Lumberjack*

Cool Cutters†
Tarun Goyal (180010038), Manjeet Kapil (180010021), Shagun Bera (180010031)
Computer Science and Engineering, IIT Dharwad

November 6, 2019

Abstract

This paper describes the algorithm and heuristics followed by the program written by Cool Cutters for the *Lumberjack* problem listed in the online platform Optil.io.

1 Our Algorithm

We store the total number of trees , maximum time allowed and the grid size in their respective variables k , t , n . Similarily their properties x coordinate , y coordinate , height , thickness , weight of the tree , profit obtained by cutting that tree are stored in their respective vectors in order.
We have some other temporary variables which store our current position others like xInitial , yInitial .
Each tree can be identified by its index (the order at which it was stored), all other vectors use this index as reference.
Now we execute findyprofit() which essentialy tells us the profit each tree could yield if it is cut in any direction left , right , up or down. We store this information in 4 vectors proLeft , proRight ,proUp ,proDown.
These Pro Vectors are 2D vectors , the first element stores net profit (considering domino effect) and then the index of the trees that fall due to the domino effect . A sample Pro Vector looks like :-

100 2 3 4
600 5 19 20
350 8 5 6

*This is a report on the course project for the course CS211 Data Structures and Algorithms Lab

†Email IDs of team members: 180010038@iitdh.ac.in 180010021@iitdh.ac.in 180010031@iitdh.ac.in

We have 4 functions left() , right() , up() , down() which populate these vectors.
We have 3 more vectors diru , uttuprofit and flag which store the direction in which that tree is to be cut i.e the direction which gives maximum profit , the trees that fall due to the domino effect and if the tree is already cut or not respectively.
Now we have all the necessary information for executing run()

We now have a new vector timy which stores the time required to reach and cut that tree (if a tree is cut already we store infinity as the time required to reach it).
We find the ratio of profit obtained and time for reaching and cutting tree , and proceed to cut the tree which has that maximum ratio . After cutting the tree the domino effect is considered using information from uttuprofit .The corresponding steps "move up" and "move down" etc are also printed here.The direction is also printed using information from the diru vector. The flags are also updated and run() is called again recursively till the maximum time limit is reached or we cant find any feasible tree .

2 Previous algorithms

2.1 4th Evaluation

We use the ratio of net profit and time taken if number of trees are less than 4000 and use maximum profit if there are more trees.

2.2 2nd Evaluation

We used maximum profit to determine the next tree to be cut (within given time).