Kenneth Toombs
CS130B Programming Assignment 1

Runtime Analysis of Algorithms

Brute Force algorithm:
N = 10^2 : 0m0.479s
N = 10^3 : 0m0.840s
N = 10^4 : 0m26.975s
N = 10^5 : 44m31.396s

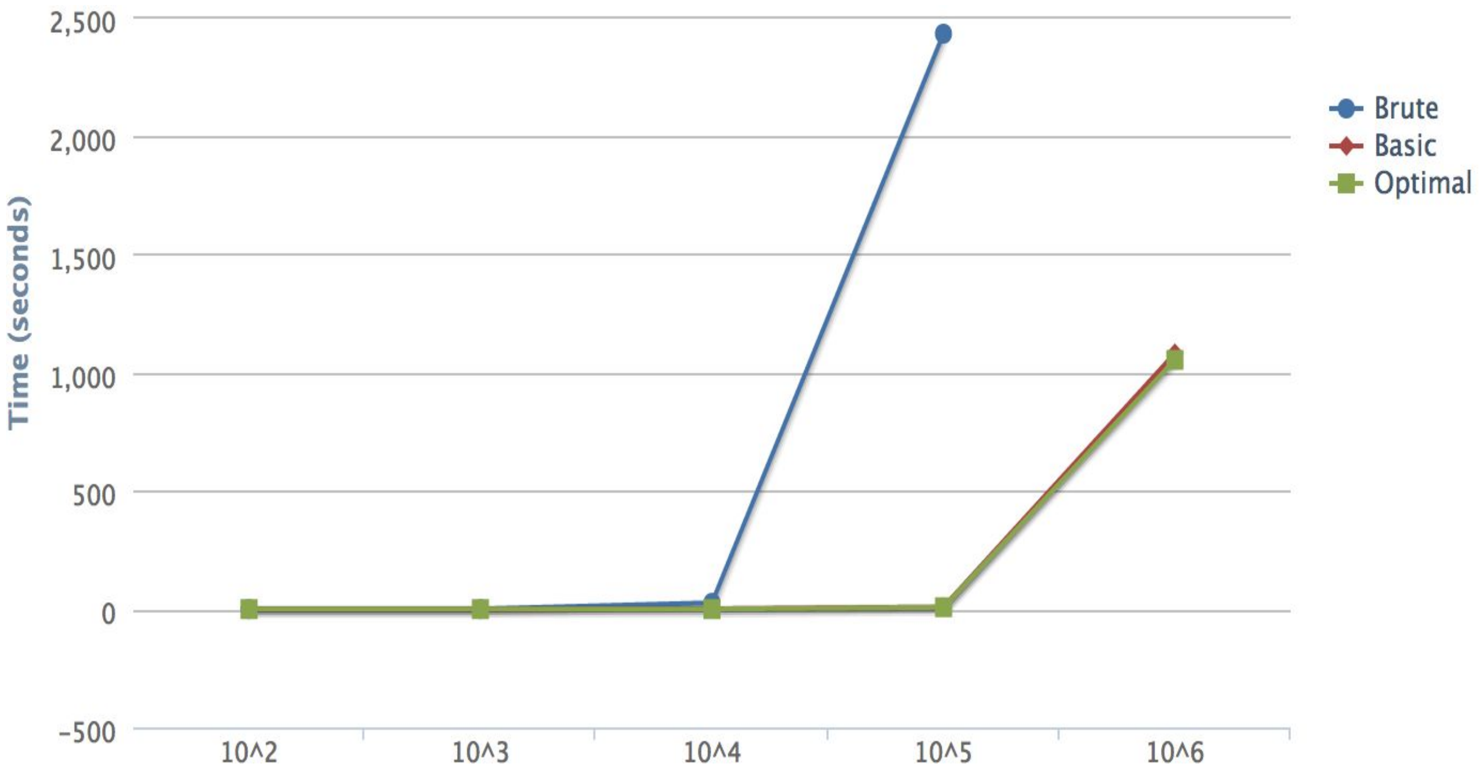
Basic algorithm:
N = 10^2 : 0m0.454s
N = 10^3 : 0m0.589s
N = 10^4 : 0m1.079s
N = 10^5 : 0m8.635s
N = 10^6 : 17m26.493s

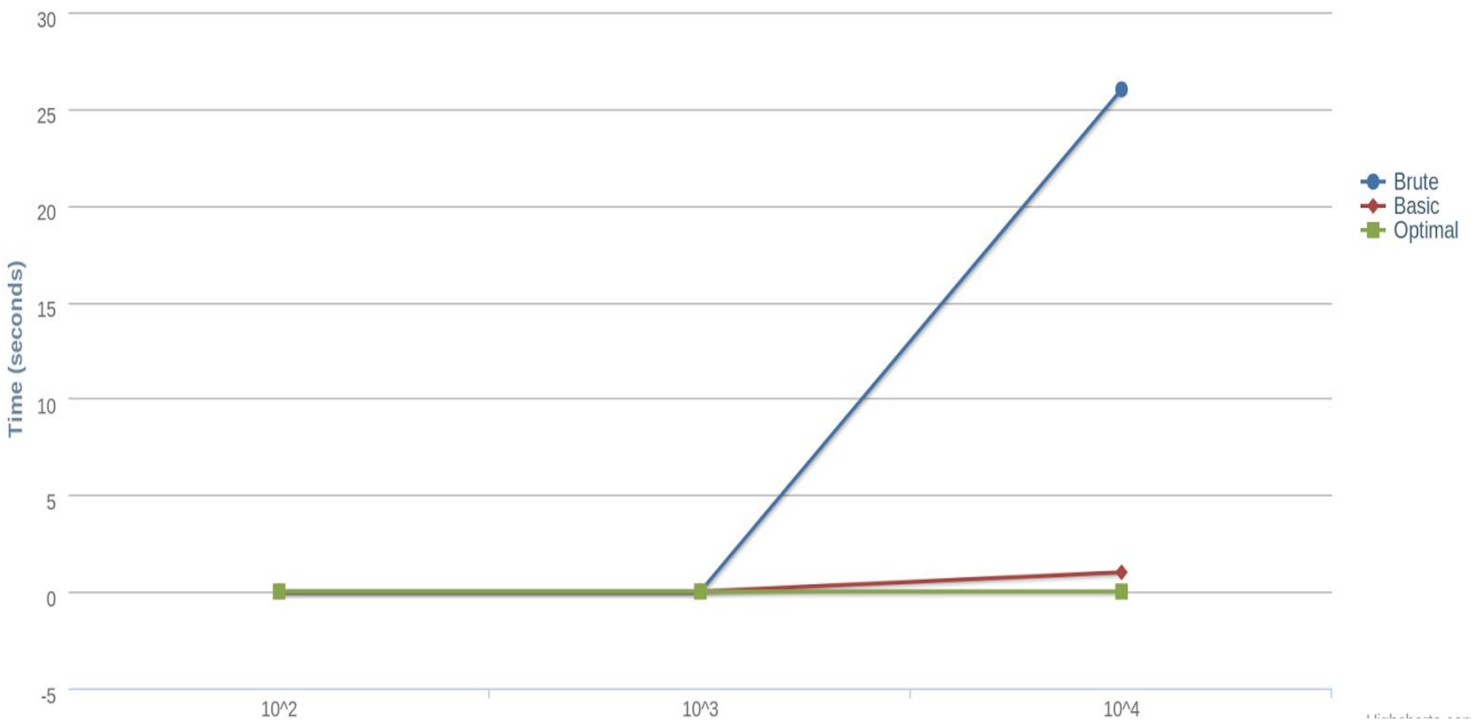
Optimal Algorithm:
N = 10^2 : 0m0.108s
N = 10^3 : 0m0.166s
N = 10^4 : 0m0.498s
N = 10^5 : 0m7.358s
N = 10^6 : 17m13.319s

Given the running times of these algorithms based on the input size, the pattern of the run times is expected. In other words, the basic and optimal algorithms execute much faster than the brute force algorithm. This can easily be seen once the input size reaches 10^4 and 10^5. Furthermore, the optimal algorithm is barely, but slightly, faster than the basic algorithm on the same input sizes. This is because the optimal algorithm sorts the inputs only once, whereas the basic algorithm sorts them on each call. On very large inputs, the algorithms seem to run a bit slower than expected. This might be due to the randData function running thousands of times in order to generate random input points. This would add some extra execution time, making it seem like the algorithms are running slower than expected when " time ./randData [n] | ./closestPair [algorithm] " is executed. Another possible reason could be that Java generally is a little slower than, say, C++. This is partially due to how some of the libraries are written, array accesses, and memory usage to name a few. Overall, the graph of the brute force algorithm appears exponential, and the graphs of the optimal and basic algorithms appear logarithmic.

Below are graphs that compare the running times of each of the algorithms based on the input size. The lower graph is to give a better idea of how the running times of the algorithms compare on the inputs 10^2, 10^3, and 10^4 since the first graph does not show that interval in great detail.

Input Size Vs. Runtime


Time (seconds)
10^2
10^3
10^4
10^5
10^6
Brute
Basic
Optimal


Time (seconds)
10^2
10^3
10^4
Brute
Basic
Optimal