1. Algorithm 1

Select k Alg #1

Line #	Cost	Times
1	3	$k+1$
2	3	$(n-2+2) + (n-3+2) \ldots = (n-1+1) + (n-2+1) \ldots + (n-k+1)$; $\sum_{i=1}^{k+1} 1 + \sum_{i=1}^{k+1} n-i = (k+1) + (k+1)n - \frac{(k+1)^2}{2} - \frac{k+1}{2}$
3	max(If) = 8 + = 23	Same as above w/o extra $\sum_{i=1}^{k+1} 1 \ldots = (k+1)n - \frac{(k+1)^2}{2} - \frac{k+1}{2}$
4	4	$\emptyset$
5	6	$\emptyset$
6	5	$\emptyset$
7	4	1

$$T(n) = 3(k+1) + 3\left[k+1 + (k+1)n - \frac{(k+1)^2}{2} - \frac{k+1}{2}\right] + 23\left[(k+1)n - \frac{(k+1)^2}{2} - \frac{k+1}{2}\right] + 4$$

Let $k = 5$

$$T(n) = 18 + 3[6 + 6n - 15 - 3] + 23[6n - 21]$$

$$T(n) = 18 + 18n - 45 + 138n - 483$$

$$\boxed{T(n) = 156n - 510 \quad \text{for } n \geq 5}$$

Algorithm 2

Select k - Alg #2

Line #	Cost	Times
1	2	k+1
2	2	k
3	4	$(n-k+1+1)+(n-k-2+1+1)+\dots+(n-2+1)$ $= \sum_{i=1}^{k} n-i + \sum_{i=1}^{k} 1 = kn - \frac{k^2}{2} - \frac{k}{2} + k$
4	4	same as above w/0 + one $= kn - \frac{k^2}{2} - \frac{k}{2}$
5	max(t) = 3+ = 17	$= kn - \frac{k^2}{2} - \frac{k}{2}$
6	4	~~0~~
7	6	~~0~~
8	4	~~0~~
9	4	1

$T(n) = 2(k+1)+2(k)+4[kn - \frac{k^2}{2} - \frac{k}{2} + k] + 4[kn - \frac{k^2}{2} - \frac{k}{2}] + 17[kn - \frac{k^2}{2} - \frac{k}{2}] + 4$

let $k=5$

$T(n) = 12 + 10 + 4[5n - \frac{25}{2} - \frac{5}{2} + 5] + 4[5n - \frac{25}{2} - \frac{5}{2}] + 17[5n - \frac{25}{2} - \frac{5}{2}] + 4$

$T(n) = 22 + 20n - 40 + 45n - 135 + 85n - 255 + 4$

$\boxed{T(n) = 150n - 404 \quad \text{for } n \geq 5}$

Algorithm 3

Select k Alg #3

Line #	Cost	Times
1	2	$k+1$
2	6	k
3	1	1
4	2	k
5	9	$k-1$
6	4	$(n-k-1+1+1)+(n-k-2+1+1)+\dots+(n-k+1+1+1)$
7	max(k) = 7 +	
8	6	$\emptyset$
9	1	$\emptyset$
10	2 *	$k-1 = k-2+1$
11	9 *	
12	4	1

$$T(n) = 2(k+1) + 6k + 1 + 2k + 9(k-1) + 4\left[n-k + \frac{(n-k)^2}{2} + \frac{(n-k)}{2}\right] + \left[7 + 6 + 1 + 2\left[k-1 + \frac{k^2}{2} + \frac{k}{2} - 1\right] + 9\left[\frac{k^2}{2} + \frac{k}{2} - 1\right]\right]\left[\frac{(n-k)^2}{2} + \frac{(n-k)}{2}\right] + 4$$

Let $k = 5$

$$T(n) = 12 + 30 + 1 + 10 + 36 + 4\left[n-5 + \frac{n^2 - 10n + 25}{2} + \frac{n-5}{2}\right] + \left[14 + 2\left[4 + \frac{25}{2} + \frac{5}{2} - 1\right] + 9\left[\frac{25}{2} + \frac{5}{2} - 1\right]\right]\left[\frac{n^2 - 10n + 25}{2} + \frac{n-5}{2}\right] + 4$$

$$T(n) = 89 + 4n - 20 + 6 + 7n - 35 + 4$$

$$\boxed{T(n) = 6n - 269 \quad n \geq 5}$$

Algorithm 4

Select k Algo #4

Base Case

Line #	Cost	Times
1	2	1
2	$25n+13$	1
3	5	1
5	6	1
6	4	1

Recursive Case

Line #	Cost
1	2
2	$25n+13$
3	$5 \rightarrow Max(If) = 11 + 9T(\frac{n}{2})$ ← checking costs
4	$T(\frac{n}{2})+3$
5	6
7	$T(\frac{n}{2})+9$

$$T(n) = \begin{cases} 25n+30 \\ T(\frac{n}{2})+25n+38 \end{cases} \simeq \begin{cases} 25n \\ T(\frac{n}{2})+25n \end{cases}$$

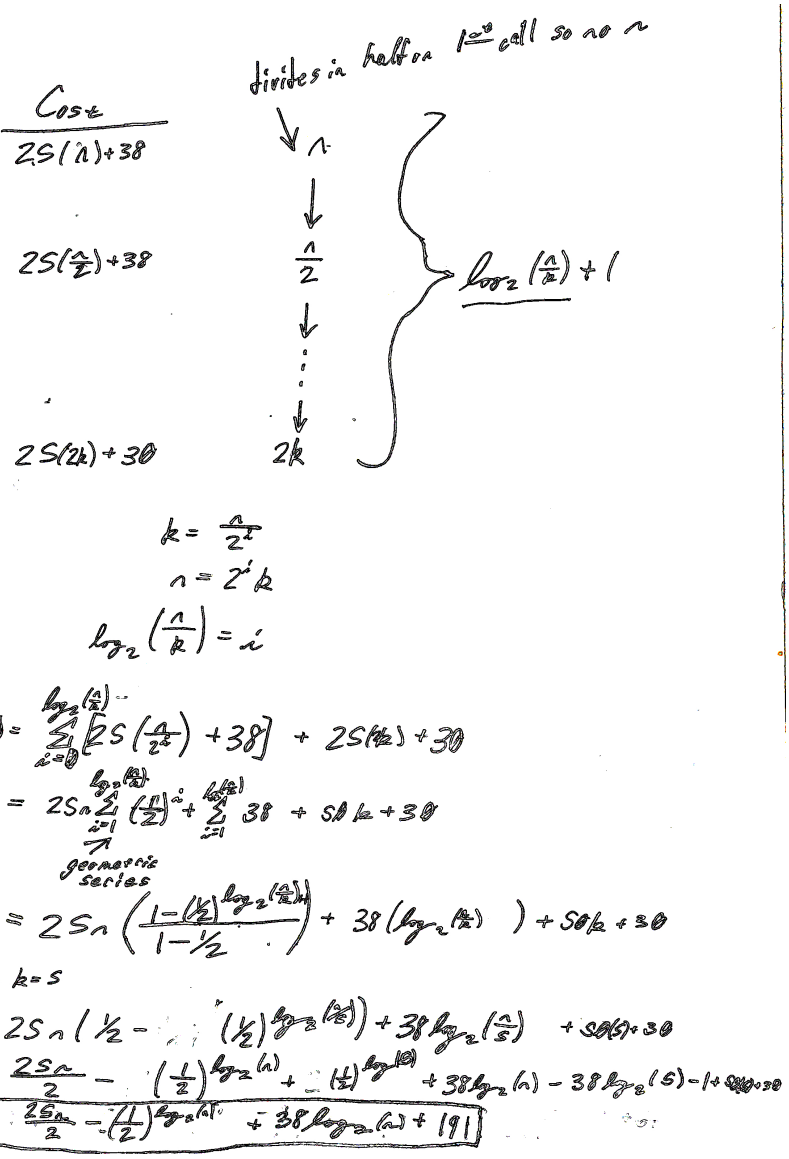
Partition

Swap = temp = var1
var1 = var2
var2 = temp

Line #	Cost	Times
1	4	1
2	13 → based using temp variable	1
3	2	1
4	4	n = (r−p+1)
5	max(if) = 5+. = 21	n−1
6	13	~~0~~
7	3	~~0~~
8	13	1
9	2	1

$T(n) = 4 + 13 + 2 + 4n + 21(n-1) + 13 + 2$

$T(n) = 25n + 13$



2. Each algorithm to within some error matches its given graph.

   Algorithm 1 matches on the slope well, but is off by a constant factor. However, this can be accounted for by system specific things such as stack and functional call time.

   Algorithm 2 has approximately the same slope as its real counterpart. The constant factor is within enough of an error bound to be a good approximation of the T(n).

Algorithm 3 has approximately the same slope as its real counterpart. The theoretical complexity is faster than its real complexity; however, this can be accounted for by the system specifics such as multiple processes running. This is reasonable due to the main difference being in a constant factor over all.

Algorithm 4 is the most markedly different between its real and theoretical T(n). This however is accounted for by its recursive nature. Any recursive algorithm will run slower than its theoretical due to system specifics such as stack frames and function calls. This explains the difference in slopes.

3. For the real running time of the algorithms, they were ordered least to most efficient: algorithm 1, algorithm 2, algorithm 4, algorithm 3.

4. For the theoretical running time of the algorithms, they were ordered least to most efficient: algorithm 1, algorithm 2, algorithm 4, algorithm 3.

5. Yes, both the theoretical and real complexity of the algorithms lead to the same ordering of the algorithms based on efficiency. This is to be expected. The only slightly odd exception is that the recursive algorithm 4 theoretically stayed in the same order. This was because its theoretical efficieny is linear and with a smaller constant factor than algorithm 3. However, algorithm's 4 logrithmic term boosted to a slower rate than algorithm 3.