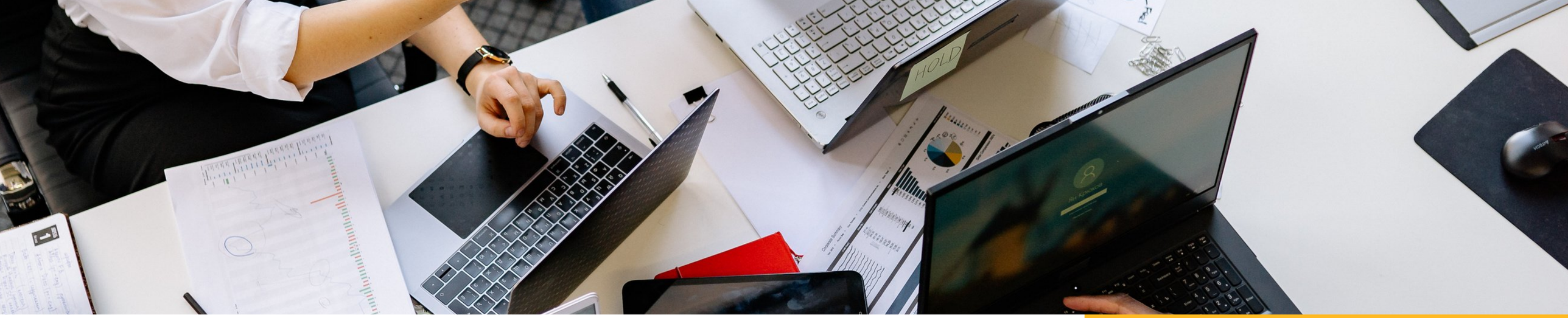


CS201 DSA Project

CODERS

PRESENTS

LUMBERJACK PROBLEM

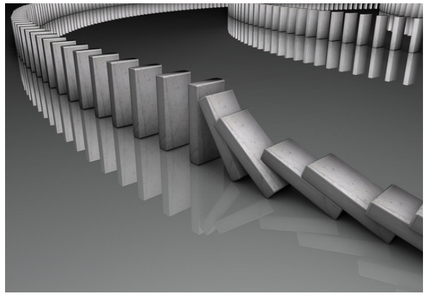
ALGORITHM LOGIC

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

- Taking the path involving the trees in shorter range to previous range.
- The extraction of maximum profit from the Domino Effect.

Hence, we decided the tree which needs to cut on basis of the Quantity profit/time

The Problem



Extraction of Maximum profit from Domino Effect

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

We will explain the functions explicitly in the coming slides.

DATA STRUCTURE USED

- We are using a vector data structure which is a dynamic array because it will change its size when it is completely full and then the user can store the data again. In simple array we have to increase the size again and again but it is not with vectors, and we are using iterator to pull the data into vector and for accessing the data from vectors also

- Time complexity for Algorithm
- $D(n^2)$

.

Major theoretical observation made to solve this problem is

1. Initially we need to find the shortest path to reach to the tree and minimum time to cut that tree.
2. We have to use Domino effect proportionally Because need to cut minimum tree and we also want that tree v well we will cut will fall on next tree and that tree e will fall on itself without cutting in order to save time

For maximum profit we had only to find the technique.

Pseudocode

- Step I: START.
- Step 2: INCLUDE some inbuilt library
  #include<iostream>
  #include<vector>
  #include<cstdlib.h>
  #include<bits/stdc++.h>
- Step 3: DECLARE namespace command
- Step 4: DECLARE a user define data
  type struct lumber

- Step 5: INSIDE struct lumber initialize some variable
  int x , y , h , d , c , p
  int price , weight ,direc(direction)
  vector track
- Step 6: DECLARE more variable for tracing lumber path
  int a [ ]
  int c_x=0, c_y=0,px=0,py=0
  int n_x, n_y int t, n, k
- Step 7: DECLARE The functions prototype
  int path()
  void calculate_profit()
  void printPath()
  int pathI()

int cutright_profit()
int cutdown_profit()
int cutup_profit()
int cutleft_profit()

- Step 8: Now main() function starts
- Step 9: INSIDE main() function

  Take t , n , k , i.e.

  Time as t, grid size as n, Number of tress as k
- Step 10: while(i<k)

  Taking input for each single tree till all trees are cutdown

  For each Tree

  Input v[i].x
  v[i].y

  Position v[i].h
  v[i].d

  Height & thickness v[i].c
  v[i].p

  Weight & volume

Then CALCULATE

v[i].weight = v[i].c * v[i].d *v[i].h

v[i].price = v[i].p * v[i].h * v[i].d

TRACKING

v[i].profit = v[i].price

- Step II: DECLARE

  Variable inside main function

  int total_p = 0 , m , pos , Total

- Step I2 : WHILE(i<k)

  IF ((abs(v[i].x)+abs(v[i].y<m))

  THEN TAKE m=abs(v[i].x)+abs(v[i].y)

  IF (v[pos].x+v[pos].y+v[pos].d<=t)

  THEN total_p=total_p+v[pos].price

  IF (total_p!=11475)

  THEN calculate the profit using profit function calculate_profit()

- Step 13 : WHILE (t>0)
  int temp2=0
  IF (v.size() > 0)
  DECLARE variable i = path()
  DECLARE another vector temp of int type
vector<lumber>:: iterator it
  DECLARE it
  it = v.begin()
  PUT n_x= v[i].x
  AND n_y= v[i].y
  CALL printpath function

WHEN (t>0)
THEN using cutup_profit, cutright_profit, cutdown_profit, cutleft_profit functions

v[i].direc = 0 THEN print the message "cut up"

v[i].direc = 1 THEN print the message "cut right"

v[i].direc = 2 THEN print the message "cut down"

v[i].direc = 3 THEN print the message "cut left"

- THEN calculate the time required to cut the tree

t= t- v[i].d

USING sort function sort(temp.begin(), temp.end())

WHILE (j<temp.size())

CALLING v.erase() function

DECLARING it = v.begin()

WHILE (j<v.size)

IF(n_x ==v[j].x && n_y == v[j].y)

CALLING v.erase() function again

DECLARING

c_x = n_x;

c_y = n_y;

ELSE

come out of loop


Thank
you!