ADS 4

P.1/c - merge sorte complexity is

$\Theta(n \log n)$ $\quad T(n) = 2T(n/2) + O(n) : n>1$

in the variant algorithm the recursion stops a n/k subarrays, so the new recursion depth is: $\log_2(n/k) = \log_2(n) - \log_2(k)$

the new time complexity is now:
$T(n) = O(n \log n) - O(n \log k) + O(n)$

- for small k $\log k$ is insignificant and $O(n \log n)$ dominates
- for medium values it starts affecting significantly thus improving the performance due to efficiency in sorting nearly sorted arrays while worst case remains $O(\log(n))$
- large k, insersion sort dominates so the worst case becomes closer to $O(n^2)$ and best

case closer to $O(n)$.

d- explained in the text file.

P.2/ a- $T(n) = 36T(n/6) + 2n$

$f(n) = (2n)$

$O(n^{\log_6 36}) = O(n^2)$

$f(n) = O(n^{\log_6(36) - 1}) = O(n)$

so $T(n) = \Theta(n^{\log_6 36}) = \Theta(n^2)$

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

$f(n) = 17n^{1.2} = O(n^{\log_3 5 - 0,2})$

$\Rightarrow T(n) = \Theta(n^{\log_3 5}) \approx \Theta(n^{1,4})$

c- $T(n) = 12T(n/2) + n^2 \log(n)$

$f(n) = n^2 \log(n)$

$n^{\log_2 12} = n^{3,58}$

$f(n) = O(n^{3,58})$

$\Rightarrow T(n) = \Theta(n^{3,58})$

d. $T(n) = 3T(n/5) + T(n/2) + 2^n$

$f(n) = 2^n$ we use the recursion tree method :

the recurence grows expenentially because of 2^n at each level we sum an expenancialy growing term since 2^n daminates.

$T(n) = \Theta(2^n)$

e. $T(n) = T(2n/5) + T(3n/5) + \Theta n$

the recursion depth is $\simeq O(\log(n))$

the extra work is $O(n)$

summing across $\log n$ level :

$T(n) = \Theta(n)$.

d_