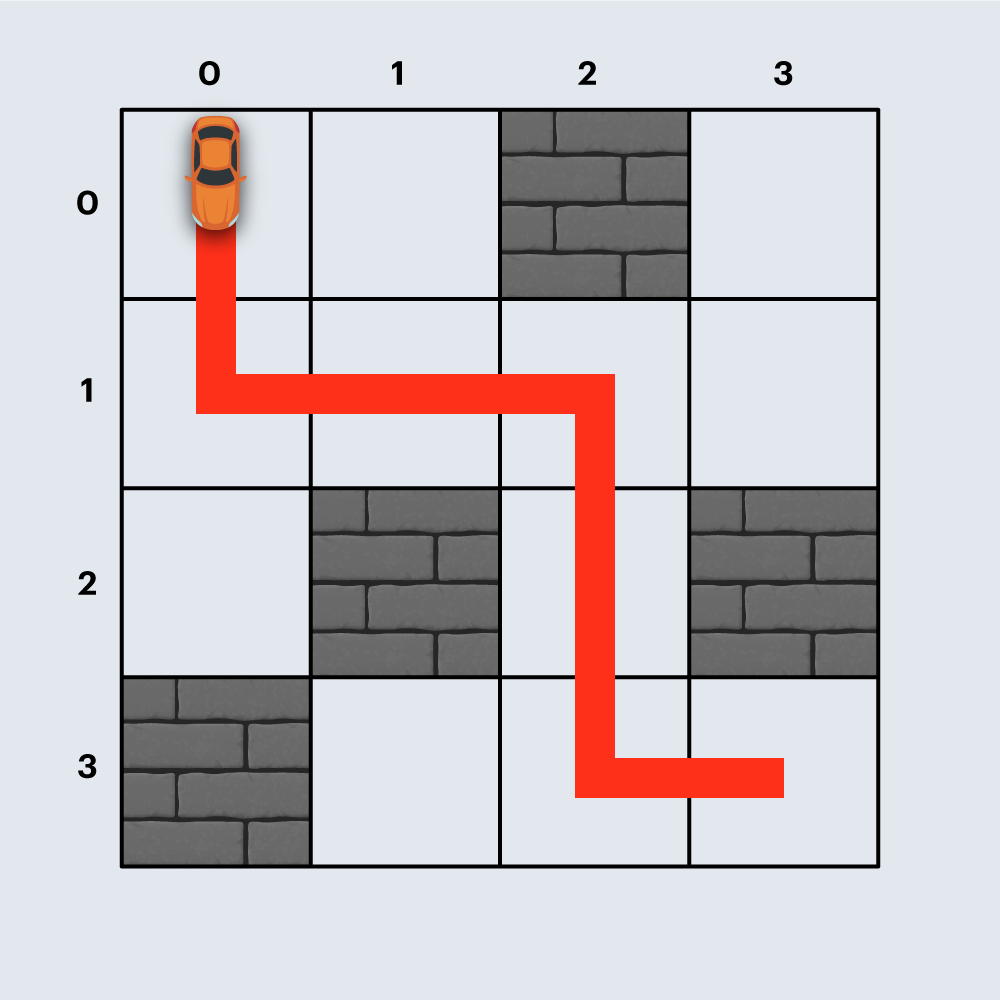
본문의 예시와 같습니다.

**입출력 예 #2**



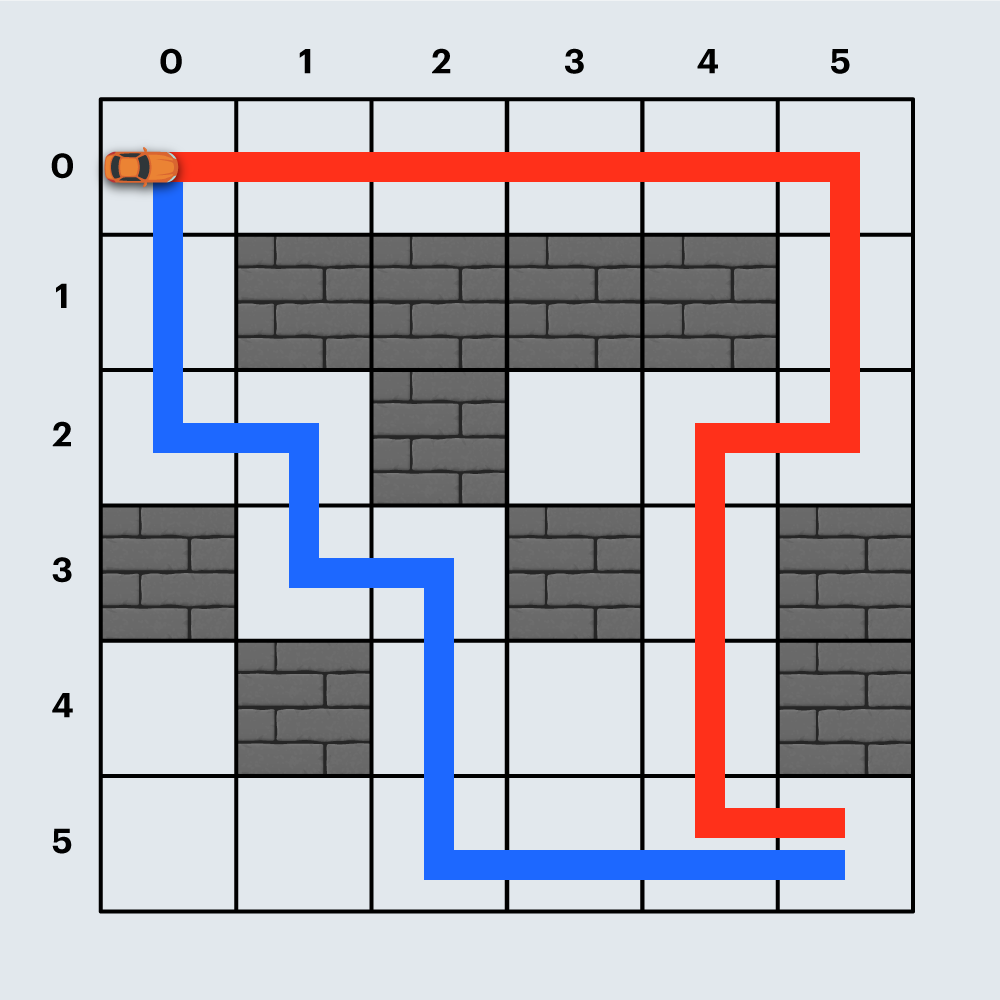
위와 같이 경주로를 건설하면 직선 도로 18개, 코너 4개로 총 3800원이 듭니다.

**입출력 예 #3**



위와 같이 경주로를 건설하면 직선 도로 6개, 코너 3개로 총 2100원이 듭니다.

**입출력 예 #4**



붉은색 경로와 같이 경주로를 건설하면 직선 도로 12개, 코너 4개로 총 3200원이 듭니다.  
만약, 파란색 경로와 같이 경주로를 건설한다면 직선 도로 10개, 코너 5개로 총 3500원이 들며, 더 많은 비용이 듭니다.

###### 문제 설명

**[본 문제는 정확성과 효율성 테스트 각각 점수가 있는 문제입니다.]**



오지 탐험가인 프로도는 탐험 도중 n개의 방으로 이루어진 지하 동굴을 탐험하게 되었습니다. 모든 방에는 0부터 n - 1 까지 번호가 붙어있고, 이 동굴에 들어갈 수 있는 유일한 입구는 0번 방과 연결되어 있습니다. 각 방들은 양방향으로 통행이 가능한 통로로 서로 연결되어 있는데, 서로 다른 두 방을 직접 연결하는 통로는 오직 하나입니다. 임의의 서로 다른 두 방 사이의 최단경로는 딱 한 가지만 있으며, 또한 임의의 두 방 사이에 이동이 불가능한 경우는 없습니다.

탐험에 앞서 이 지하 동굴의 지도를 손에 넣은 프로도는 다음과 같이 탐험 계획을 세웠습니다.

1. 모든 방을 적어도 한 번은 방문해야 합니다.
2. 특정 방은 방문하기 전에 반드시 먼저 방문할 방이 정해져 있습니다.  
   2-1. 이는 A번 방은 방문하기 전에 반드시 B번 방을 먼저 방문해야 한다는 의미입니다.  
   2-2. 어떤 방을 방문하기 위해 반드시 먼저 방문해야 하는 방은 없거나 또는 1개 입니다.  
   2-3. 서로 다른 두 개 이상의 방에 대해 먼저 방문해야 하는 방이 같은 경우는 없습니다.  
   2-4. 어떤 방이 먼저 방문해야 하는 방이면서 동시에 나중에 방문해야 되는 방인 경우는 없습니다.

위 계획 중 2-2, 2-3, 2-4는 순서를 지켜 방문해야 하는 두 방의 쌍이 A → B(A를 먼저 방문하고 B를 방문함) 형태로 유일함을 의미합니다. 즉, 프로도는 아래와 같은 형태로 방문순서가 잡히지 않도록 방문 계획을 세웠습니다.

* A → B, A → C (방문순서 배열 order = [...,[A,B],...,[A,C],...]) 형태로 A를 방문 후에 방문해야 할 방이 B와 C로 두 개 또는 그 이상인 경우
* X → A, Z → A (방문순서 배열 order = [...,[X,A],...,[Z,A],...]) 형태로 A를 방문하기 전에 방문해야 할 방이 X와 Z로 두 개 또는 그 이상­인 경우
* A → B → C (방문순서 배열 order = [...,[A,B],...,[B,C],...) 형태로 B처럼 A 방문 후이면서 동시에 C 방문 전인 경우

그리고 먼저 방문해야 할 방과 나중에 방문할 방을 반드시 연속해서 방문해야 할 필요는 없어 A방을 방문한 후 다른 방을 방문한 후 B방을 방문해도 좋습니다.

방 개수 n, 동굴의 각 통로들이 연결하는 두 방의 번호가 담긴 2차원 배열 path, 프로도가 정한 방문 순서가 담긴 2차원 배열 order가 매개변수로 주어질 때, 프로도가 규칙에 맞게 모든 방을 탐험할 수 있을 지 return 하도록 solution 함수를 완성해주세요.

**[제한사항]**

* n은 2 이상 200,000 이하입니다.
* path 배열의 세로(행) 길이는 n - 1 입니다.
  + path 배열의 원소는 [방 번호 A, 방 번호 B] 형태입니다.
  + 두 방 A, B사이를 연결하는 통로를 나타냅니다.
  + 통로가 연결하는 두 방 번호가 순서없이 들어있음에 주의하세요.
* order 배열의 세로(행) 길이는 1 이상 (n / 2) 이하입니다.
  + order 배열의 원소는 [방 번호 A, 방 번호 B] 형태입니다.
  + A번 방을 먼저 방문한 후 B번 방을 방문해야 함을 나타냅니다.

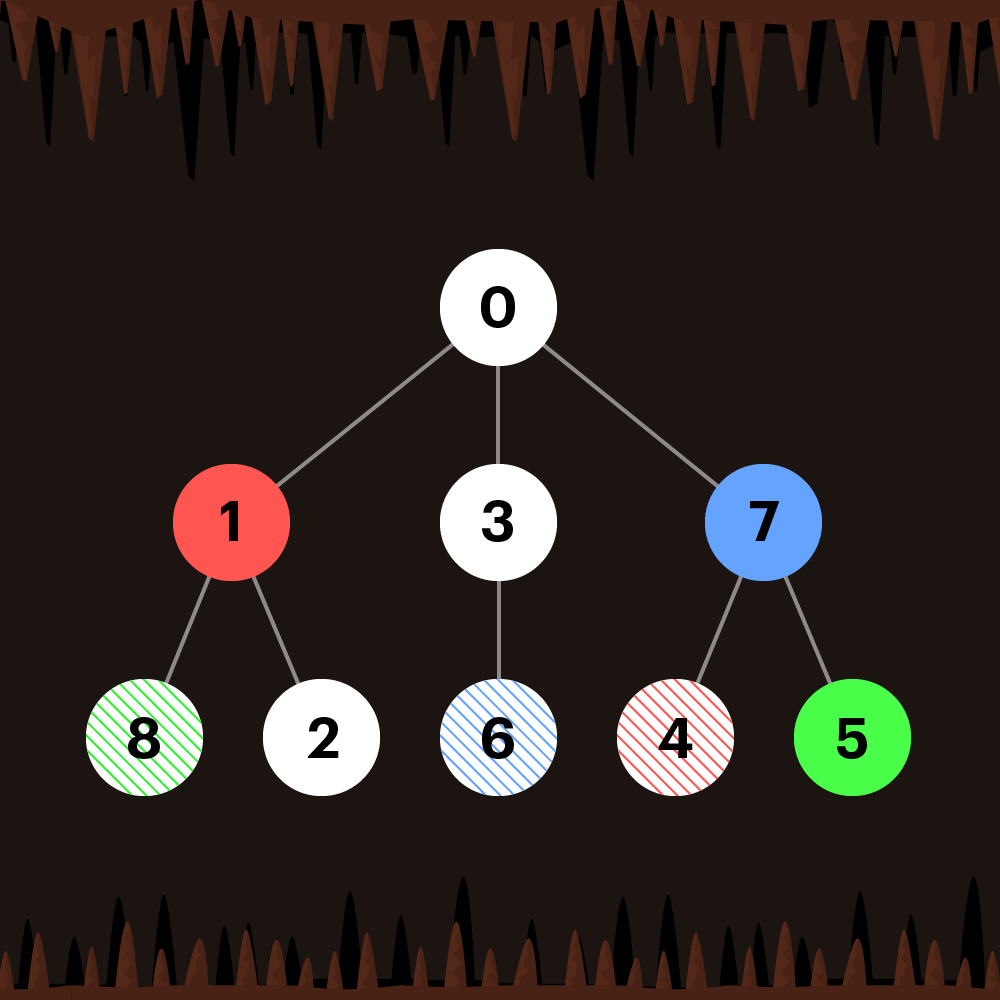
**입출력 예**

| **n** | **path** | **order** | **result** |
| --- | --- | --- | --- |
| 9 | [[0,1],[0,3],[0,7],[8,1],[3,6],[1,2],[4,7],[7,5]] | [[8,5],[6,7],[4,1]] | true |
| 9 | [[8,1],[0,1],[1,2],[0,7],[4,7],[0,3],[7,5],[3,6]] | [[4,1],[5,2]] | true |
| 9 | [[0,1],[0,3],[0,7],[8,1],[3,6],[1,2],[4,7],[7,5]] | [[4,1],[8,7],[6,5]] | false |

**입출력 예에 대한 설명**

**입출력 예 #1**

동굴 그림은 아래와 같습니다.



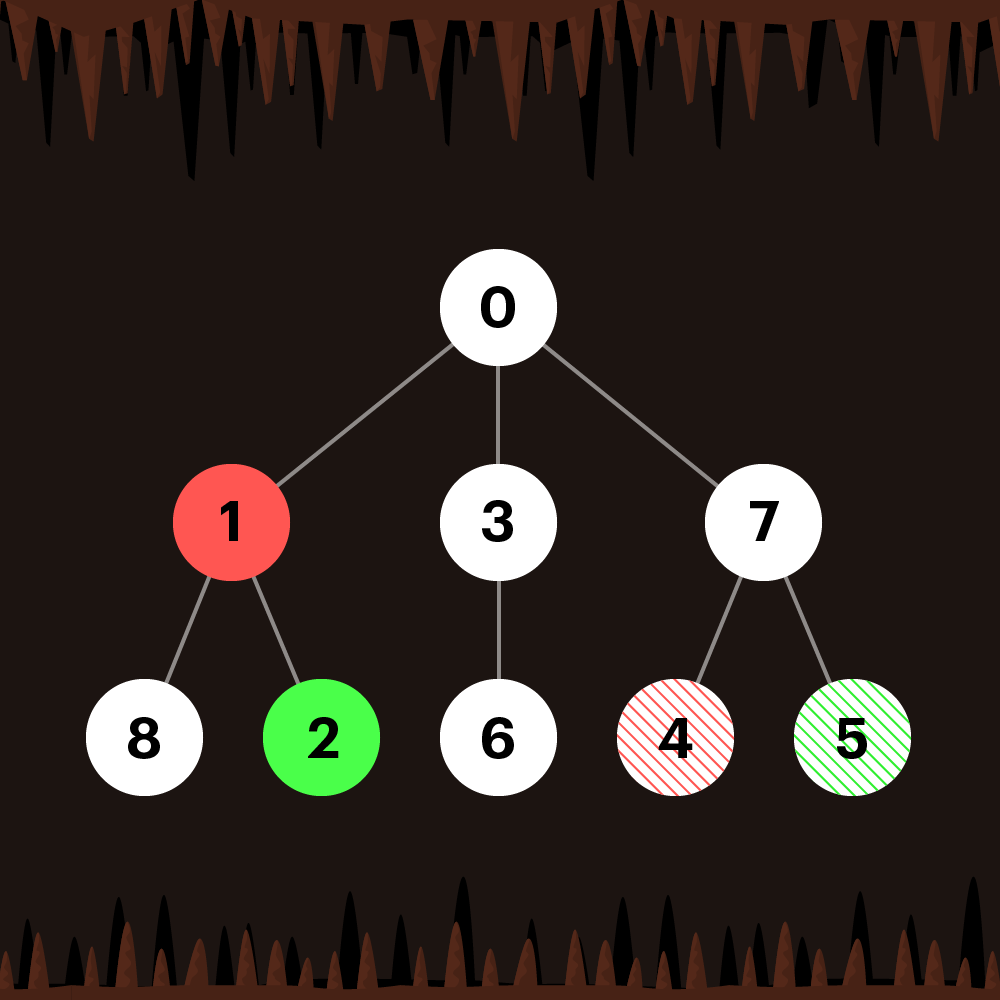
방문 순서를 지켜야 하는 방 번호는 다음과 같습니다.

* 6번 → 7번
* 4번 → 1번
* 8번 → 5번

따라서 모든 방을 방문할 수 있는 방법 중 하나는 다음과 같습니다.

* 0번 → 3번 → 6번 → 3번 → 0번 → 7번 → 4번 → 7번 → 0번 → 1번 → 8번 → 1번 → 2번 → 1번 → 0번 → 7번 → 5번

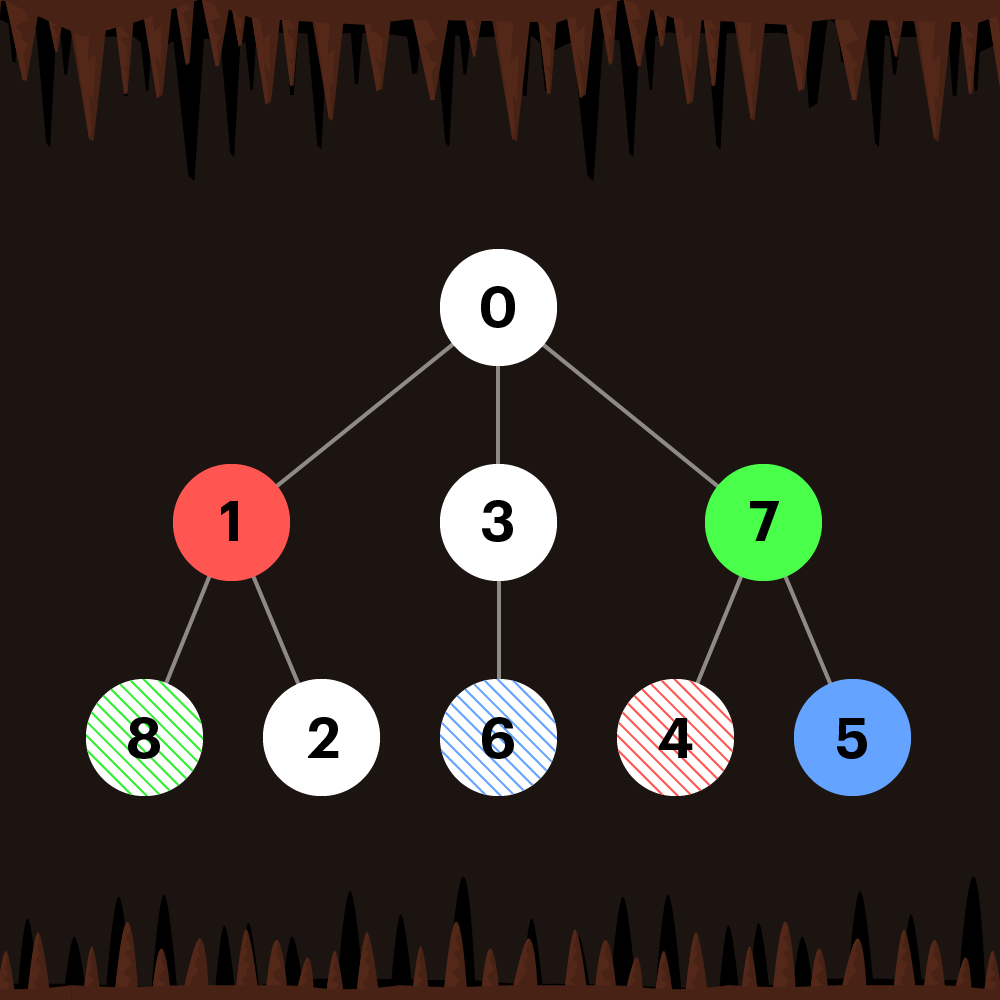
**입출력 예 #2**



다음 순서로 각 방을 방문하면 됩니다.

* 0번 → 7번 → 4번 → 7번 → 5번 → 7번 → 0번 → 3번 → 6번 → 3번 → 0번 → 1번 → 8번 → 1번 → 2번

**입출력 예 #3**



규칙에 맞게 모든 방을 방문할 수 있는 방법이 없습니다.