CMPSC 130B Winter 2021 Homework #2

Due Feb 3, 2021, 11:59 pm

1. (10 points) Genius-1 thinks they have a solution to the prefix coding problem of k symbols into bits by examining the most frequent symbol first. Genius assigns 0 to the most frequent symbol, 10 to the next frequent symbol, .., $1^{k-1}0$ to the second least frequent symbol, and $1^{k-1}1$ to the least frequent symbol.

    a. Is this scheme a legal prefix coding?

    b. Is it optimal? If so, present a proof. If not, present a counterexample by showing that the expected cost of the encoding is higher than an optimal encoding.

2. (10 points) Genius-2 thinks they have a divide-and-conquer solution to the prefix coding problem of k symbols into bits by sorting the symbols (based on frequency), finding a subtree encoding each half, and then merging the two subtrees (defining a new root and then making the two subtrees its children).

    a. Is this scheme a legal prefix coding?

    b. Is it optimal? If so, present a proof. If not, present a counterexample by showing that the expected cost of the encoding is higher than an optimal encoding.

3. (10 points) Your friends develop four different solutions to a problem using a divide & conquer strategy. The running times of the solutions are coded as T_1, T_2, T_3, T_4. For each solution, the base case (n=1) takes a constant time. The recurrences for the four schemes are as follows:

    a. $T_1(n) = T_1\left(\frac{n}{2}\right) + 1$
    b. $T_2(n) = T_2\left(\frac{n}{4}\right) + \log log\ n$
    c. $T_3(n) = 2T_3\left(\frac{n}{2}\right) + n$
    d. $T_4(n) = 3T_4\left(\frac{n}{4}\right) + n$

    Evaluate the closed forms of these recurrences and rank the four solutions from the most efficient to the least efficient.

4. (10 points) Consider problem 2 from the previous homework. Find an algorithm that runs is O(n log n) time. Argue correctness as well as the time complexity. Hint: This problem is not about divide and conquer but about union find.

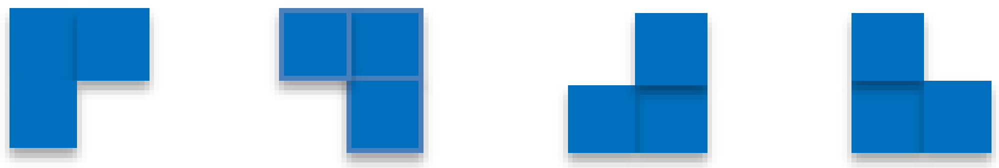
    You are given a set of n jobs, each of which takes one unit of time to complete. Each job i has an associated "latest completion time" t_i and finishing the job by then earns a reward of d_i dollars (nothing otherwise). Assume completion time >= 0 and that the rewards are distinct. It is possible for two jobs to have the same "latest completion time."

    Example input:

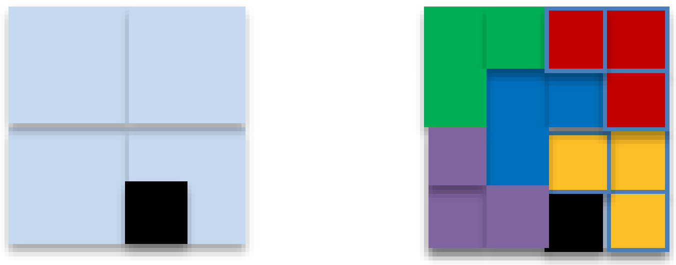
Job	Latest completion time	Reward

1	5	40
2	5	50
3	4	60
4	4	30
5	3	15
6	3	5

5. (10 points) In a square grid of *side length n* (assume a power of two), one unit square is blocked (represented by coloring it black). Your task is to cover the remaining $n^2 - 1$ squares with L-shaped tiles consisting of three squares that can be rotated by any multiple of 90 deg, as shown below.



The tiles may not overlap each other, nor cover anything outside the grid. A valid tiling for n = 4 is shown below.



(a) Given an input grid of sides n (assume a power of two) and a blocked tile position, develop a *divide and conquer* algorithm to tile it. Argue the correctness.

(b) Derive the time complexity of your solution using the Master theorem. Remember that n is the length of each side.

6. (20 points) Suppose you are given an undirected graph of n nodes where edges define cliques of possibly different sizes (graph is a union of cliques). You can test if there exists an edge between any two nodes. Devise a divide and conquer algorithm that uses O(n log n) tests to determine if there exists a clique of more than n/2 nodes (call such a clique a “dominant” clique). Now, find an O(n) algorithm for the same problem. Argue the correctness as well as the complexity of your solutions.

7. (20 points) Given a set P of n points in the 2D plane, where each p_i from P is a 2-tuple, i.e., $p_i = (x_i, y_i)$ representing the x and y coordinate of the i-th point. Find a triplet of points p_i, p_j, p_k such that the sum of distances = dist[p_i, p_j] + dist[p_j, p_k] + dist[p_i, p_k]) is minimized where dist is the Euclidean distance between the points given by dist[p_i,p_j] = sqrt($(x_i$-$x_j)^2$ + $(y_i$-$y_j)^2$). Briefly discuss the correctness and running time of your algorithm.
   Note: For full credit, give an algorithm with running time O(n log n). Algorithms with sub-optimal running time will receive partial credit, in particular, algorithms with running time $O(n^3)$ will only fetch a small portion of the total points.