Homework 4

1. Best case : ~~the complexity for best-case~~ ~~Because~~ The best-case time complexity for insertion sort is $O(n)$. For best-case arrays and larger and larger k values, insertion (*) sort is applied. Therefore the complexity is $O(n)$.

Average case: The average time complexity for insertion sort is $O(n^2)$ and for merge sort it's $O(n \log n)$. Therefore, as our algorithm uses both, $\forall$ $k \in \mathbb{N}$ the complexity will be $O(n/k \log n/k + k^2)$.

Worst case: time complexity for worst case merge-sort is $O(n \log n)$ and for insertion sort $O(n^2)$. As the k gets larger and larger, only insertion sort is applied (same as *). So complexity will be $O(n^2)$.

d) Since merge-sort has a better worse case (not as bad) as insertion sort, I'd choose $k = 1$ if ~~[illegible]~~ we are talking about avg case / worst case scenario arrays, as the $k = 1$ will mean merge sort $\Rightarrow O(n \log n)$

But insertion sort has a better best-case scenario. So if we have an already sorted array, I'd choose a big k, say $k = n$ so that insertion sort is applied $\Rightarrow O(n)$.

4.2

Master Theorem

$T(n) = aT(n/b) + f(n)$

$n^{\log_b(a)} \Leftrightarrow f(n)$

a) $T(n) = 36\,T(n/6) + 2n$

$n^{\log_6 36}$ ~~$\leq 2n$~~ $\Leftrightarrow n^2$

$n^{\log_b a} > f(n) \Rightarrow \Theta(n^2)$

b) $T(n) = 5\,T(n/3) + 17n^{1.2}$

$n^{\log_3 5} = n^{1.46}$ $\quad n^{\log_b a} > f(n) \Rightarrow \Theta(n^{1.46})$

~~c) $T(n) = 3\,T(n/5) + T(n/2) + 2n$~~

~~c) $T(n) = 12\,T(n/2) + n^2$~~

c) $T(n) = 12\,T(n/2) + n^2 \lg n$

~~$n^{\log_2 12} < n^2 \lg n$~~

$n^{\log_b a} = n^{3.58}$

$n^{3.58} > n^2 \ln n \Rightarrow O(n^{3.58})$