Ryan Everett

CS251

Pattern Recognition Analysis

| Brute | | |
|---|---|---|
| **Operation** | **Frequency** | **Notation** |
| Array Declaration | N | ~N |
| Scan Points | N+1 | ~N |
| Sort Array | nlog(n) | ~Nlog(n) |
| Collinear Check | (n-3)(n-2)(n-1)(n) | 1/24 * $N^4$ |

| Fast | | |
|---|---|---|
| **Operation** | **Frequency** | **Notation** |
| Array Declaration | N | ~N |
| Scan Points | N+1 | ~N |
| Sort Array by coordinates | nlog(n) | ~Nlog(n) |
| Array Copy | n(n-3) | ~1/6 * $N^2$ |
| Sort Array by Angle PASS1 | (n-3)*n | ~1/6 * $N^2$ |
| Sort Array by Angle PASS2 | n*n | ~$N^2$ |
| Check Collinear | n * i *(i<n)* | ~N |

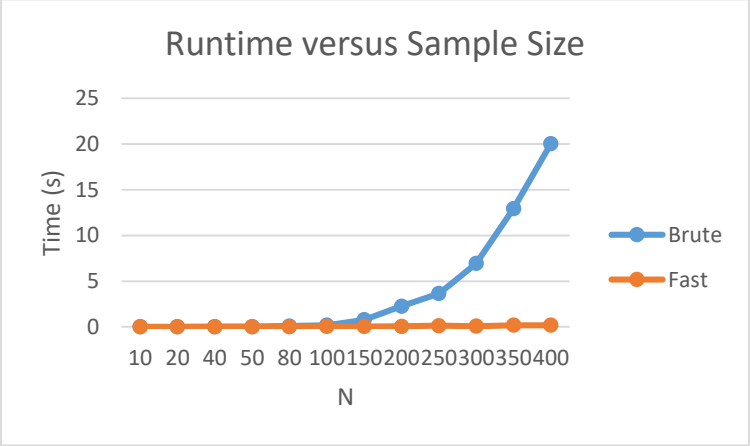
Brute: $\sim\frac{N^4}{24}$ Fast: $\sim\frac{4N^2}{3}$

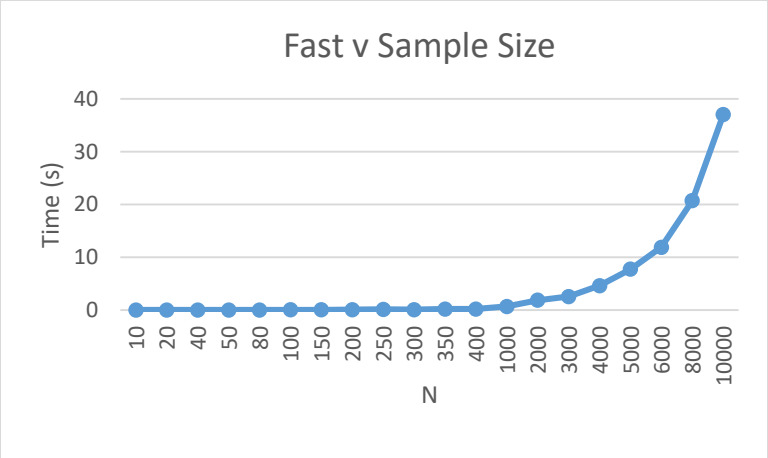
Arrays.sort = nlog(n) by the Documentation

The result of the Brute algorithm stems from the fact that the Collinear Check easily outgrows every other operation. The result of the Fast algorithm is also derived from a similar fashion. Operations Array Copy, both passes of Sort Array, and Check Collinear are all executed N times within an N loop. Check Collinear executes N times at its worst case, but averages far less. The rest sum up to $\frac{6N^2}{6}+\frac{N^2}{6}+\frac{N^2}{6}=\frac{4N^2}{3}$.

| N | Brute(N) | Fast(N) |
|---|---|---|
| **10** | 0.005563s | 0.003563s |
| **20** | 0.012s | 0.007s |
| **40** | 0.018s | 0.011s |
| **50** | 0.024s | 0.013s |
| **80** | 0.1s | 0.021s |
| **100** | 0.177s | 0.041s |
| **150** | 0.776s | 0.043s |
| **200** | 2.272s | 0.067s |
| **250** | 3.639s | 0.135s |
| **300** | 6.931s | 0.074s |
| **350** | 12.926s | 0.192s |
| **400** | 20.036s | 0.185s |
| **1000** | 679.193s | 0.671s |
| **2000** | | 1.854s |
| **3000** | | 2.568s |
| **4000** | | 4.615s |
| **5000** | | 7.746s |
| **6000** | | 11.9s |
| **8000** | | 20.71s |
| **10000** | | 37.038s |

The difference between the two algorithms is clear from the get-go. The Brute force algorithm begins to slow down very fast; N=400 is the largest tested N value that is sub-200 seconds. Whereas the Fast Algorithm can go all the way to N=10,000 without coming close to the 200 second ceiling. Looking at the Runtime graph of Brute versus Fast, it's hard to see the difference since Fast stays under one second for the N values. It's better to plot Fast on its own graph to get a proper view of how it grows.






In order to estimate the future runtimes of the two algorithms, we have to create the log-log regression equation of the values.

Brute: $y = 4.528409489\ ln(x) - 3.313432392$

Fast: $y = 2.90786942\ ln(x) - 2.440491734$

Further decomposing these formulas to retrieve non-logarithmic values gives us.

Brute: $T(N) = 0.1005906 \cdot N^{4.528409489}$; Fast: $T(N) = 0.184221 \cdot N^{2.90786942}$

Using N=1,000,000 gives us $1.4894\ x\ 10^{26} s$ for Brute and $5.1589\ x\ 10^{16} s$ for Fast.