Level with maximum nodes

Given a binary tree with `n` nodes, determine the level (considering the root at level 0) that contains the maximum number of nodes. In the case that two or more levels have the same number of nodes, you must output the last such level (i.e. the one with the greatest level index).

You will be given multiple test cases. It is guaranteed that the sum of `n` over all test cases in the input does not exceed 2×10^5.

Input Format

1. The first line contains an integer `T`, the number of test cases.
2. For each test case:
3. The next line contains an integer `n`, the number of nodes in the tree.
4. The following line contains `n` space-separated integers, where the `i^th` integer represents the parent of the `i^th` node. If the integer is -1, the `i^th` node is the root of the tree.

Output Format

For each test case, output a single integer – the level (0-indexed) that contains the maximum number of nodes. If multiple levels have the same maximum count, output the level with the highest index.

Constraints

The sum of `n` over all test cases does not exceed 2×10^5.

The tree is guaranteed to be a valid binary tree.

Helper Code

You may use the following snippet for constructing the binary tree.

Copy

Submit solution

My submissions
All submissions
Best submissions

Points: 100 (partial)
Time limit: 1.0s
Memory limit: 256M

Allowed languages
C

```
    int data;
    struct node *left;
    struct node *right;
};

struct node *create_node(int data) {
    struct node *new_node = (struct node *)malloc(sizeof(struct
node));
    new_node->data = data;
    new_node->left = NULL;
    new_node->right = NULL;
    return new_node;
}

int main() {
    int n; scanf("%d", &n);
    struct node *nodes[n];
    for (int i = 0; i < n; i++) {
        nodes[i] = create_node(i);
    }
    int root = -1;
    for (int i = 0; i < n; i++) {
        int par; scanf("%d", &par);
        if (par == -1) {
            root = i;
        } else {
            if (nodes[par]->left == NULL) {
                nodes[par]->left = nodes[i];
            } else if (nodes[par]->right == NULL) {
                nodes[par]->right = nodes[i];
            }
        }
    }
}
```

Example

Input

Copy

```
5
-1 0 0 1 1
7
-1 0 0 1 1 2 4
1
-1
```

Output

Copy

```
2
2
0
```

Explanation:

For the first tree, level 1 and 2 both have 2 nodes, output the one with greater index. For second tree, level two has maximum number of nodes and for third tree, there is only one level present having one node.

Copy

```
Level 0     0
           / \
Level 1   1   2
         / \
Level 2 3   4
```

Copy

```
Level 0          0
                / \
Level 1        1   2
              / \   \
Level 2      3   4   5
                /
Level 3        6
```

Copy

```
Level 0          0
```

Clarifications

Report an issue

No clarifications have been made at this time.