Algorithm Design

P1.1) Basic operation: IF (array[i] > array[j])

- This is a comparison

Number of executions of basic operation:

$$\sum_{i=0}^{n}\sum_{j=i+1} 1 = \sum_{i=0}^{n}[n-(i+1)+1] = \sum_{i=0}^{n} n-i = \sum_{i=0}^{n} n - \sum_{i=0}^{n} i$$

$$= n\sum_{i=0}^{n} 1 - \frac{1}{2}n(n+1) = n(1+n) - \frac{1}{2}n(n+1) = \frac{2n(1+n)-n(n+1)}{2}$$

$$= \frac{2n+2n^2-n^2-n}{2} = \frac{n+n^2}{2} = \boxed{\frac{n(1+n)}{2}}$$

Efficiency Class

$$\frac{n(1+n)}{2} \simeq \frac{1}{2}n^2 \quad \boxed{\in \Theta(n^2)}$$

- This makes sense as the algorithm itself is a for loop within a for loop thus creating a complexity of n^2.

P1.2) Basic operation: Comparison

Number of executions of basic operation and efficiency class:

$n = 2^k$

$C(n) = 2C(n/2) + n/2, \quad C(1) = 0$

$C(n) = 2C(n/2) + n/2 \quad [C(n/2) = 2C(n/4 + n/4)]$

$= 2(2C(n/4) + n/4) + n/2$

$= 2^2 C(n/2^2) + n/2 + n/2 \quad [C(n/4) = 2C(n/8) + n/8]$

$= 2^2(2C(n/8) + n/8] + n/2 + n/2$

$= 2^3 C(n/2^3) + n/2 + n/2 + n/2$

$= 2^3 C(n/2^3) + 3\frac{n}{2}$

$C(n) = 2^i C(n/2^i) + i\,n/2$

$n = 2^k \quad i = k$

$C(n) = 2^k C(n/2^k) + k\frac{n}{2}$

$C(1) = 0 \quad k = \log_2 n \rightarrow C(n) = \frac{1}{2} n \log_2 n \in \Theta(n \log n)$

Master Theorem

$T(n) = T(n/2) + T(n/2) + n$

$T(n) = 2T(n/2) + O(n)$ $a=2$ $b=2$

$F(n) = O(n)$

Comparison:

$n^{\log_b(a)} \Rightarrow O(n)$

$n^1 = O(n)$

- we see that the cost of $F(n)$ and the sub problems are the same so this is:

$T(n) = O(n \log n)$

P1.3) Execution Time

Brute force:	21477 ms	21234 ms | 20984 ms
MergeSort:	74 ms	51 ms | 55 ms

Theoretical Analysis:

Brute force: This program loops through an entire array of integers and compares two elements and swaps them if $i > j$.

MergeSort: this program splits the array in half recursively, until only two elements remain in an array. It will then sort them into asleading order and merge them back together.

P2.1) Basic operation: comparison within 3 for loops

Number of executions of basic operation and efficiency class

$$\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=0}^{n} 1 = \sum_{i=0}^{n}\sum_{j=0}^{n} n+1 = \sum_{i=0}^{n}\sum_{j=0}^{n} n + \sum_{i=0}^{n}\sum_{j=0}^{n} 1$$

$$= \sum_{i=0}^{n} n\sum_{j=0}^{n} 1 + \sum_{i=0}^{n} (1+n) = \sum_{i=0}^{n} n(n+1) + \sum_{i=0}^{n} 1 + \sum_{i=0}^{n} n$$

$$= \sum_{i=0}^{n} n^2 + \sum_{i=0}^{n} n + (n+1) + n(n+1) = n^2(n+1) + n(n+1) + (n+1) + n(n+1)$$

$$= n^3 + n^2 + n^2 + n + n + 1 + n^2 + n$$

$$= n^3 + 3n^2 + 3n + 1$$

Efficiency:

$$n^3 + 3n^2 + 3n + 1 \simeq n^3 \in \Theta n^3$$

P2.2) Basic operation: comparison

Number of executions of basic operation and efficiency class

$n = 2^k$

$C(n) = 2C(n/2) + n/2, \quad C(1) = 0$

$C(n/2) = 2C(n/4 + n/4)$

$= 2(2C(n/4 + n/4)) + n/2$

$= 2^2 C(n/2^2) + 2\, n/2$

$C(n) = 2^i C(n/2^i) + i\, n/2$

$n = 2^k \quad i = k$

$C(n) = 2^k C(n/2^k) + k\, n/2$

$C(1) = 0 \quad k = \log_2 n$

$C(n) = \frac{1}{2} n \log_2 n \in \Theta(n \log n)$

Master Theorem

$T(n) = T(n/2) + T(n/2) + n$

$T(n) = 2T(n/2) + O(n)$ $a=2$ $b=2$

$F(n) = O(n)$

Comparison:

$n^{\log_b(a)} \rightarrow O(n)$

$n^1 = O(n)$

The cost of F(n) and the sub Problem are the same so this is

$T(n) = O(n \log n)$

P2.3) Execution Time

Brute Force:	313 ms	316 ms	298 ms
Quick Hull: (Divide and Conquer)	1 ms	12 ms	2 ms

* Run with much less coordinates as the initial 30000 causes the brute force algorithm to take an extensive amount of time

Theoretical Analysis

Brute Force: For a given array of x and y coordinates a line is calculated For two coordinates and the location of other Points relative to this line must be found.

Divide and Conquer: A line is calculated for the smallest and largest X coordinate. The Convex hull Points can be Found on either side of this line recursively. If a coordinate is found with a max distance from this line then a new line is formed with this coordinate and we recur on either side of this new line.