Analyzation of Time Complexities

The code for measurement average execution time is as follows:

total time took for all architecture additions is set to zero

total time took for all architecture removal is set to zero

for t times

get system time as start

for n times

add architecture to street

get system time as end

add difference in times to sum of time took for addition

get system time as start

for n times

remove architecture from street

get system time as end

add difference in times to sum of time took for removal

divide total time took for addition to t to get average time took

divide total time took for removal to t to get average time took

As for the addition is:

check if architecture can be built if street was empty

check for all architectures existing on the street if architecture to be added overlaps

if overlaps throw exception

add architecture to street

And removal of any architecture from any type of street is constant time. With these we can see that in first code block time complexity is $\theta(n) \cdot A(n) + \theta(n) \cdot R(n)$ where A(n) is time complexity of addition and R(n) is time complexity of removal, which is constant time. But to add a architecture system checks if it can be built on the street if it was empty, it cannot be built if it was outside of the street length for example, making this best case $\theta(1)$, but worst case is none of the existing architectures overlaps and given architecture is valid to be added, making it $\theta(n)$. So, addition time complexity is $O(n)$. Therefore, measured time in the program has a complexity of

$$\theta(n) \cdot O(n) + \theta(n) \cdot \theta(1) = O(n^2) + \theta(n) = O(n^2)$$

For every street type