Algorithms Lab

Assignment 2: 2-D Maxima

Aditya Badayalya

510819056

IT-Hy

Brute Force Algorithm:

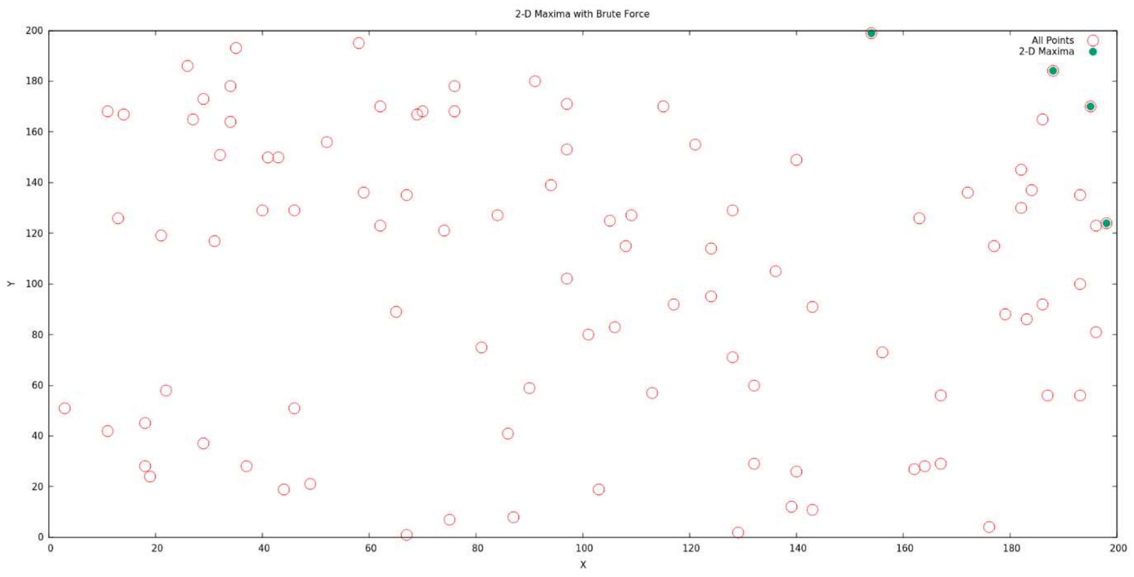
```
vector<pair<int, int>> bruteForce(vector<pair<int, int> >&v){

    vector<pair<int, int> >ans;
    bool maxima = true;
    for(int i=0;i<v.size();i++){
        maxima=true;
        for(int j=0;j<v.size();j++){
            if((i!=j)&&(v[i].first <= v[j].first)&&
              (v[i].second<= v[j].second)){
                maxima = false;
                break;
            }
        }
        if(maxima){
            ans.push_back(v[i]);
        }
    }
    return ans;

}
```

Here in the pair datatype, the first element is the x-coordinate of the point whereas the second one is the y-coordinate

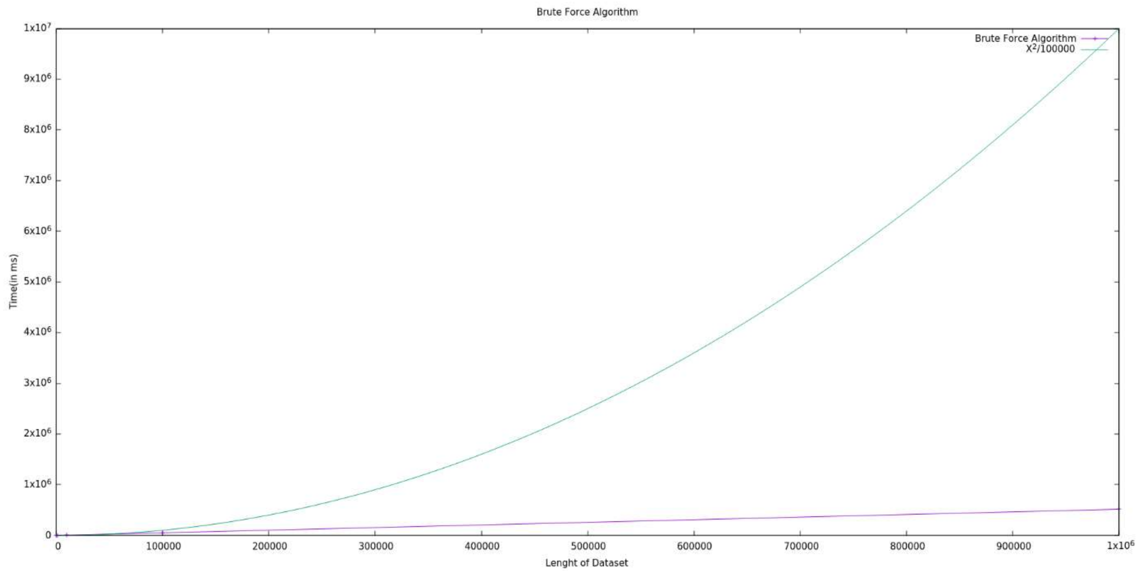
In this method the given point is compared with all the other points in the sample space. If it is found that the give point is dominated in both x and y prospects, it is discarded from the maxima consideration.



The above plot denotes a set of distinct points and the 2-D maxima of that set.

There are two nested loops in the algorithm. As each point is compared with every other point but itself, the inner loop can execute at most n times where is n is the length of the dataset and the outer loop always executes n times.

Therefore, the worst-case time complexity of this algorithm can be considered as $O(n^2)$ where n is the length of data set.

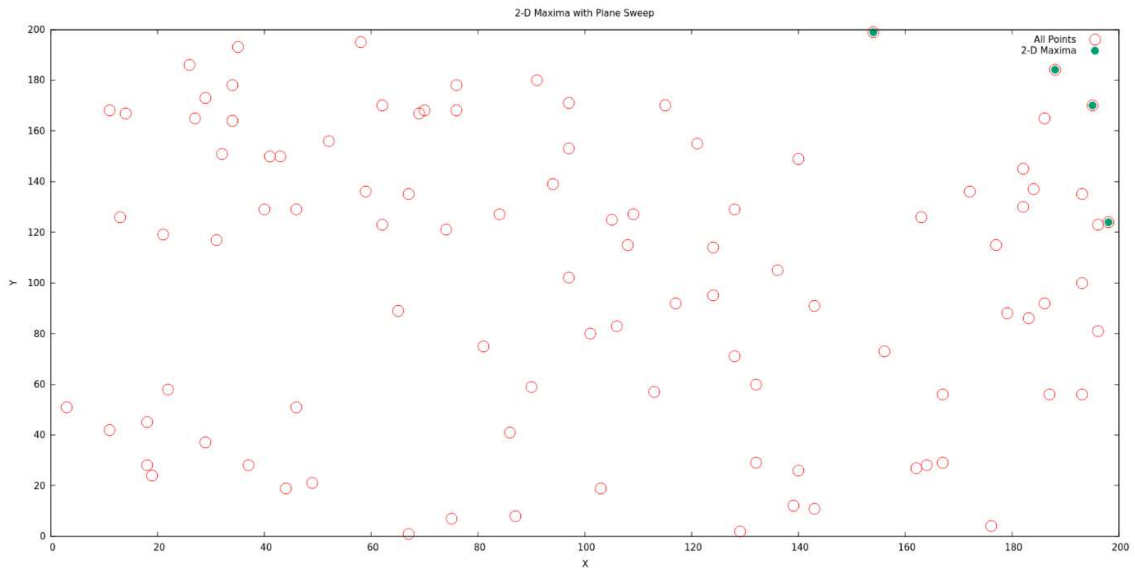


The above plot depicts the time-dataset curve for the brute force algorithm and another curve $y = x^2/100000$ to get a comparative grasp of the situation.

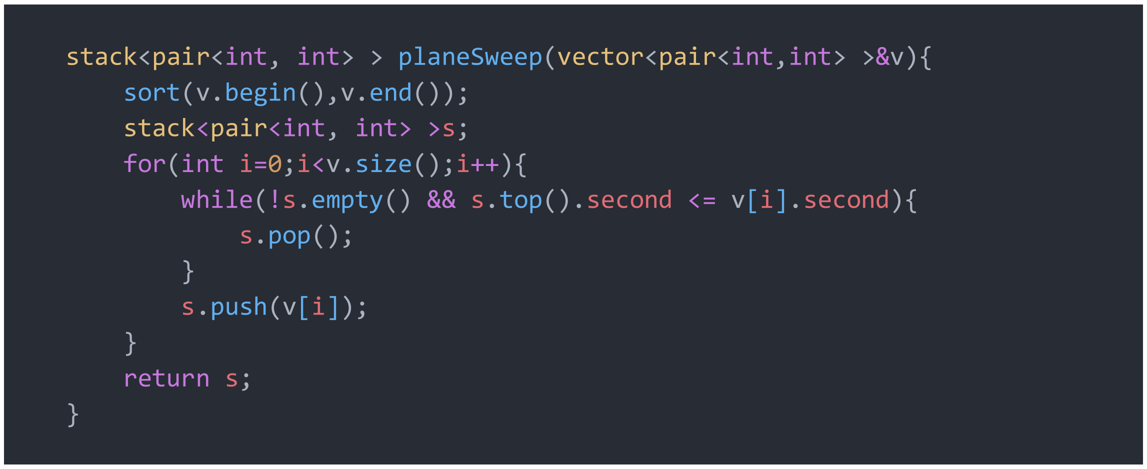
The x-axis denotes the length of the dataset where all the points in each dataset length are unique and the y-axis depicts the time required for the algorithm in milliseconds.

Plane Sweep Algorithm:

This approach involves sorting the dataset consisting of various points in an order (here ascending) and then looping through them while comparing them to the point on the top of the stack while simultaneously popping the elements from the stack if they are being dominated by the current point.

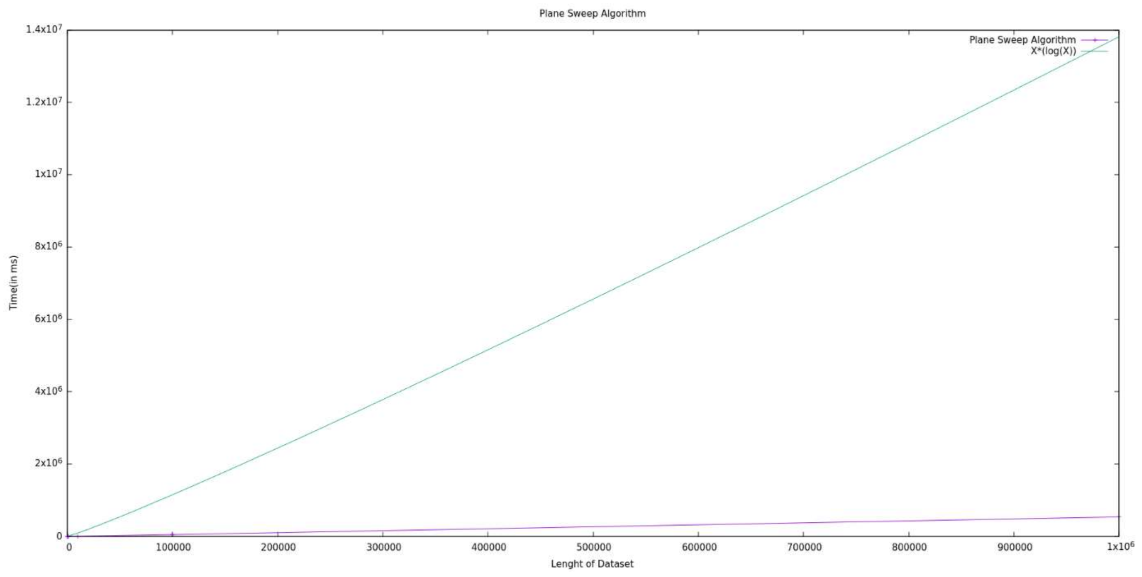


After looping over the dataset completely, the residual points that are still available in the stack are the points of 2-D Maxima.



```
stack<pair<int, int> > planeSweep(vector<pair<int,int> >&v){
    sort(v.begin(),v.end());
    stack<pair<int, int> >s;
    for(int i=0;i<v.size();i++){
        while(!s.empty() && s.top().second <= v[i].second){
            s.pop();
        }
        s.push(v[i]);
    }
    return s;
}
```

Here the sorting algorithm sorts the points in the increasing order of their x-coordinates. The sorting algorithm has the time complexity of O(n*log(n)) where n is the length of the data set. Now the outer loop always is executed for n times and when we look closely at the inner contents of the loop, we can see that at most 2*n operations can be performed inside the loop as at most n elements can be pushed into the stack and similarly at most n elements can be popped out of it. This results in the time complexity of loop being equivalent to O(n). When we consider the overall time complexity of the algorithm, we get the expression T(n) = O(n*log(n)) + O(n) and since O(n*log(n)) is the dominating term, the overall time complexity of the algorithm is considered to be O(n*log(n)).

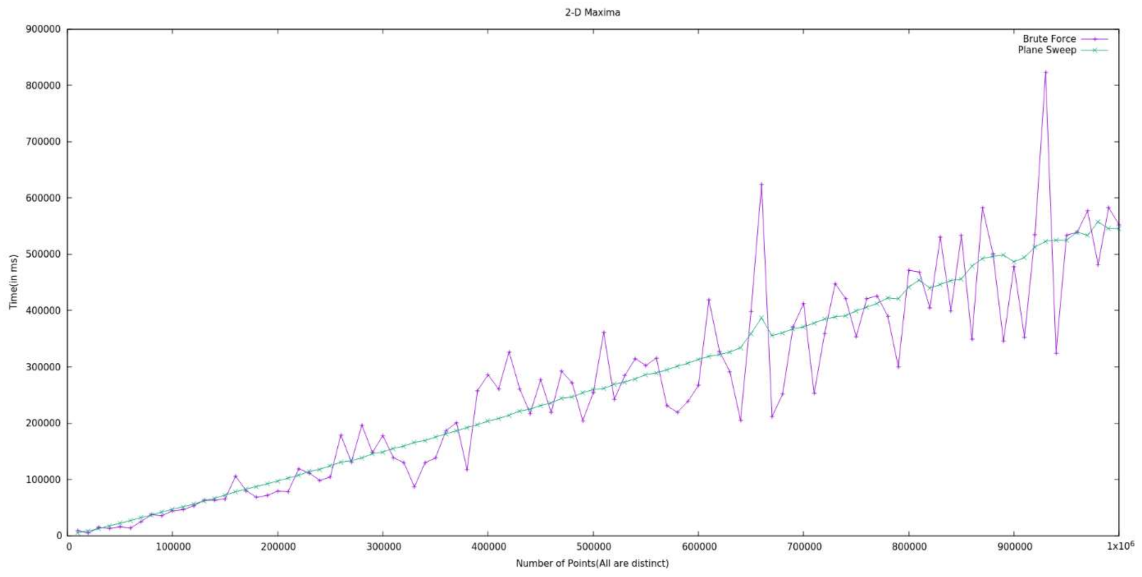


The above plot shows the curve of time required by the plane sweep algorithm against the length of dataset. The curve of y = n*log(n) is also plotted for comparison.

Comparative Analysis:

The time required by the plane sweep algorithm can be further optimized by using appropriate sorting methods as the maximum of its time is being utilized in the sorting of the points. The brute force method however cannot be optimized further and is only suitable for small datasets.

Here is a plot depicting the comparative behavior of the brute force and the Plane Sweep Algorithm over a range.



As it can be seen from the above plot, the Plane Sweep algorithm is far more consistent as to how much time is required as compared to the brute force algorithm. And with increase in the length of datasets, the brute force algorithm requires a considerably larger amount of time as compared to the plane sweep algorithm.