ShanghaiTech University

CS240 Algorithm Design and Analysis

Spring 2024

Problem Set 1

Due: 23:59, March 28, 2024

1. Submit your solutions to the course Gradescope.
2. If you want to submit a handwritten version, scan it clearly.
3. Late homeworks submitted within 24 hours of the due date will be marked down 25%. Homeworks submitted more than 24 hours after the due date will not be accepted unless there is a valid reason, such as a medical or family emergency.
4. You are required to follow ShanghaiTech's academic honesty policies. You are allowed to discuss problems with other students, but you must write up your solutions by yourselves. You are not allowed to copy materials from other students or from online or published resources. Violating academic honesty can result in serious penalties.

Problem 1:

1. Analyze the time complexities of the following algorithms and explain your reasoning.

Algorithm 1

```
for i ← 1 to n by i ← 2i do
    for j ← n downto 0 by j ← j/2 do
        for k ← j to n by k ← k + 2 do
            res ← res + ij + jk
        end for
    end for
end for
```

Algorithm 2

```
for i ← n downto 0 do
    for j ← 1 to n by j ← 2j do
        for k ← 0 to j do
            res ← res + ij + jk
        end for
    end for
end for
```

2. Solve the following recurrence relations. Do not use the Master Theorem. Show all of your work.

$$T(n) = T(n-k) + 2k$$

$$T(n) = 2^k T(n/2^k) + kn$$

$$T(n) = 2^k T(n/2^k) + 2^k - 1$$

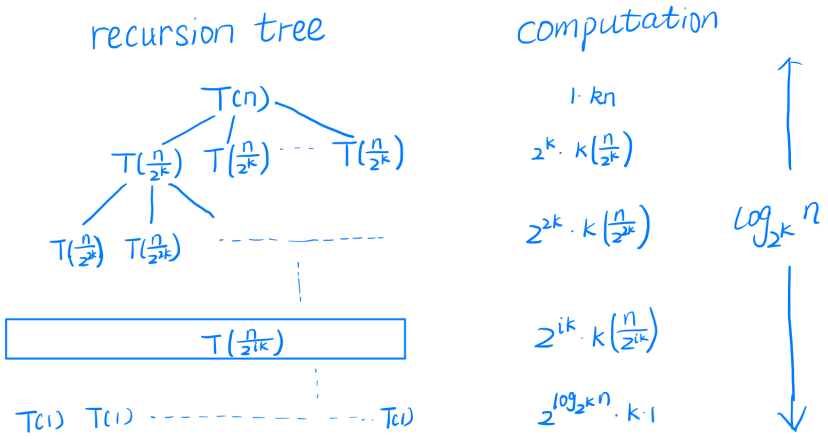

Figure 1: The recursion tree of $T(n) = 2^k T(n/2^k) + kn$

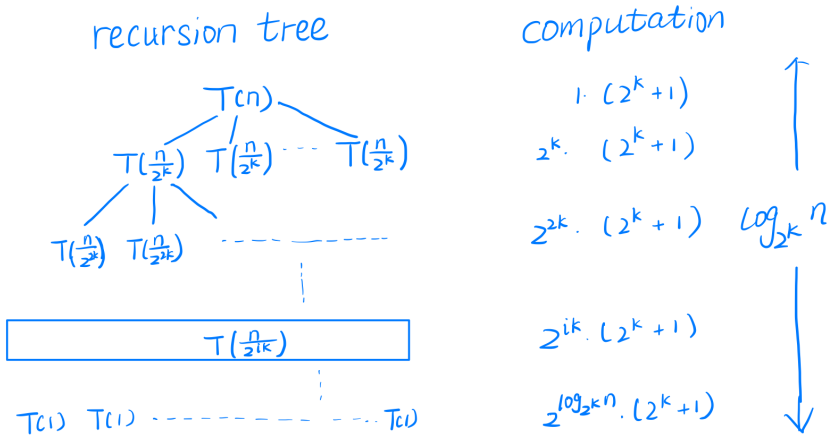

Figure 2: The recursion tree of $T(n) = 2^k T(n/2^k) + 2^k - 1$

Problem 2:

Sort the following functions in ascending order of growth.

$$f_1(n) = \log_2 n \tag{1}$$
$$f_2(n) = n^{\sqrt{n}} \tag{2}$$
$$f_3(n) = (\log_2 n)^{\log_2 n} \tag{3}$$
$$f_4(n) = n^{\frac{3}{4}} \tag{4}$$
$$f_5(n) = \log_2 \log_2 n \tag{5}$$
$$f_6(n) = 2^{\log_2 n} \tag{6}$$
$$f_7(n) = 10^{2024} \tag{7}$$
$$f_8(n) = 5^n \tag{8}$$
$$f_9(n) = \log_2 n \cdot \log_2 3^n \tag{9}$$
$$f_{10}(n) = (\log_2 n)^n \tag{10}$$

Problem 3:

Suppose you have two arrays each with n values, and all the values are unique. You want to find the median of the $2n$ values, i.e. the n'th smallest value. To do this you can make queries to either array, where a query for k to an array returns the k'th smallest value in that array. Give an algorithm which finds the median of the two arrays using $O(\log n)$ queries.

Problem 4:

Suppose there are n values in an array, and we want to sort the array using "flipping" operations. A flip takes two inputs i and j, with $1 \leq i \leq j \leq n$, and reverses the order of the values between indices i and j in the array. For example, if the array is $[1, 1, 5, 3, 4]$, then a flip with indices 2 and 5 changes the array to $[1, 4, 3, 5, 1]$. Assume a flip with inputs i and j has cost $j - i$.

(a) Assume first that all the values in the array are either 1 or 2. Design an algorithm which sorts the array using $O(n \log n)$ cost, and analyze its cost.
(Hint: mergesort)

(b) Now suppose the array contains arbitrary values. Design an algorithm which has $O(n \log^2 n)$ expected cost, and analyze its cost. Note that your algorithm is allowed to make randomized choices.
(Hint: quicksort, and the previous algorithm)

Problem 5:

Suppose n contestants will participate in a set of contests G. In each contest the contestants are split into two teams, possibly of different sizes, and such that each team contains at least one contestant. We require that for any two contestants, there is some contest in G in which these contestants are on different teams. Design the contests in G in order to minimize the total number of contests $|G|$.

Problem 6:

Suppose you have a graph with n nodes. Initially the graph contains no edges, and your task is to add a set of directed edges to the graph so that for any pair of nodes i and j where $i < j$, there is a path from i to j using at most 2 of the edges you added. You can only add edges of the form (i, j) for $i < j$. Give an algorithm to find a set of $O(n \log n)$ edges which satisfy this requirement.