Data Structures in Java

Shikhar Bakhda
ssb2189

Question 1

Exercise 2.1

Following is the order of the functions from fastest to slowest growth rate:

1. $\frac{2}{N}$
2. 37
3. $\sqrt{N}$
4. N
5. $N \log \log N$
6. $N \log N$
7. $N \log N^2$
8. $N (\log N)^2$
9. $N^{1.5}$
10. N^2
11. $N^2 \log N$
12. $2^{\frac{N}{2}}$
13. 2^N
14. N^3

The functions $N \log N$ and $N \log N^2$ grow at the same rate:

$$O(N \log N) = O(N \log N^2)$$
$$O(N \log N) = O(2N \log N)$$

Since 2 is a constant, the equality is true.

Question 2

Exercise 2.6

Part a

If the fine on day N was given by F(N),

$$F_N = 2^{2^{N-1}}$$

Part b

The number of days it will take will be of the order $O(\log \log N)$

Question 3

Part a

1	int sum = 0;
1 + 24 + 23	for (int i = 0; i < 23; i ++)
1 + (N + 1) + N	for (int j = 0; j < n ; j ++)
2	sum = sum + 1;

Total time $= 1 + 1 + 24 + 23 + 1 + \ N + 1 + N + 2 = \ 2N + 53$

$$T(N) = O(N)$$

Part b

1	int sum = 0;
1 + N + 1 + N	for (int i = 0; i < n ; i ++)
N + (1 + 2 + 3 + … + N) +1+(1 + 2 + 3 + … + N)	for (int k = i ; k < n ; k ++)
2	sum = sum + 1;

Total time $= 1 + 1 + N + 1 + N + N + 2 * \frac{N(N+1)}{2} + 1 + \ 2 = N^2 + 4N + 6$

$$T(N) = \ O(N^2)$$

Part c
Since in every iteration, we call the function with n = n/k, only a fraction of n is sent each time, so we have:

$$T(N) = \mathrm{O}(\log N)$$

Question 4

Exercise 2.11

a. $T(500) = \ 0.5 * \frac{500}{100} = 2.5\ ms$
b. $T(500) = \ 0.5 * \frac{500 \log 500}{100 \log 100} = 3.374$ ms
c. $T(500) = \ 0.5 * \frac{500^2}{100^2} = 12.5\ ms$
d. $T(500) = \ 0.5 * \frac{500^3}{100^3} = 62.5\ ms$

Question 5

Exercise 2.15

Since the array is sorted, we can use binary search:

1. Check if the middle element A[mid] equals i. If yes, return True
2. If the i > A[mid], then the lower boundary is shifted to mid + 1
3. If the i < A[mid], then the upper boundary is shifted to mid – 1
4. Repeat the process till the lower boundary is greater than the upper boundary. Return False.

Every iteration, we divide the array into half. Thus the runtime is O(log N).