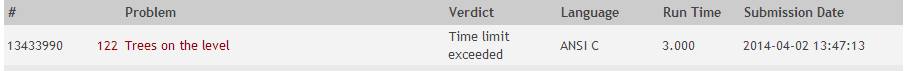
Trees on the Level

UVA Online Judge

DASALGO Problem Code: T1 Status: Time Limit Exceeded

UVA Problem 122 – Trees on the Level Runtime: 3 seconds



(Source code with comments at the last page)

Discussion

Table 1: Test Cases

|  |  |  |
| --- | --- | --- |
| (3,)() | Tree only contains root node | 3 |
| (5,RRL) (6,R) (7,L) () | Tree is not completely specified | Not complete |
| (7,L) (5,) (4,LL) (5,RR) (10,RRR) (3,R) (8,RL) (9,LR) () | Tree is completely specified but is not full | 5 7 3 4 9 8 5 10 |
| (8,L) (5,R) (9, LL) (3,) (4,LR) (1,RL)(2,RR)() | Tree is full and completely specified | 3 8 5 9 4 1 2 |
| (1,)(2,L)(3,LL)(4,LLL)(5,LLLL)() | Tree is one sided and looks linear | 1 2 3 4 5 |

A binary tree can easily be represented as an array. A full binary tree with at most 255 nodes has at most 8 levels.

The code starts by going through the characters in the input and reading them accordingly. The character ‘(‘ specifies the start of the input for a single node. When the program reads this character, it checks if this character is followed by an integer. If not, then, assuming all input is correct, then the next character is a ‘)’, which indicates the end of input for a single tree. Otherwise, it stores this input in a temporary integer variable. It then continues to read characters until it detects the ‘,’ character, at which point the program will start calculating for the index of the node. The next input, again, assuming that all input is correct, is a string of ‘R’’s and ‘L’’s that describe the placement of the node in the tree. The index calculation starts with a value of 0. For every R that the program reads, it increases the index by (2v+2) where v is the index. Every L that it reads increases it by (2v+1). When the program reads the ‘)’ character after this string, it stores the value of the node into its calculated index and marks any possible children nodes (at indexes 2i+2 and 2i+1, where i is the index of the node) as ‘has parent’ (where 0 means no parent, and 1 means has parent). It then increments the number of times the node at this placement has been given a value.

To check whether the tree is complete or not, the program simply iterates over the nodes and checks the number of times it has been given a value and the Boolean stating whether or not it has a parent. If the node being checked has been given a value and has no parent, then the tree is automatically tagged as incomplete and the words “not complete” are printed onto the output.

If the tree is complete, the program iterates over the nodes again, starting at index 0. Any node that has been given a value is then printed onto the output.

UVA did not accept this code, stating that it exceeded the given time limit.

References

"UVa Online Judge." UVa Online Judge. N.p., n.d. Web. 2 Apr. 2014. <http://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&page= show\_problem&problem=58>