CSCI 6212: Design and Analysis of Algorithm – Project 2

Shubham Jadhav | G30570862 | Oct 11, 2022

1. Problem Statement

Let P be a set of n points in a 2-dimensional plane. A point p belongs to P is Pareto-optimal if no other point is both above and to the right of p. The sorted sequence of Pareto-optimal points describes a top-right staircase with the interior points of P below and to the left of the staircase. Describe a divide-and-conquer algorithm to compute the staircase of P in O (n log n) time.

2. Theoretical Analysis

To derive a divide-and-conquer algorithm, we need to break down the problem into smaller sections and finally merge the results. We can use splitting/two-way divide method and steps for the approach is as follows:

1. For the list of co-ordinates we have, choose point ***p*** by taking the point with maximum sum of their x and y coordinates, which eventually gives us the guaranteed Pareto optimal point (max(p.x +p.y))
2. This ***p*** point now acts as a pivot and now we can split the graph in 4 sections. There won't be any points to the right-top of ***p***, as it is Pareto optimal. Similarly, we can ignore points to the left-bottom sections. As all these points cannot be optimal as they fall under point ***p***. This way, we will get two list: one to the left-top of ***p*** and one to the right-bottom of ***p***
3. To consider maximum case, let's assume there are no points to the left-bottom of ***p***
4. Now, we have two list of points, and we can recursively follow the steps 1-4 for each of the list, until there is only one point in the corresponding list (recursive function call)

<u>To find time complexity of the selected algorithm:</u>

1. To find the point having maximum *x+y* coordinate will take **n** number of scans i.e., **O(n)** time complexity (*step 1 in approach*)
2. Similarly, to divide the list of points in sections i.e., points to ignore, top-left to ***p***, and bottom-right to ***p*** will take **n** number of scans which is **O(n)** time complexity (*step 2 in approach*)
3. To consider maximum time complexity, let's assume there are no points to ignore; the left-bottom of ***p.*** The list of **n** points will be divided in 2 list of $\boldsymbol{\frac{n-1}{2}}$ points, for simplicity let's take $\boldsymbol{\frac{n}{2}}$ points in each list (*step 3,4 in approach*)
4. To calculate time complexity of the recursive calls, the recurrence relation is **T(n) = 2T(n/2) + 2O(n)**

<u>By using master's theorem:</u>

Comparing with recurrence relation with T(n) = a T(n/b) + f(n), we get
a = 2, b = 2, f(n)=n

then, $c = n^{log_b a} = n^{log_2 2} = n$

<u>Case 2 of master's theorem:</u> if f(n)= $n^{log_b a}$, then there are log(n) terms, and each term is same as f(n).
Therefore, T(n) = f(n)*log(n) = (2n) * log(n); which is same as **nlog(n)**

The final time complexity of the computing the staircase using Divide and Conquer algorithm is **O(nlog(n)).**

3. Experimental Analysis

3.1 Program Listing

For experimental analysis, a python program was coded using the sample code from the project question. The program was executed with values of n as; 10^2, 10^3, 10^4, 10^5, and 10^6. The program also calculates the value of the theoretical time complexity derived above for the same n values. The values of theoretical, experimental, and adjusted values of theoretical are presented in sub section 3.3.

3.2 Data Normalization

The theoretical and experimental results for selected n values vary and cannot be compared directly or does not present a viable insight. Therefore, we need to derive a scaling constant to adjust either or the list of values. To derive the constant, we calculated the mean of both list of time values and found the ratio as follows

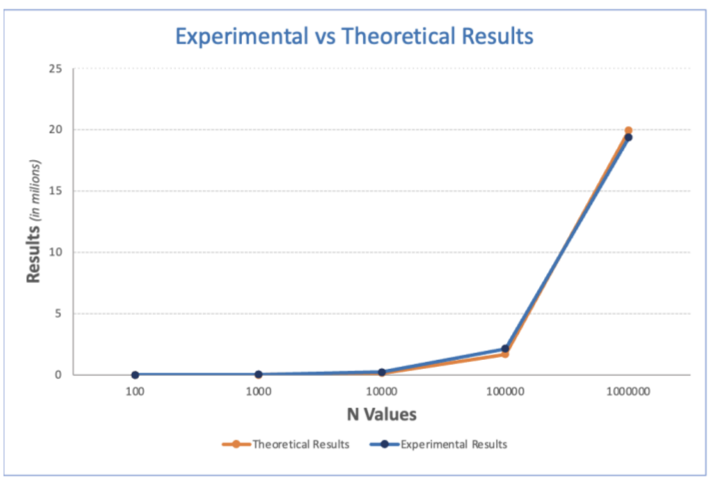
Scaling constant = $\frac{\textit{mean of experimentally derived time values}}{\textit{mean of theoretically derived time values}} \approx 4300000$

Note: we cannot calculate a fixed constant as experimental time values keeps on changing

3.3 Output Numerical Data

n	Theoretical	Experimental	Adjusted Experimental
100	664.38	4.11 x e^{-4}	1801.19
1000	9965.78	3.44 x e^{-3}	15126.70
10000	132877.12	5.42 x e^{-2}	238007.73
100000	1660964.04	4.84 x e^{-1}	2125015.18
1000000	19931568.56	4.41	19356089.09

3.4 Graph



3.5 Graph Observation

It can be seen from the above graph that, as value of n increases both, theoretical as well as experimental results poses the same pattern/behavior; increases with value of n.

4. Conclusion

In conclusion, derived theoretical complexity of the program seems accurate and consistent when compared with the normalized experimental complexity results using scaling constant from the observations and insights gained from the visualization in section 3.4.