Algorithm

포화이진트리

[프로그래머스] 표현가능한 이진 트리

- 이진수를 이진 트리로 표현하는 것
- 값이 1이면 노드가 있고 값이 0이면 노드가 없음
- 왼쪽으로 갈수록 자릿수가 크고 오른쪽으로 갈수록 자릿수가 작음
- 왼쪽 자식노드는 부모노드보다 자릿수가 크고 오른쪽 자식노드는 부모노드보다 자릿수가 작음

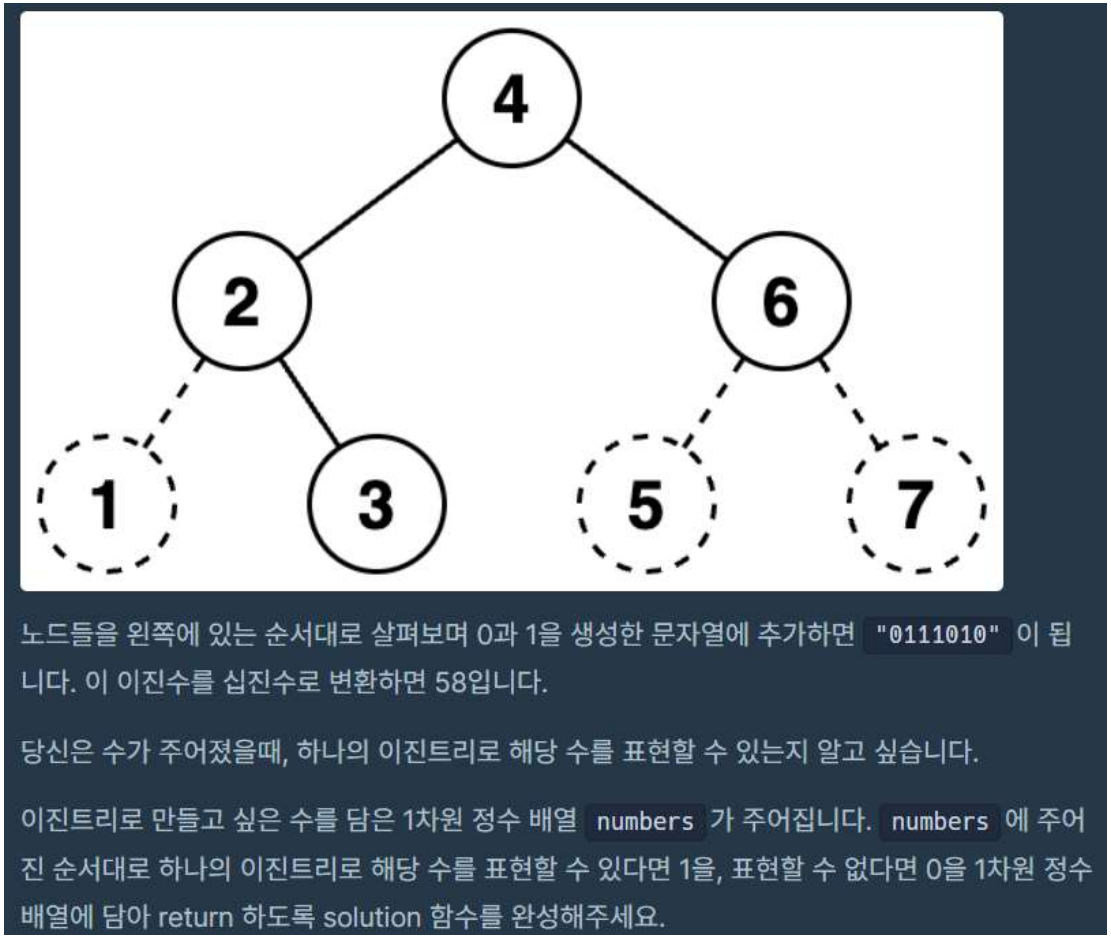
이진트리를 수로 표현하는 방법은 다음과 같습니다.

1. 이진수를 저장할 빈 문자열을 생성합니다.
2. 주어진 이진트리에 더미 노드를 추가하여 포화 이진트리로 만듭니다. **루트 노드는 그대로 유지합니다.**
3. 만들어진 포화 이진트리의 노드들을 가장 왼쪽 노드부터 가장 오른쪽 노드까지, 왼쪽에 있는 순서대로 살펴봅니다. **노드의 높이는 살펴보는 순서에 영향을 끼치지 않습니다.**
4. 살펴본 노드가 더미 노드라면, 문자열 뒤에 0을 추가합니다. 살펴본 노드가 더미 노드가 아니라면, 문자열 뒤에 1을 추가합니다.
5. 문자열에 저장된 이진수를 십진수로 변환합니다.

이진트리에서 리프 노드가 아닌 노드는 자신의 왼쪽 자식이 루트인 서브트리의 노드들보다 오른쪽에 있으며, 자신의 오른쪽 자식이 루트인 서브트리의 노드들보다 왼쪽에 있다고 가정합니다.

[프로그래머스] 표현가능한 이진 트리

- 왼쪽 자식노드는 부모노드보다 자릿수가 크고 오른쪽 자식노드는 부모노드보다 자릿수가 작음
- 이진트리가 존재할 수 있으면 1을 리턴, 존재할 수 없으면 0을 리턴
- 특정 자식노드가 0이 아닌데, 부모노드가 0일 경우 이진트리가 성립할 수 없음. 이런 경우는 0으로 리턴해야함.



포화이진트리 원리1: 노드의 개수

- 포화이진트리에서 트리의 노드의 개수는 h가 트리의 높이 일때 노드의 개수 = 2^h-1
- 100110과 같은 2진수는 노드의 개수가 6개이지만 포화 이진트리가 되기 위해서는 2^3-1인 7개의 노드가 필요하므로 2진수의 왼쪽에 0을 하나를 더 붙이면 된다.
- 10111011과 같은 2진수는 노드의 개수가 8개 이므로 포화이진트리가 되기 위해서 2^4-1 인 15개의 노드가 필요하므로 7개의 0을 왼쪽에 붙여주어 000000010111011과 같은 문자열을 만들어 주어야한다.

포화이진트리 원리1 활용: 문자열 셋팅

- 일단 주어진 숫자로 이진 문자열을 만들고 탐색에 유효한 이진 문자열로 셋팅해줘야한다.
- 이진 변환은 bin[2:]을 사용한다.
- 트리의 높이는 이진 문자열의 길이를 log2()하여 나온 지수를 사용
- 노드의 개수는 h가 트리의 높이일 때 노드의 개수 = 2^h-1
- 탐색에 유효하도록 현재의 이진 문자열이 노드의 개수를 충족하도록 좌측 빈 공간에는 빈 공간의 크기만큼 0을 삽입해준다.

```
for number in numbers:
    binary = bin(number)[2:]
    indices = int(log2(len(binary))) + 1
    digit = 2 ** indices - 1
    s = '0' * (digit - len(binary)) + binary
```

포화이진트리 원리2: 노드 탐색 및 판별

- 포화 이진 트리를 중위 순회한 리스트의 가운데 값은 해당 트리의 root노드이다.
- 한 부모노드의 값이 0이라면 해당 노드의 자식 노드 뿐만 아니라, 모든 자식 노드에는 1이 존재할 수 없다.

포화이진트리 원리2 활용: 노드 탐색 및 판별

- 재귀함수로 푼다.
- 판별조건1: 부모 노드가 0이라면 모든 자식 노드도 0이어야 한다. 그렇지 않으면 False 리턴
- 판별조건2: 노드 길이가 1이면 규칙에 맡게 끝까지 살아남은 노드이므로 True 리턴
- 재귀조건: 현재 부모 노드 기준으로 좌측과 우측을 별도로 탐색, 각 문자열에 중간에 위치한 원소가 새 부모 노드가 되어 탐색 재귀

```python
def is_tree(s, parent):
    if s[parent] == '0':
        if not all(child == '0' for child in s):
            return False
    if len(s) == 1:
        return True
    left_s = s[:parent]
    right_s = s[parent + 1:]
    return (is_tree(left_s, len(left_s) // 2)
            and is_tree(right_s, len(right_s) // 2))
```

이진 트리 전체 코드

- 전체 코드

1. 먼저 이진 문자열 만들고
2. 문자열의 높이를 구하고(지수)
3. 높이에 따른 유효문자열 길이를 구하고
4. 부족한 부분은 0을 채워주고
5. 재귀로 부모가 0이면 자식들도 전부 0인지 판단
6. 재귀는 문자열의 길이가 1이 될 때까지 반복

```python
from math import log2

def solution(numbers):
    answer = []
    for number in numbers:
        binary = bin(number)[2:]
        indices = int(log2(len(binary))) + 1
        digit = 2 ** indices - 1
        s = '0' * (digit - len(binary)) + binary
        if is_tree(s, len(s) // 2):
            answer.append(1)
        else:
            answer.append(0)
    return answer

def is_tree(s, parent):
    if s[parent] == '0':
        if not all(child == '0' for child in s):
            return False
    if len(s) == 1:
        return True
    left_s = s[:parent]
    right_s = s[parent + 1:]
    return (is_tree(left_s, len(left_s) // 2)
            and is_tree(right_s, len(right_s) // 2))
```