Jeff Guo

Analysis of Slow implementation

Private functions that will be called:

slide_left() has a runtime of O(n). Because it is checking if ‘i’ is less than ‘n-1’ as n is the size of the vector. Worst-case is just ‘i’ is less than ‘n-1’. Therefore, that the loop must check linearly so O(n).

find() has a runtime of O(n). Because the for loop checks to see if ‘i’ is less than the size of the vector. Since it’s is comparing the size it is iterating therefore worst-case is that ‘i’ is less than vector size so it must iterate. Hence, O(n) runtime. As for if statement, worst-case is that vect[i] equals ‘x’ and returns ‘i’ which is just O(1). So this leads to O(n * 1) which is just O(n).

push_front() has runtime of O(n). Worst-case is that vec.size()-1 is greater than 0 therefor it will loop linearly. As for the statement inside the loop it has is constant since the loop found what ‘i’ is so all it does it assigning it which only takes O(1). So the runtime is O(n).

length():

- This function has a worst-case runtime of O(1). Because all it does is to return the size of the queue (elements) in vector.

give_buzzer(): O(n)

- if statement gives worst-case runtime of O(1). Because the complexity for .size, .back, .pop_back are all constant. So overall it’s O(1)
- else statement gives O(1) due to .size
- worst-case is push_back having to reallocation which will case an O(n) complexity.
- Therefore, the total runtime is O(1 + 1 + n) which is just O(n)

seat(): O(n^2)

- if statement gives O(1) since it is just returning -1
- worst case is when the_queue.size does not equal 0 which will invoke the else statement.
- else statement will give O(n^2). Because it will call the function slide_left which is O(n) and then push_back which is also an O(n) operation. Hence, this will be O(n*n) so it’s an O(n^2) operation.
- Therefore, the total runtime is O(1 + n * n) which is O(n^2)

kick_out(): O(n^2)

- idx is assign to find and find is an O(n) function.
- Worst-case is that if statement is not true so it will invoke the else statement.
- Else statement will call slide_left (O(n) complexity) with one of the parameters being idx which is assign to a O(n). So this is already an O(n^2). push_back is also a O(n) but since push_back has nothing to do with slide_left it is O($n^2 + n$) which is O(n^2)
- Therefore, the total runtime is O(n^2).

take_bribe(): O(n^2)

- idx is assign to find function which is O(n)
- worst case is if statement is false and invoke else statement.
- Else statement calls function slide_left (O(n)) with parameter of idx and calls push_front (O(n)). This will result O($n^2 + n$) which is O(n^2).
- Total is be O($n + n^2$) which is O(n^2)

snapshot(): O(n)

- clear removes/destroys element from the vector. Since it has to check the vector is will be linearly in size. O(n)