LUMBERJACK*

Blaze[†]

Abhinav Pratap Singh (180010001), Rishit Saiya (180010027), Utkarsh Prakash (180030042)

Computer Science and Engineering, IIT Dharwad

November 6, 2019

Abstract

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

1 Environment Variables

- For storage of different parameters of trees, a structure has been used.
- Profit = Price + Profit earned by Domino Effect i.e. Price of other tree that fall due to domino effect.
- direc variable stores the direction in which the tree should be cut so as to get the maximum profit.
- Vector v stores information of all the trees.
- c_x and c_y store the information about the current x,y coordinates i.e. the coordinates of the most recently cut tree.
- n_x and n_y store the coordinates of next tree to be cut.
- t is the time that has been given to us cut the tree (not the execution time of the program).
- We have given **color** to every tree in order to know whether the tree has been cut or not.

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

[†]Email IDs of team members : [180010001]`@iitdh.ac.in`, [180010027]`@iitdh.ac.in`, [180030042]`@iitdh.ac.in`

2 Main Algorithm Idea

The main idea remains to maximize the profit to maximum extent.

The main factors affecting the profit are :

- The extraction of maximum profit from the Domino Effect.
- The time given to us is limited, hence we can't cut all the tree. Moreover, since the exceution time of the program is also limited hence we can't check all the possible path and find the route that yields maximum profit.

So, we decided that we would choose the tree having the maximum value of:

$$\frac{profit}{time} \tag{1}$$

2.1 Extraction of maximum profit from Domino Effect

The functions used for the above mentioned factors are calculate_profit(), cutup_profit(), cutdown_profit(), cutright_profit(), cutdown_profit().

- **cutup_profit() function**

  This function first finds the nearest neighbour of the given tree (in the "up" direction) and then checks whether that tree can have a domino effect or not.

  Then it makes a recursive call to find that whether the next nearest neighbour of the given tree (in the up direction) can participate in the domino effect.

  The function returns the extra-profit that can be made when the tree is cut in the up-direction due to the domino effect.

  The function also calculates which trees will be cut due to domino effect.

- **calculate_profit() function**

  This function calculates the direction in which the profit is maximum and updates the profit of each tree in that direction.

NOTE : Similarly functions cutright_profit(), cutdown_profit() and cutleft_profit calculate the profit of cutting the trees in right, down and left respectively.

2.2 Choosing the tree next tree to be cut

The function used for the above mentioned factor is path().

- **path() function**

  This function first computes the cost of cutting a tree i.e. the cost of reaching to that tree plus the diameter of the tree.

  If this cost is less than the time left with us, then it calculates the profit/time for that tree. Then it finds the maximum of this quantity among all the trees and the next tree to be cut will be the tree which has the maximum of this quantity.

2.3 Printing expected output

The function used for the above mentioned factor is print_path().

- **print_path() function**

  This function simply prints the path that is to be followed to reach a tree.

2.4 Updating our database

- When we want cut a tree we color it black. We update the c_x and c_y as n_x and n_y respectively.

3 Final Notes

- The other variables used in the program like h, p, c, etc. have the same meaning as mentioned in the problem statement.
- With this algorithm and heuristics, we were finally able to make a total profit of **14,44,21,499 units**.