I've learned about graphs problem in the last few weeks. So, this week I pick this type of the problem. This problem is WalkOverATree. At first, the meaning of this problem is a tree, there's a person on node 0, and now you're asked, how many different nodes can this person visit in L steps, and a node can be walked multiple times, but only once.

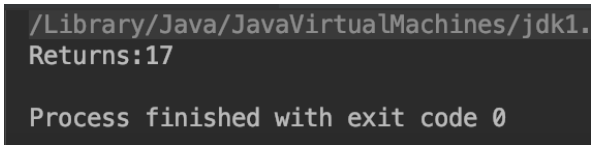
The key to solving this problem is that I can go to each point up to two times, because I can either go to this point once, or I can go back to this point and go to other points, and there's no way I can go three times, so I need to think about that.

So, enumerated a one-time walk path, as the main road, the main road there are a few bypass connections, so I can access this path beside some of the points, and then return to the main road, the point I need to use a total of two times the number to go out, because want to return to the main road, and easy to find, for any one beside the way are the same.
Therefore, some code like this:

```
int N = parent.length + 1;
int[] dist = new int[N];
int max = 0;
for (int i = 1; i < N; i++) {
    dist[i] = dist[parent[i - 1]] + 1;
    max = Math.max(max, dist[i]);
}
if (L <= max) {
    return L + 1;
}
return Math.min(N, (L - max) / 2 + max + 1);
```

Class: WalkOverATree
Test passed.
Since I can't upload it to web submission, the example is shown below and the result looks like this after I run it.

```
{0,0,0,1,1,3,5,1,4,5,2,2,10,5,10,10,11,13,8,3,18,15,20,20,23,8,11,26,4}
26
Returns: 17
```



https://community.topcoder.com/stat?c=problem_statement&pm=13955