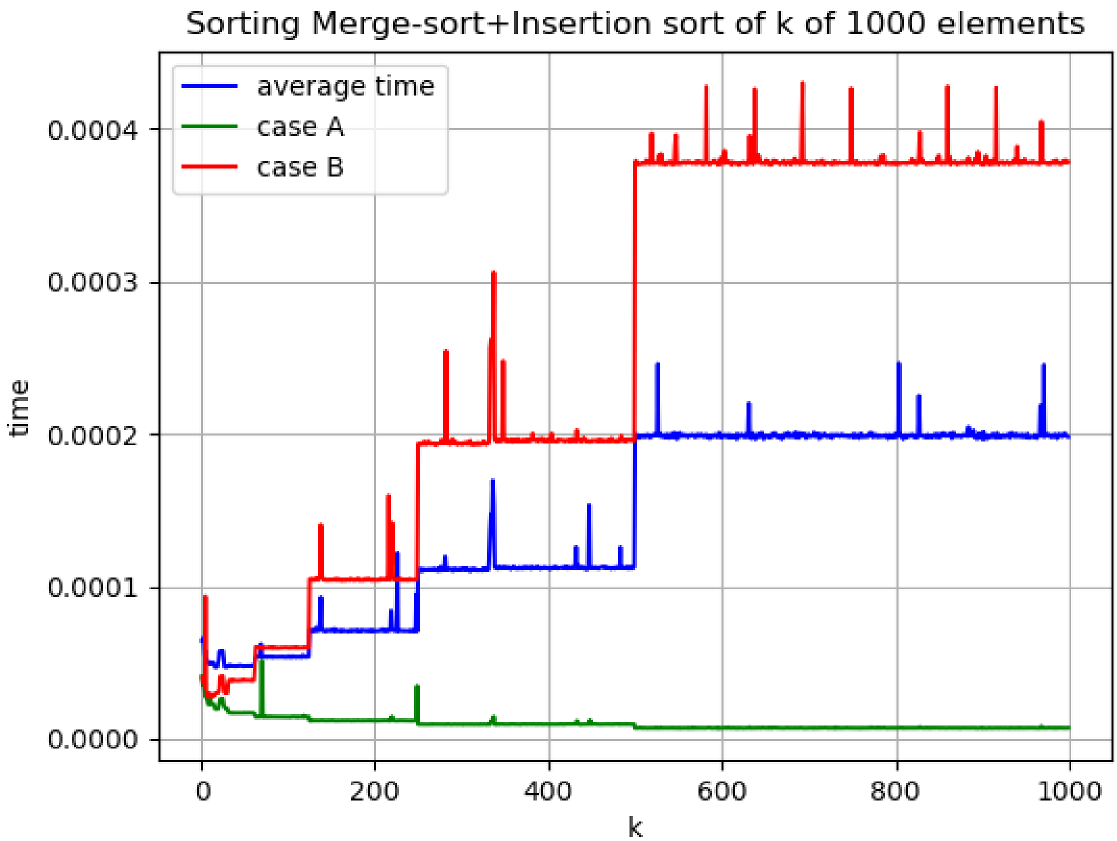
Homework 4 by Timofei Podlorytov
4.1



the graph above was created for sorting of 1000 elements and 30 sample size.
(a-b)
The algorithm for the task is in a p1.cpp file and the graph was created using the graph.py file which I also provide.

The best, worst and average cases are presented on the graph. I tested for different values of k which are shown on the x axis.
(c)
As we see here the graph is kinda like stairs and the jumps appear on k=2^(n) for the average and worst case. While the best case declines in time as we turn to using insertion sort faster and thus need to merge less lists together→ perform less operations and go to n complexity in insertion sort. When we have k=2^(n) we equally split the array and can do the merging for efficiently with the list of closer size.

The jumps happen between k=2^n and k=2^n-1 as I have proven by checking the values near those k in the python script. Currently they are commented out, but can be easily run, if needed.

(d)
We can clearly see the time increase for larger k as well as values of k=2^n-1 being good compared to the ones that come after. So, the best choice would be to choose a small k that equals 2^n-1. For instance, k=1 would work well.

4.2

Time complexity:
(a) (2 points) $T(n) = 36T(n/6) + 2n$,
(b) (2 points) $T(n) = 5T(n/3) + 17n^{1.2}$,
(c) (2 points) $T(n) = 12T(n/2) + n^2 * \lg n$,
(d) (2 points) $T(n) = 3T(n/5) + T(n/2) + 2^n$,
(e) (2 points) $T(n) = T(2n/5) + T(3n/5) + \Theta(n)$.

a)
$T(n) = 36T(n/6) + 2n$,
In this case the master theorem can be used as we have only one T(n) and all the values of a and b satisfy.

$\log_b a=\log_6 36=2 \rightarrow f(n)=O(n^{(2-1)})=O(n)$ as $f=2n \rightarrow$ first case and $T(n)= \Theta(n^{(\log_6 36)})=\Theta(n^2)$

b)
We follow a similar logic for this case.
$T(n) = 5T(n/3) + 17n^{1.2}$
Here we have $\log_b a=\log_3 5\approx 1.465..$ while $f=17n^{1.2} \in O(n^{1.2}) \in O(n^{(\log_3 5)}) \rightarrow$ first case and $T(n)= \Theta(n^{(\log_3 5)})$

c)
$T(n) = 12T(n/2) + n^2 * \lg n$
We can't apply the master method here as $n^2 \lg n$ is not a polynomial and we have to use something else.

Let's try using the tree:

n^2*lg(n)	n^2*lg(n)	^
| |||||		|
f(n/2) …… f(n/2) -12 kids	12*(n/2)^2*lg(n/2)	| height=
|		| $\log_2 n$
f(n/4) … 12^2 kids	12^2*(n/4)^2*lg(n/4)	v
….		

$T(n)=n^2*\lg(n)+12*(n/2)^2*\lg(n)-12*(n/2)^2*\lg(2)+12^2*(n/4)^2*\lg(n)-12^2*(n/4)^2*\lg(4)\ldots=$
$n^2*\lg(n)+3*(n)^2*\lg(n)-3*(n)^2*\lg(2)+3^2*(n)^2*\lg(n)-3^2*(n)^2*\lg(2)*2+$
$+3^3/(2^6)*(n)/2^6\lg(n)-3^3/(2^6)*(n)/2^6\lg(2)*3=\ldots.$

We get:
$n^2*\lg(n)(1+3+9+...)-n^2\lg(2)(1+3+9*2+27*3+3^4*4\ldots)$ in each sum we have $\log_2 n$ elements

we know that $n^2*\lg(n)$ grows faster than n^2 so we can dismiss n^2 for the upper bound. Plus the sum is positive and
Now we get:
$n^2\lg(n)*(1-3^{\log_2 n})/(1-3)$ which is $\Theta(n^2 * \lg(n)*3^{\lg(n)})$

d)
T (n) = 3T (n/5) + T (n/2) + 2^n
here the master theorem won't work as we have two separate function for T(n) inside the main one plus non polynomial as f(n).

<u>Substitution:</u>
let's try 2^n.
Base step:
T(1)=2^1=O(2^n) true
Induction step:assume true for any k<n
T(n)=3T(n/5)+T(n/2)+2^n=3*2^(n/5)+2^(n/2)+2^n = 2*(2^n)-(2^n-3*2^(n/5)-2^(n/2))=
desired >0   residual>0 for n>=3

=c*2^n-residual>0 → O(2^n) true

$\blacksquare$

Is this the tightest fit though? If mot there must exist a tighter O(f(n))

$O(n^k) \subset O(2^n)$

Base case n=1
$T(1)=2^1 \neq 1^k$ we have an issue of an exponential component in the function in its root. Thus getting anything less will be impossible.

Θ(2^n)

e)
T (n) = T (2n/5) + T (3n/5) + Θ(n).
Master theorem won't work here due to two recursive function calls of T(n) in T(n)

Let's try a recursion tree:

Θ(n)	1 child	0
| |		
Θ(2n/5) Θ(3n/5)	2 children	1
| | | |		
Θ(4n/25) Θ(6n/25) Θ(6n/25) Θ(9n/25)	4 children	2
| | | | |		
…..	…	….
	2^n children	n

but as we know Θ(const*n)=Θ(n)→ in each step the expression is multiplied by a constant inside Θ and we can simplify it in the following way

Θ(n)	1 child	0	Θ(n)
| |			
Θ(n) Θ(n)	2 children	1	2Θ(n)
| | | |			
Θ(n) Θ(n) Θ(n) Θ(n)	4 children	2	4Θ(n)
| | | | |			
…..	…	….	

2^n children n $2^n\Theta(n)$

Overall sum is $\Theta(n)+2\Theta(n)+4\Theta(n)+8\Theta(n)\ldots\ldots+2^n\,\Theta(n)=$
$=\Theta(n)(1+2+4+...2^k)=\Theta(n)$*(2^k-1)/1

the final n is the height of the tree which is $\log_{0.4}(n)$
$\Theta(n*2$^$\log_{0.4}(n))$