LumberJack

Team-FAB

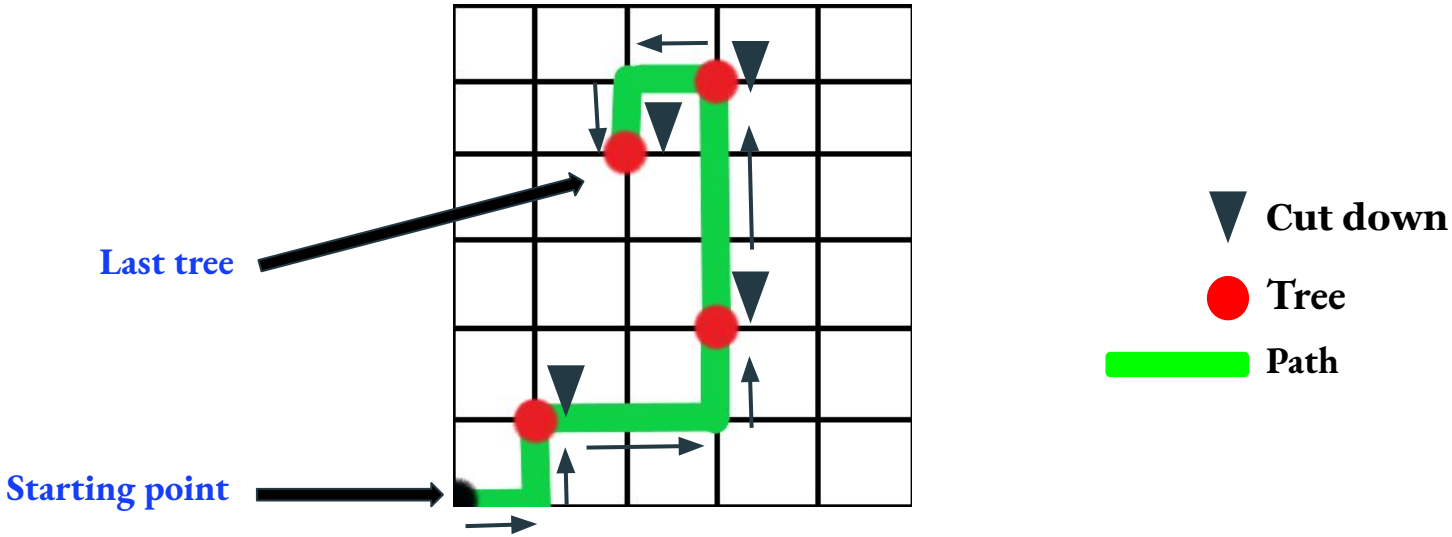
Algorithm-1

Algorithm submitted for first evaluation.

Our Algorithm:

- ➔ Takes the input and stores the data in an array.
- ➔ x and y coordinates are stored in first and second slots in every six slots in an array i.e., **a[6*i]** and **a[6*i + 1]**.
- ➔ Then creates a 2D array g[][].
- ➔ Cost of each tree(**(a[6*i + 2])*(a[6*i + 3])*(a[6*i + 5])**) is stored in the 2D array at their respective coordinates(**g[a[6*i]][6*i + 1]**).
- ➔ Initialises the starting point coordinates as (x,y) = (0,0).
- ➔ Finds the closest point from the starting point.
- ➔ Goes to the closest point and cuts the tree downwards.
- ➔ Removes data of the tree from the array.

- ➔ Then equates x and y coordinates to the closest point.
- ➔ Finds the closest point from (x, y), goes to closest point and cuts the tree downwards.
- ➔ Then equates x and y coordinates to the closest point and removes tree's data from array.
- ➔ This continues till all the trees are cut.



Our algorithm was not accepted by optil.io platform because,

➔ Our algorithm doesn’t include the domino effect.

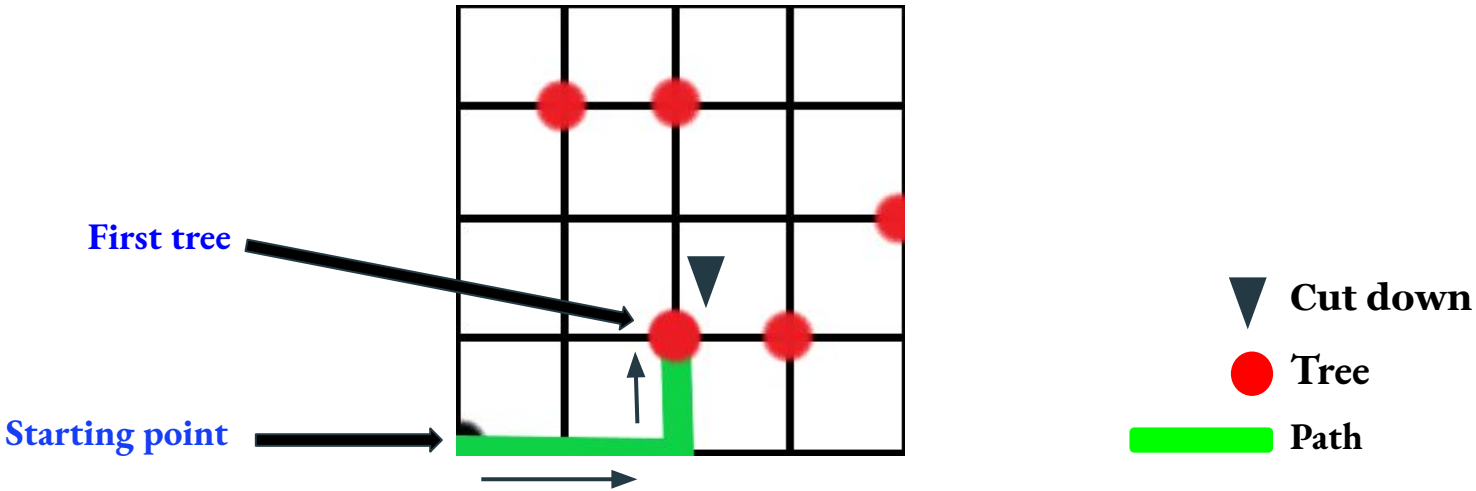
So, it was showing WA(cutting tree in a position where there is no tree).

Algorithm-2

Algorithm submitted for second and third evaluations.

Our Algorithm:

- ➔ Takes the input and stores the data in an array.
- ➔ x and y coordinates are stored in first and second slots in every six slots in an array i.e., **a[6*i]** and **a[6*i + 1]**.
- ➔ Then creates a 2D array g[][].
- ➔ Cost of each tree(**(a[6*i + 2])*(a[6*i + 3])*(a[6*i + 5])**) is stored in the 2D array at their respective coordinates(**g[a[6*i]][6*i + 1]**).
- ➔ Initialises the starting point coordinates as (x,y) = (0,0).
- ➔ Finds the closest point from the starting point.
- ➔ Goes to the closest point and cuts the tree downwards.
- ➔ And terminates.

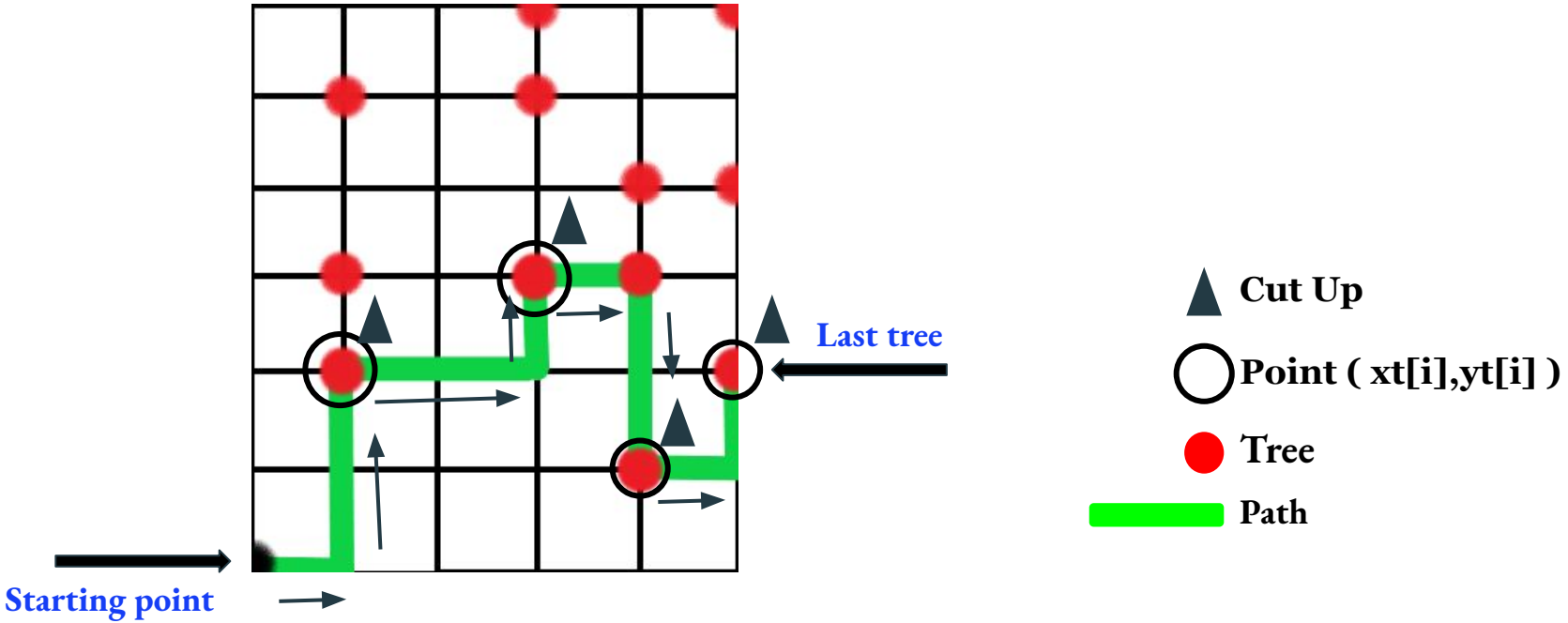

First tree
Starting point
Cut down
Tree
Path

Algorithm-3

Algorithm submitted for fourth evaluation.

Our Algorithm:

- ➔ Takes the input and stores the data in an array.
- ➔ x and y coordinates are stored in first and second slots in every six slots in an array i.e., **a[6*i]** and **a[6*i + 1]**.
- ➔ Then creates a 2D array g[][].
- ➔ Cost of each tree(**(a[6*i + 2])*(a[6*i + 3])*(a[6*i + 5])**) is stored in the 2D array at their respective coordinates(**g[a[6*i]][6*i + 1]**).
- ➔ Then all the unique x-coordinates are stored in an array xt[](in ascending order).
- ➔ And a corresponding y coordinate is stored in array yt[] whose value is equal to the minimum of all y values whose x values are equal.
- ➔ Initialises the starting point coordinates as (x,y) = (0,0).
- ➔ Goes to point (xt[i],yt[i]) and cut the tree upwards.
- ➔ This goes on till the end of the array xt[].


Last tree
Starting point
Cut Up
Point (xt[i],yt[i])
Tree
Path

Thank You