1. I'm using the min heap for D in this project. The main reason is, we need to always check the earliest arrival time of the process in D, and compare it with the current time to decide whether it needs to be popped from D, and inserted into Q for execution. For the time complexity, the min heap would be the best choice here.

2. I would update the comparator in Q. For now, it's only compared by the priority of the process, however, if earlier arrival time would be popped first instead of choosing arbitrarily, the comparator logic would be updated to (in pseudocode)

```
int diff = a.priority - b.priority;
if (diff == 0) {
  return a.arrivalTime - b.arrivalTime;
}
return diff
```

3. One change I can think about to improve the efficiency, is to do late deletion when we try to remove the object from the min heap Q at Line 75 in my code. Since each deletion from the minheap Q is O(n) time unless we delete the top element from the heap. Thus we can just use another hashmap to store all elements that need to be deleted from the heap, and delete them when they are popped from the heap. (We need to check whether the element is already deleted from the hashmap when it is popped from the heap)