Kelby Hubbard

CS 411 F21 HW4

1a) $T(n) = 4T(n/2) + n,\ T(1) = 1$

$a = 4,\ b = 2,\ f(n) = n,\ d = 1$

$a > b^d \Rightarrow T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_2 4}) = \Theta(n^2)$

1b) $T(n) = 4T(n/2) + n^2,\ T(1) = 1$

$a = 4,\ b = 2,\ f(n) = n^2,\ d = 2$

$a = b^d \Rightarrow T(b) \in \Theta(n^d logn) = \Theta(n^2 logn)$

1c) $T(n) = 4T(n/2) + n^3,\ T(1) = 1$

$a = 4,\ b = 2,\ f(n) = n^3,\ d = 3$

$a < b^d \Rightarrow T(n) \in \Theta(n^d) = \Theta(n^3)$

2) Merge sort on average taking *nlogn* comparisons and binary search taking on average *logn* compared to a sequential search taking on average *n/2* we can set up an equation to solve.

x = smallest number of searches needed

$$n\ log\ (n) + x\ log\ (n) \le x\ (n/2)$$

$$\frac{n\ log\ (n)}{n/2 - log\ (n)} \le x$$

$$n = 1,000,000 \Rightarrow x \ge 40$$

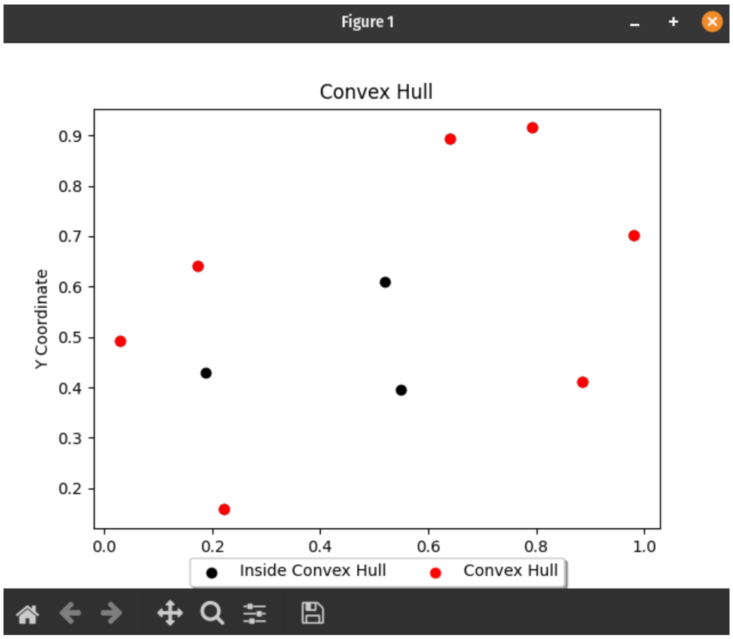
At least 40 searches will be needed to justify the time spent presorting an array of 1,000,000 elements.

3A)

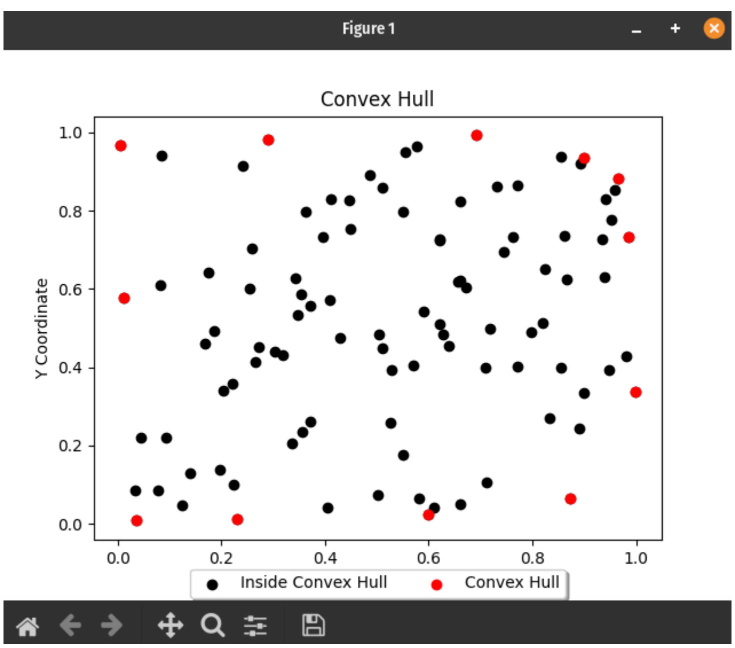
n (2D Randomly Distributed Points)	Run Time
10	0.00011444091796875 seconds

100	0.0009560585021972656 seconds
1000	0.023084402084350586 seconds
10000	0.28388452529907227 seconds

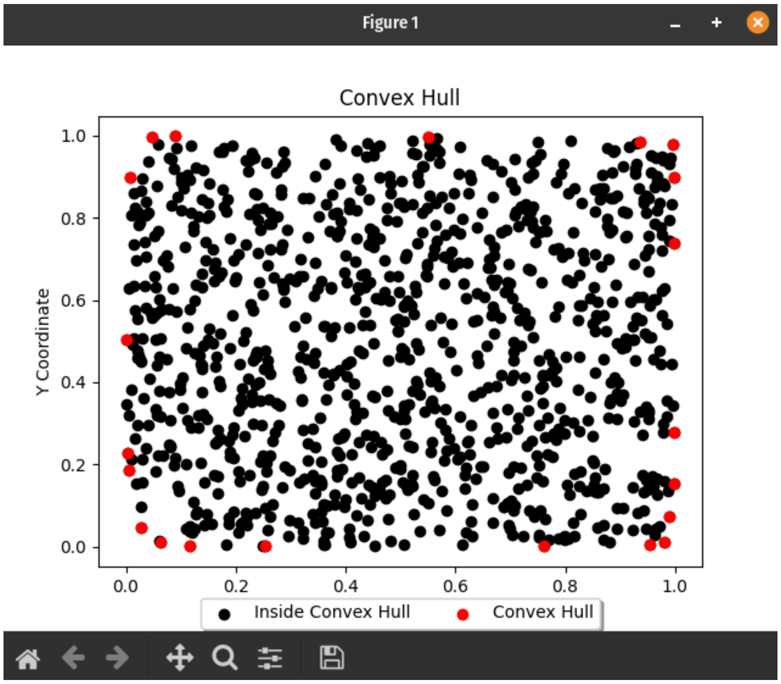
3B)
n = 10:



n = 100:



n = 1000:



n = 10000: