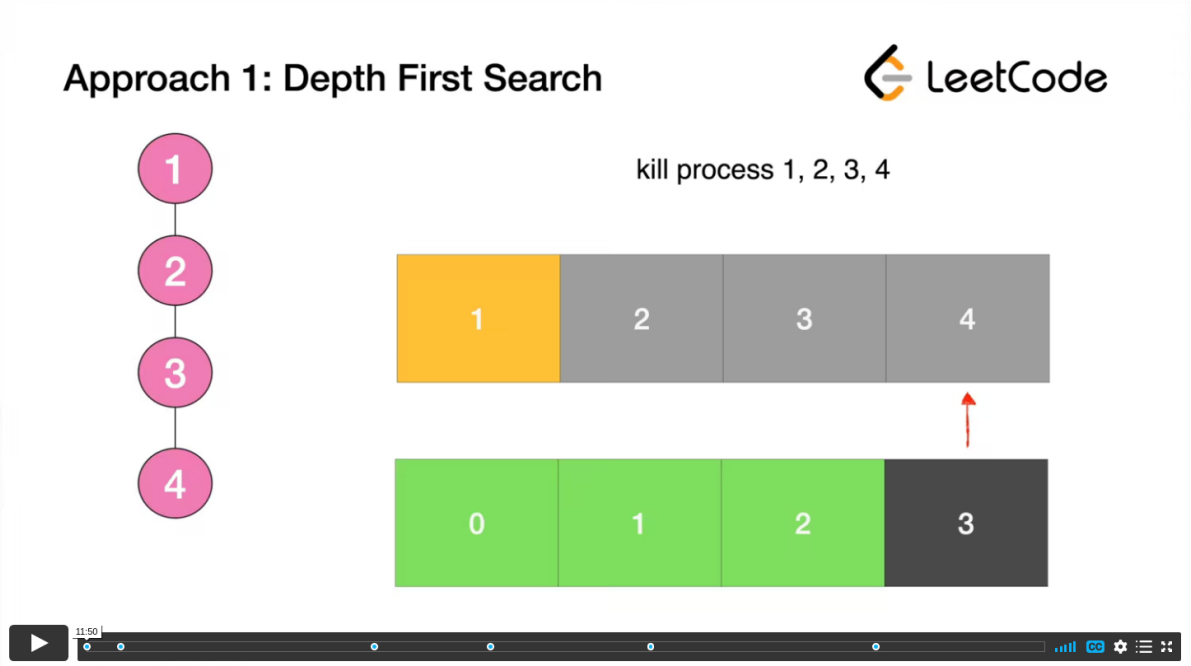
Average Rating: 4.77 (26 votes)

Premium Video

Video Solution



Solution Article

Approach #1 Depth First Search [Time Limit Exceeded]

Algorithm

Since killing a process leads to killing all its children processes, the simplest solution is to traverse over the $ppid$ array and find out all the children of the process to be killed. Further, for every child chosen to be killed we recursively make call to the `killProcess` function now treating this child as the new parent to be killed. In every such call, we again traverse over the $ppid$ array now considering the id of the child process, and continue in the same fashion. Further, at every step, for every process chosen to be killed, it is added to the list l that needs to be returned at the end.

Java

Copy

```
public class Solution {

    public List < Integer > killProcess(List < Integer > pid, List < Integer > ppid, int kill) {
        List < Integer > l = new ArrayList < > ();
        if (kill == 0)
            return l;
        l.add(kill);
        for (int i = 0; i < ppid.size(); i++)
            if (ppid.get(i) == kill)
                l.addAll(killProcess(pid, ppid, pid.get(i)));
        return l;
    }
}
```

Complexity Analysis

- Time complexity : $O(n^n)$. $O(n^n)$ function calls will be made in the worst case
- Space complexity : $O(n)$. The depth of the recursion tree can go upto n.

Approach #2 Tree Simulation [Accepted]

Algorithm

We can view the given process relationships in the form of a tree. We can construct the tree in such a way that every node stores information about its own value as well as the list of all its direct children nodes. Thus, now, once the tree has been generated, we can simply start off by killing the required node, and recursively killing the children of each node encountered rather than traversing over the whole $ppid$ array for every node as done in the previous approach.

In order to implement this, we've made use of a $Node$ class which represents a node of a tree. Each node represents a process. Thus, every node stores its own value($Node.val$) and the list of all its direct children($Node.children$). We traverse over the whole pid array and create nodes for all of them. Then, we traverse over the $ppid$ array, and make the parent nodes out of them, and at the same time add all their direct children nodes in their $Node.children$ list. In this way, we convert the given process structure into a tree structure.

Now, that we've obtained the tree structure, we can add the node to be killed to the return list l. Now, we can directly obtain all the direct children of this node from the tree, and add its direct children to the return list. For every node added to the return list, we repeat the same process of obtaining the children recursively.

Java

Copy

```
public class Solution {
    class Node {
        int val;
        List < Node > children = new ArrayList < > ();
    }
    public List < Integer > killProcess(List < Integer > pid, List < Integer > ppid, int kill) {
```

```
        HashMap < Integer, Node > map = new HashMap < > ();
        for (int id: pid) {
            Node node = new Node();
            node.val = id;
            map.put(id, node);
        }
        for (int i = 0; i < ppid.size(); i++) {
            if (ppid.get(i) > 0) {
                Node par = map.get(ppid.get(i));
                par.children.add(map.get(pid.get(i)));
            }
        }
        List < Integer > l = new ArrayList < > ();
        l.add(kill);
        getAllChildren(map.get(kill), l);
        return l;
    }
    public void getAllChildren(Node pn, List < Integer > l) {
        for (Node n: pn.children) {
            l.add(n.val);
            getAllChildren(n, l);
        }
    }
```

Complexity Analysis

- Time complexity : $O(n)$. We need to traverse over the $ppid$ and pid array of size n once. The `getAllChildren` function also takes atmost n time, since no node can be a child of two nodes.
- Space complexity : $O(n)$. map of size n is used.

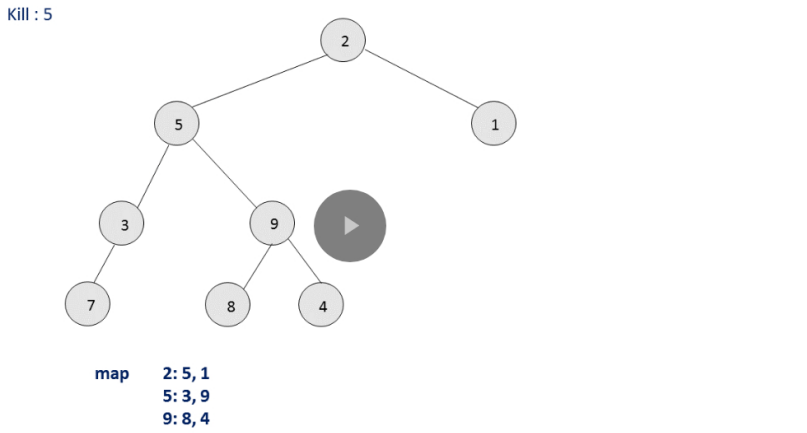
Approach #3 HashMap + Depth First Search [Accepted]

Algorithm

Instead of making the tree structure, we can directly make use of a data structure which stores a particular process value and the list of its direct children. For this, in the current implementation, we make use of a hashmap map, which stores the data in the form $parent : [listofallitsdirectchildren]$.

Thus, now, by traversing just once over the $ppid$ array, and adding the corresponding pid values to the children list at the same time, we can obtain a better structure storing the parent-children relationship.

Again, similar to the previous approach, now we can add the process to be killed to the return list, and keep on adding its children to the return list in a recursive manner by obtaining the child information from the structure created previously.



Java

Copy

```
public class Solution {
    public List < Integer > killProcess(List < Integer > pid, List < Integer > ppid, int kill) {
        HashMap < Integer, List < Integer >> map = new HashMap < > ();
        for (int i = 0; i < ppid.size(); i++) {
            if (ppid.get(i) > 0) {
                List < Integer > l = map.getOrDefault(ppid.get(i), new ArrayList < Integer > ());
                l.add(pid.get(i));
                map.put(ppid.get(i), l);
            }
        }
        List < Integer > l = new ArrayList < > ();
        l.add(kill);
        getAllChildren(map, l, kill);
        return l;
    }
    public void getAllChildren(HashMap < Integer, List < Integer >> map, List < Integer > l, int kill) {
        if (map.containsKey(kill))
            for (int id: map.get(kill)) {
                l.add(id);
                getAllChildren(map, l, id);
            }
    }
}
```

Complexity Analysis

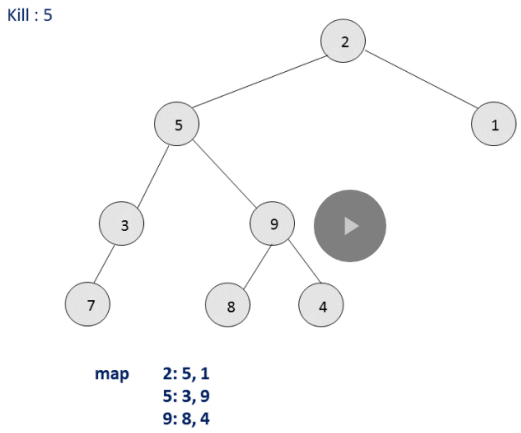
- Time complexity : $O(n)$. We need to traverse over the $ppid$ array of size n once. The `getAllChildren` function also takes atmost n time, since no node can be a child of two nodes.
- Space complexity : $O(n)$. map of size n is used.

Approach #4 HashMap + Breadth First Search [Accepted]:

Algorithm

We can also make use of Breadth First Search to obtain all the children(direct+indirect) of a particular node, once the data structure of the form $(process : [listofallitsdirectchildren]$ has been obtained. The process of obtaining the data structure is the same as in the previous approach.

In order to obtain all the child processes to be killed for a particular parent chosen to be killed, we can make use of Breadth First Search. For this, we add the node to be killed to a $queue$. Then, we remove an element from the front of the $queue$ and add it to the return list. Further, for every element removed from the front of the queue, we add all its direct children(obtained from the data structure created) to the end of the queue. We keep on doing so till the queue becomes empty.



1 / 9

Java

Copy

```java
public class Solution {

    public List < Integer > killProcess(List < Integer > pid, List < Integer > ppid, int kill) {
        HashMap < Integer, List < Integer >> map = new HashMap < > ();
        for (int i = 0; i < ppid.size(); i++) {
            if (ppid.get(i) > 0) {
                List < Integer > l = map.getOrDefault(ppid.get(i), new ArrayList < Integer > ());
                l.add(pid.get(i));
                map.put(ppid.get(i), l);
            }
        }
        Queue < Integer > queue = new LinkedList < > ();
        List < Integer > l = new ArrayList < > ();
        queue.add(kill);
        while (!queue.isEmpty()) {
            int r = queue.remove();
            l.add(r);
            if (map.containsKey(r))
                for (int id: map.get(r))
                    queue.add(id);
        }
        return l;
    }
}
```

Complexity Analysis

- Time complexity : $O(n)$. We need to traverse over the $ppid$ array of size n once. Also, atmost n additions/removals are done from the $queue$.
- Space complexity : $O(n)$. map of size n is used.

Report Article Issue

Comments: 28

Best | Most Votes | Newest to Oldest | Oldest to Newest

Type comment here... (Markdown is supported)

Post

laotzu ★ 17 Last Edit: September 6, 2018 10:47 PM

I think the time complexity for first approach is n^2, because there is no repetition nodes which are already explored during the search process.

14 Show 4 replies Reply

cleverKnight ★ 66 Last Edit: September 5, 2018 12:05 PM

I feel like you should also mention that a simple topological sort from the 'kill' node generates the answer - particularly Approach #3 does basically just that.

6 Show 1 reply Reply

nicksg3395 ★ 168 September 23, 2019 3:13 AM

Should be an easy problem

44 Reply

slz250 ★ 422 November 4, 2018 2:58 PM

can someone explain why the brute force approach (#1) runtime is O(n^n)?

3 Show 3 replies Reply

mehakwaliaC ★ 252 February 14, 2021 5:00 AM

The time complexity for the first approach should be O(n^2) or otherwise, I just don't know what I am doing here!

2 Reply

Mshubhi ★ 2 July 17, 2018 1:03 PM

```java
List < Integer > l = new ArrayList < > ();
        if (kill == 0)
            return l;
        l.add(kill);
        for (int i = 0; i < ppid.size(); i++)
            if (ppid.get(i) == kill)
                l.addAll(killProcess(pid, ppid, pid.get(i)));
        return l;
```

This code is wrong as if we are killing 0 then it will return an empty list ,but since 0 is the root process ideally all process should be in the result.

Read More

1 Show 1 reply Reply

codingWithJaz ★ 10 August 24, 2020 11:43 AM

Easy to understand Python3 Solution using Hashmap and Queue:

```
def killProcess(self, pid: List[int], ppid: List[int], kill: int) -> List[int]:
        children_map = defaultdict(list)

        for i in range(len(pid)):
            if ppid[i] != 0:
                children_map[ppid[i]].append(pid[i])


        queue = [kill]
```

Read More

1 Reply

anupHack ★ 4 June 19, 2017 6:06 PM

I was also thinking if it can be done like, find the node that needs to be killed. Then mark that with respect to its parent as null, (e.g. in this 5 is the node to be killed so mark 2's left to be null) but save that tree in a temporary node, say temp. After that just traverse using one of the traversal and add the nodes to be printed in an array or list and print them. How about this?

0 Reply

vinod23 ★ 670 June 1, 2017 3:08 AM

@mrvon @aprilyin Sorry there was some problem. We have fixed it. Thanks

0 Reply

mrvon ★ 11 June 1, 2017 1:29 AM

It seems mismatching answer and problem.

0 Reply