Danny Tan
CS 2134 HW 1

1a) O(n)
1b) O(n^2)
1c) O(n^3)
1d) O(n^2)
1e) O(n^3)
1f) O(n)
1g) O(n^2 log(n))
1h) O(n!)
1i) O(n!)

2a) O(1)
2b) O(n)
2c) O(n^2)
2d) O(n^2)
2e) O(n^3)
2f) O(n)
2g) O(n^2)
2h) O(n^3)
2i) O(log(n))
2j) O(n log(n))
2k) O(log(n))

3) 1. n^3
2. n^2
3.$n^{1.5}$
4.n $\log^2$ n
5. n log n
6. n and n/2 is same
7. $\sqrt{n}$

4) $2^{13} / 2^{11} = 4$

a) 4 * 0.05 s = 0.2 s
b) 4^2 * 0.05 s = 0.8 s
c) 4^4 * 0.05 s = 12.8 s

5) O(n^2) because every time n is increased by 2, the time is increase by 4 or 2^2

6) $3n^2 + 2n \log n + 6n + 19 \leq 4n^2$ for all $n \geq 13$ therefore by the definition of the Big O notation, $3n^2 + 2n \log n + 6n + 19 = O(n^2)$

7)

N	maxSubSum1 O(n^3)	maxSubSum2 O(n^2)	maxSubSum4 O(n)
128	0.001069	4.6e-05	3e-06
256	0.007678	0.000171	3e-06
512	0.060455	0.000565	3e-06
1024	0.488088	0.002048	6e-06
2048	3.91782	0.008078	1.5e-05
4096	31.379741	0.035846	2.2e-05

8)

N	maxSubSum1 O(n^3)	maxSubSum2 O(n^2)	maxSubSum4 O(n)
256	0.008552	0.000184	0.000006
512	0.068416	0.000736	0.000012
1024	0.547328	0.002944	0.000024
2048	4.378624	0.011776	0.000048
4096	35.028992	0.047104	0.000096

9)

$2^{18} / 2^7 = 2^{11}$

For first algorithm $O(n^3)$, it will take $(2^{11})^3 * 0.001069 = 9.18 * 10^6$ seconds to compute $n = 2^{18}$.

For second algorithm $O(n^2)$, it will take $(2^{11})^2 * 4.6e\text{-}05 = 192.94$ seconds to compute $n = 2^{18}$.

For fourth algorithm $O(n)$, it will take $2^{11} * 3e\text{-}06 = 0.006144$ seconds to compute $n = 2^{18}$.

10)
For first algorithm $O(n^3)$, it will take 15 weeks, 1 day and 6 hours.
For second algorithm $O(n^2)$, it will take 3 minutes and 12.94 seconds.
For third algorithm O(n), it will take 0.006144 seconds.

11)

N	B	C	D	E
256	1e-06	0.000145	7.9e-05	0.035087
512	3e-06	0.000555	0.000292	0.272358
1024	2e-06	0.002118	0.001087	2.152884
2048	5e-06	0.008525	0.004226	17.419535
4096	1e-05	0.033426	0.017208	139.633662

12) True
13) True