

Homework Assignment 2

Li Kunle

Groupmates: Han Zifei, Lin Jiakai, Su Chang (Alphabetically ordered)

Exercise 1

1. If we divide the lists into k sublists instead of 2, the mergesort algorithm consists of 3 steps:

1. Divide the list of length n into k sublists;
2. Recursively sorts k subarrays of length n/k ;
3. Merge n elements.

The first step costs $O(k - 1)$ time and the 3rd step costs $O(n)$ times. So together these two steps cost $O(n)$ time.

Because the mergesort algorithm forms a tree of height $\log(n) + 1$, and for each level in the tree, the time complexity is $k(cn/k) = cn$, the total time complexity for the 2nd step is $(\log(n) + 1)(cn)$, which belongs to $O(n\log(n))$.

Combining the three steps together we can find that the total complexity is $O(n\log(n))$, since it is bigger than $O(n)$. Therefore, for a given n , the total complexity is independent of k .

Exercise 2

1. We consider each reservation as a line segment, with the left ending representing an arrival and the right ending representing a departure.

1. Sort n reservations by arrival date using mergesort, with earlier arrival listed first. If two reservations have the same arrival date, the one with the earlier departure is listed first. The time complexity is $O(n\log(n))$.
2. For all $2n$ arrivals and departures, count how many arrivals or departures happen on that date. If there are more than k arrivals/departures, then there are not enough rooms in the hotel. If we finish all counting, which at most take time $O(2n * k)$, then there are enough rooms to satisfy the demand. The time complexity remains in $O(n)$.

$\therefore O(n) < O(n\log(n))$

\therefore The total time complexity is $O(n\log(n))$.

Exercise 3

Suppose we have two stacks A and B, each with length n . When pushing n elements to stack A, we count the required time three times. The time complexity is still $O(1)$. When we want to push off the element that enters the first, we need to extract the rest $n - 1$ elements and push them to stack B. The original time complexity is $O(n)$, but since we have invested $O(1 * n) = O(n)$ time before, the amortised complexity of pulling the first element from the stack is still $O(1)$. By doing so, we implement a FIFO queue in amortised constant time.