71. Simplify Path

14 March 2022 10:16 AM

Given a string `path`, which is an **absolute path** (starting with a slash `'/'`) to a file or directory in a Unix-style file system, convert it to the simplified **canonical path**.

In a Unix-style file system, a period `'.'` refers to the current directory, a double period `'..'` refers to the directory up a level, and any multiple consecutive slashes (i.e. `'//'`) are treated as a single slash `'/'`. For this problem, any other format of periods such as `'...'` are treated as file/directory names.

The **canonical path** should have the following format:

- The path starts with a single slash `'/'`.
- Any two directories are separated by a single slash `'/'`.
- The path does not end with a trailing `'/'`.
- The path only contains the directories on the path from the root directory to the target file or directory (i.e., no period `'.'` or double period `'..'`)

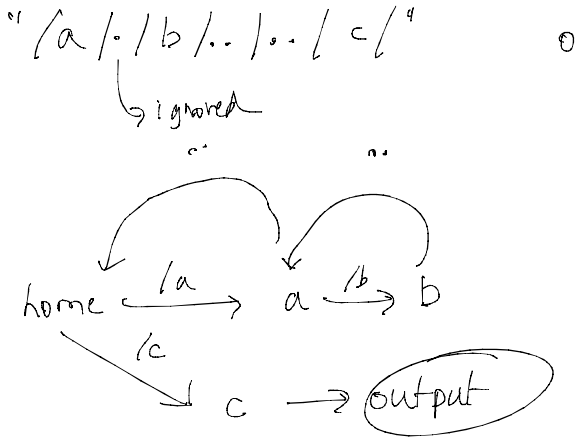
Return *the simplified* ***canonical path***.

Example 1:

```
Input: path = "/home/"
Output: "/home"
Explanation: Note that there is no trailing slash after the
last directory name.
```

Example 2:

```
Input: path = "/../"
Output: "/"
Explanation: Going one level up from the root directory is a
no-op, as the root level is the highest level you can go.
```



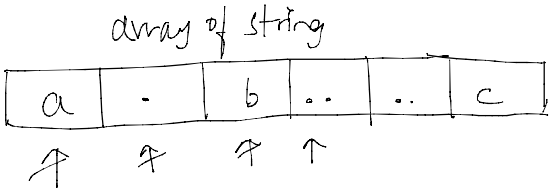
* Operation we are doing here are pushing and popping directory names based on rules.

→ /.. ⇒ pop (comes out of directory)

→ /name ⇒ push (go into the director) (/a)

→ we can use stacks or double ended queue (deque)

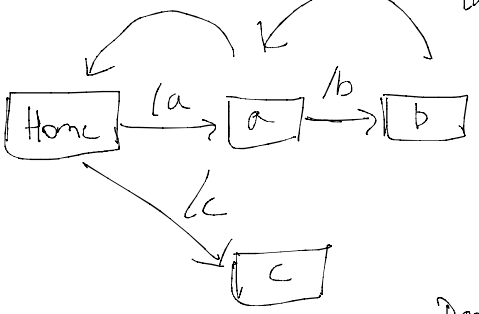
/a/./b/../../c/ ⇒



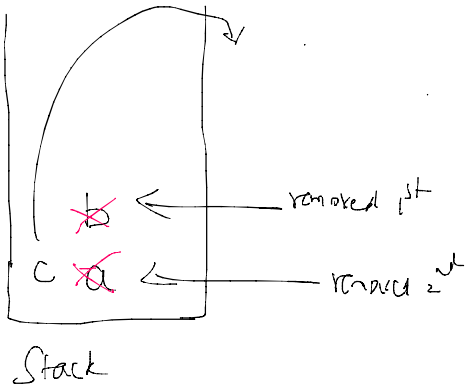
here we start from front and start adding slash with the directory name present in the queue

After completion

/.. /.. → we remove the last element



path = /c



After completion In the stack, you need to go in the reverse order because when you pop from the stack the highest one get pop, but we need lowest one first so need to append in a careful manner.

The difference between the both approach is just while popping out the elements and creating the last result string.

```java
// Deque Solution

class Solution {
    public String simplifyPath(String path) {
        Deque<String> s = new LinkedList<>();
        StringBuilder res = new StringBuilder();
        String[] p = path.split("/");

        for(int i=0; i<p.length; i++){
            // two condition now pop and pushing
            //polling
            //whenever two dots present we pop it and queue should't be empty
            if(!s.isEmpty() && p[i].equals(".."))
                s.poll();

            //pushing
            //whenever empty string and when there is not single dot or double dot
            else if(!p[i].equals("") && !p[i].equals(".") && !p[i].equals(".."))
                s.push(p[i]);
        }
        // base case
        if(s.isEmpty()) return "/";
        while(!s.isEmpty()){
            // only change in while for Dequeue
            res.append("/").append(s.pollLast());
        }
        return res.toString();
    }
}
```