Project 2

Sean Large G3512581) | sean.large1@gmail.com

1. **Problem Statement**

Staircase/Pareto-optimal points. Compute the a staircase on a 2D plane with a list of P points. See question 12 of ch 4 for more details

1. **Implementation Characteristics**

There are a few ways to implement this. My implementation starts with the X point furthest to the right then works its way left picking up the *highest* Y. To do that we need to: 1. Merge sort by x: ϴ(n log n) in descending order 2. Starting at the point with biggest X value, put its Y into a variable. 3. push these points on a list. 4. Go to the next largest X point, if its Y is bigger than the Y in the variable mentions in step 2, then repeat step 3. Otherwise do nothing. 4. Iterate over remaining points and do steps 1-3: O(n)

1. **Experimental Analysis**

## Programe Listing

See code in github or in blackboard <https://github.com/seanlarge/Staircase.git>

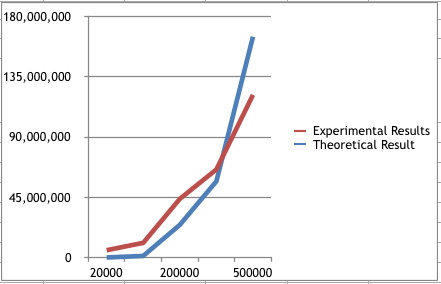
## Data Normalization Notes

###### Yes, 0.000115761710073054. I divided the average of experimental by the average of theoretical from table below

## Output Numerical Data

|  |  |  |
| --- | --- | --- |
| N | Experimental | Theoretical O( n \* (n log n)) |
| 20000 | 5624855 nanoseconds | 1720411998.27 |
| 50000 | 11120348 nanoseconds | 1720411998.27 |
| 200000 | 43940027 nanoseconds | 212041199827 |
| 300000 | 65882691 nanoseconds | 492940912925 |
| 500000 | 121531702 nanoseconds | 1.4247425E+12 |

## Graph



Type to enter a caption.

## Graph Observations

My Analysis was incorrect. My thought of O(n \* (n log n)) appears to grow much faster than experimental results for n >300,000. Up until that point they are pretty similar but for numbers greater than 500,000 theoretical grows much bigger

1. **Conclusions**

The algorithm is actually O(n\*h) because the sort is n log n and then you pick h times a point while scanning the reaming n items.O(nh/2) + O(nlogn)