Xuewen Sherry Li
Winter 2017

CS130B Programming Assignment #1: Closest Pairs

This table shows brute, basic, and optimal algorithm performance over various input sizes.

Input Size (N)	**Brute (seconds)**	**Basic (seconds)**	**Optimal (seconds)**
100	0.007	0.005	0.006
1,000	0.068	0.014	0.016
10,000	5.798	0.106	0.126
100,000	626.24	1.177	1.432
1,000,000	N/A	26.409	27.619
10,000,000	N/A	204.65	233.847

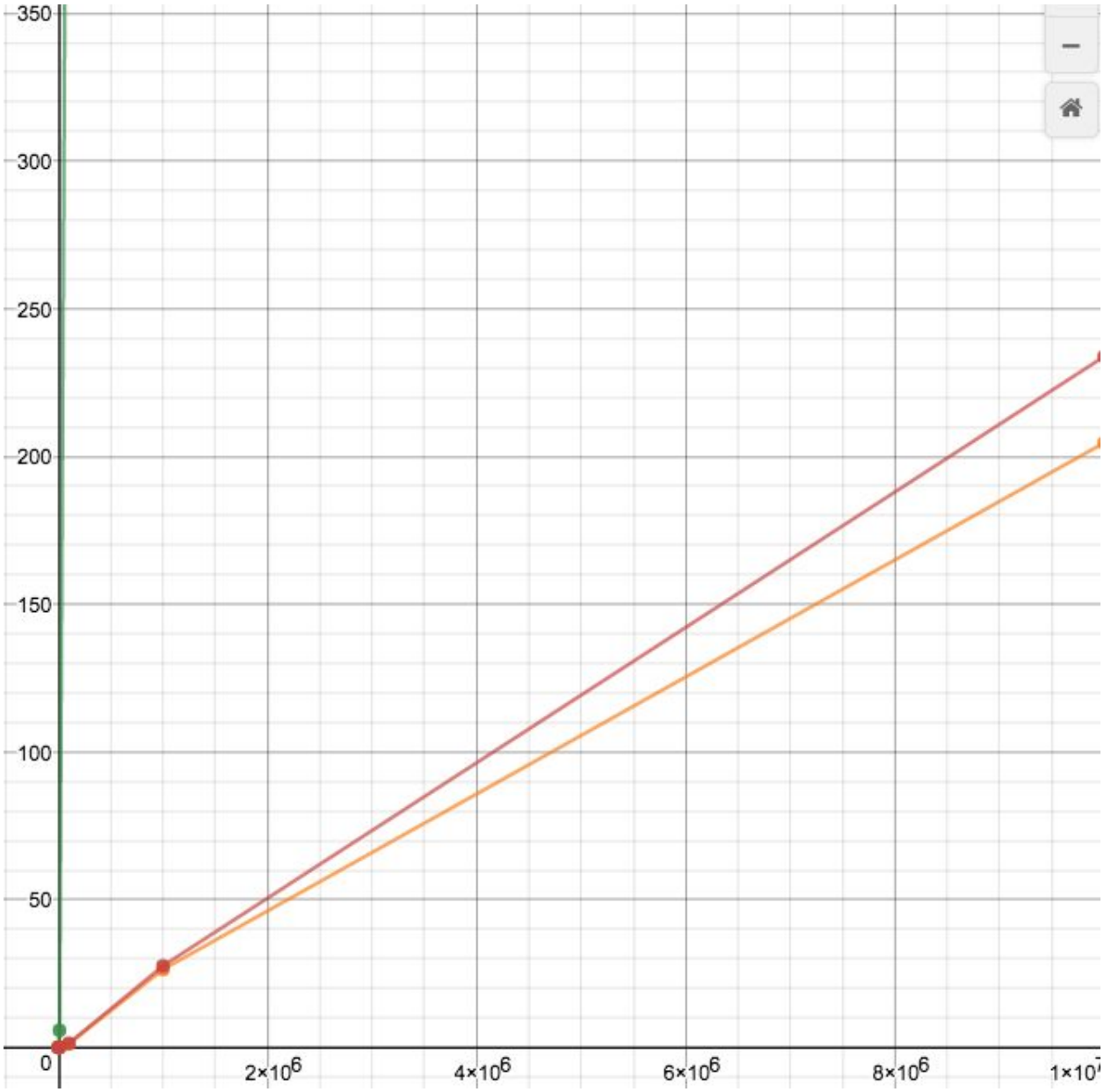
The graph below plots these points.
In general, my algorithms performed in this trend: brute < optimal <= basic. My optimal performed slightly slower than basic, which was an unexpected result. Possible causes could be that my optimal function calls a separate recursive helper to perform the algorithm, and this recursive helper needs to split the ySorted list into left and right subvectors each time it calls itself (to compute left hand side result and right hand side result). To split, I iterate through each point in ySorted list and pass it into a “left subvector” if its x coordinate is smaller or equal than the median, and into a “right subvector” if its x coordinate is greater. This iterative step is expensive and does not happen in basic, which could be why my optimal is slower than basic.

Key:

Green - brute

Orange - Basic

Red - Optimal


350
300
250
200
150
100
50
0
2×10^6
4×10^6
6×10^6
8×10^6