P2 Analysis - ServiceQueueSlow.h

Friday, March 1, 2019 4:52 PM

① snapshot() → $\boxed{\Theta(n)}$

The time complexity of this function is linear. The use of vector:: clear function produces a linear time complexity due to its dependence on the size of the vector. The statement buzzers = the_queue also results in a linear time complexity. This is because the vector class overrides the `=' operator and performs an element-by-element copy, which is also dependent on the size.

Hence, $n + n = 2n \Rightarrow \Theta(n)$.

② length() → $\boxed{\Theta(1)}$

Using the vector::size() results in a constant time complexity.

Hence, $1 \Rightarrow \Theta(1)$

③ give_buzzer() → $\boxed{\Theta(1)}$

In the if condition, vector member function size(), back(), and pop_back() were utilized. All these member functions results in a constant time complexity.

$1(1+1) = 2 \Rightarrow \Theta(1)$

The else statement only has one constant statement, therefore its time complexity is also $\Theta(1)$.

Lastly, the vector:: push_back() function has an amortized constant time complexity ($\Theta(1)$).

Hence, $1 + 1 + 1 = 3 \Rightarrow \Theta(1)$

④ seat() → $\boxed{\Theta(n)}$

The if statement uses a vector::size() function with a constant time complexity $\Theta(1)$. However, in the else statement a vector:: erase() function. This function erases a number of elements depending on the given range. Therefore, its time complexity will be linear.

Hence, $1 + n = n \Rightarrow \Theta(n)$

⑤ kickout() → $\boxed{\Theta(n)}$

The find() function's time complexity is up to linear. Therefore, it's worst-case scenario is $O(n)$. The if statement runs on a constant time complexity. However, the else statement is dominated by the $\Theta(n)$ because of the use of vector::erase(). This function uses an iterator to move to the right to-be-deleted position.

Hence, $n + 1 + n \times 1 = 2n \Rightarrow \Theta(n)$

⑥ take_bribe() → $\boxed{\Theta(n)}$

This function has a similar runtime as kickout(). The only difference is that the else statement contains an insert() function with a time complexity of $O(n)$.

Hence, $n + 1 + (n + n) \times 1 = 3n \Rightarrow \Theta(n)$