636. Exclusive Time of Functions

Description HintsSubmissionsDiscussSolution

- DiscussPick One Difficulty:**Medium**
- Total Accepted:1.7K
- Total Submissions:4.7K
- Contributor:fallcreek

Given the running logs of **n** functions that are executed in a nonpreemptive single threaded CPU, find the exclusive time of these functions.

Each function has a unique id, start from **0** to **n-1**. A function may be called recursively or by another function.

A log is a string has this format : `function_id:start_or_end:timestamp`. For example, `"0:start:0"` means function 0 starts from the very beginning of time 0. `"0:end:0"` means function 0 ends to the very end of time 0.

Exclusive time of a function is defined as the time spent within this function, the time spent by calling other functions should not be considered as this function's exclusive time. You should return the exclusive time of each function sorted by their function id.

Example 1:

Input:

```
n = 2
```

logs =

["0:start:0",

"1:start:2",

"1:end:5",

"0:end:6"]**Output:**[3, 4]**Explanation:**

Function 0 starts at time 0, then it executes 2 units of time and reaches the end of time 1.

Now function 0 **calls function 1**, function 1 starts at time 2, executes 4 units of time and end at time 5.

Function 0 is running again at time 6, and also end at the time 6, thus executes 1 unit of time.

So function 0 totally execute 2 + 1 = 3 units of time, and function 1 totally execute 4 units of time.

Note:

1. Input logs will be sorted by timestamp, NOT log id.
2. Your output should be sorted by function id, which means the 0th element of your output corresponds to the exclusive time of function 0.
3. Two functions won't start or end at the same time.
4. Functions could be called recursively, and will always end.
5. 1 <= n <= 100

```
void logAnalyse(string log, int &id, int &start, int &end)
{
    if(log.find("start")!=string::npos)
    {
        int idx = log.find("start");
        string tmpid = log.substr(0,idx-1);
        string tmpstart = log.substr(idx+6);
        id = (int)strtod(tmpid.c_str(),NULL);
        start = (int)strtod(tmpstart.c_str(),NULL);
        end = -1;
    }else if (log.find("end")!=string::npos)
    {
        int idx = log.find("end");
        string tmpid = log.substr(0,idx-1);
        string tmpend = log.substr(idx+4);
        id = (int)strtod(tmpid.c_str(),NULL);
        end = (int)strtod(tmpend.c_str(),NULL);
        start = -1;
    }
}


vector<int> exclusiveTime(int n, vector<string>& logs) {
   stack<int> timest;
   vector<int> res(n,0);
   int id,start,end;
   int prev;
   for(int i=0;i<logs.size();++i)
   {
       string log = logs[i];
       if(i==0)
       {
           logAnalyse(log,id,start,end);
           prev = start;
           timest.push(id);
           continue;
       }
       logAnalyse(log,id,start,end);
       if(start!=-1)
       {
           if(!timest.empty())
           {
               res[timest.top()] += start - prev;
```

```
            }
            timest.push(id);
            prev = start;
        }else if (end!=-1)
        {
            res[timest.top()] += end - prev + 1;
            timest.pop();
            prev  = end+1;
        }

    }
    return res;
}
```