

# Perfect Binary Tree

Problem

Submissions

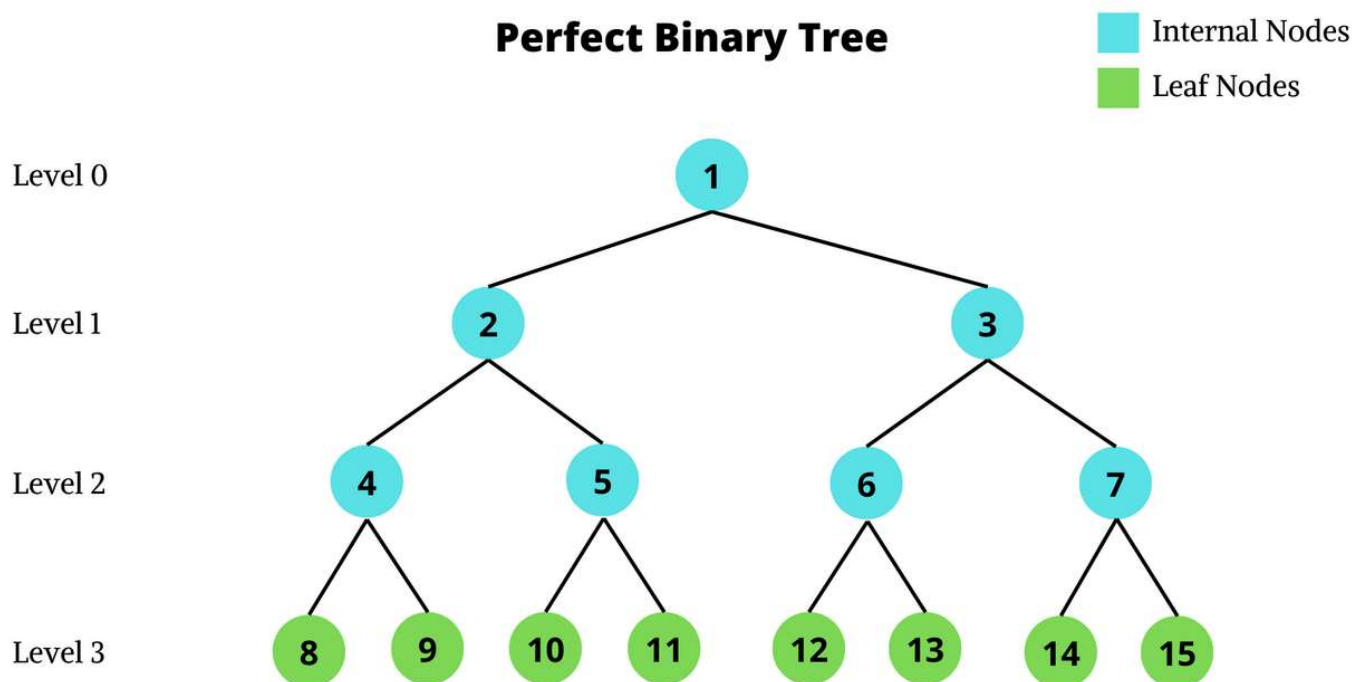
Leaderboard

Discussions

## Problem Statement

You will be given a binary tree as input in level order. You need to tell if the binary tree is perfect or not. A binary tree is called perfect if all leaf nodes are at the maximum depth of the tree, and the tree is completely filled with no gaps.

Here is an example of perfect binary tree:



Also there is formula available to tell if a binary tree is perfect or not. The formula is :

- Total number of nodes =  $2^{\text{maxDepth}} - 1$

**Note:** Here depth is counted from **1**. In the above image maximum depth is **4**, so total number of nodes are  $2^4 - 1 = 15$ . So there should be **15** nodes to call it a perfect binary tree.

## Input Format

- Input will contain the binary tree in level order. **-1** means there is no node available.

## Constraints

1.  $1 \leq \text{Maximum number of nodes} \leq 10^5$

2.  $1 \leq \text{Node's value} \leq 1000$

### Output Format

- Output ***YES*** if the tree is perfect, ***NO*** otherwise.

### Sample Input 0

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

### Sample Output 0

```
YES
```

### Sample Input 1

```
10 20 30 40 -1 60 -1 -1 -1 -1 -1
```

### Sample Output 1

```
NO
```

### Sample Input 2

```
10 20 -1 -1 -1
```

### Sample Output 2

```
NO
```

### Sample Input 3

```
10 20 30 40 50 60 70 -1 -1 -1 -1 -1 -1 -1 -1
```

### Sample Output 3

```
YES
```

[f](#) [t](#) [in](#)

Submissions: [56](#)

Max Score: 20

Difficulty: Easy

Rate This Challenge:

☆☆☆☆☆

[More](#)

C++20



```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
```

```
6
7 int main()
8 {
9     // Write your code here
10
11     return 0;
12 }
13
```

Line: 1 Col: 1

 [Upload Code as File](#) ☐ **Test against custom input**

Run Code

Submit Code

---

[Interview Prep](#) | [Blog](#) | [Scoring](#) | [Environment](#) | [FAQ](#) | [About Us](#) | [Support](#) | [Careers](#) | [Terms Of Service](#) | [Privacy Policy](#) |