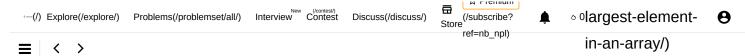
6/22/22, 8:09 AM Explore - LeetCode





- 1. As shown in the code, in each round, the nodes in the queue are the nodes which are waiting to be processed .
- 2. After each outer while loop, we are one step farther from the root node. The variable step indicates the distance from the root node and the current node we are visiting.

## Template II

Sometimes, it is important to make sure that we never visit a node twice. Otherwise, we might get stuck in an infinite loop, *e.g.* in graph with cycle. If so, we can add a hash set to the code above to solve this problem. Here is the pseudocode after modification:

```
Copy
Java
 1
 2
     * Return the length of the shortest path between root and target node.
 3
 4
     int BFS(Node root, Node target) {
       Queue<Node> queue; // store all nodes which are waiting to be processed
 6
       Set<Node> visited; // store all the nodes that we've visited
       int step = 0;
                      // number of steps neeeded from root to current node
 8
       // initialize
 9
       add root to queue;
10
       add root to visited:
11
       // BFS
12
       while (queue is not empty) {
13
         // iterate the nodes which are already in the queue
14
         int size = queue.size();
15
         for (int i = 0; i < size; ++i) {
16
           Node cur = the first node in queue;
           return step if cur is target:
17
18
           for (Node next: the neighbors of cur) {
             if (next is not in visited) {
19
20
                add next to queue;
                add next to visited;
21
22
23
24
           remove the first node from queue;
25
26
         step = step + 1;
27
                      // there is no path from root to target
28
       return -1;
29
```

There are some cases where one does not need keep the visited hash set:

- 1. You are absolutely sure there is no cycle, for example, in tree traversal;
- 2. You do want to add the node to the queue multiple times.