IIIT Vadodara
CS203:Assignment

December 20, 2021

1 Problem

Arrange the following functions in increasing order of their complexities:
(a) $1.000001^n, 3^{10000000}, n^{\sqrt{n}}, n^5.3^{n/2}$
(b) $n, \sqrt{n}, n^{1.5}, n^2, nlogn, nloglogn, nlog^2n, nlog(n^2), 2/n, 2^n, 2^{n/2}, 37, n^2logn, n^3$

2 Problem

In each of the following situations, indicate whether $f = O(g)$ or $f = \Omega(g)$ or both (in which case $f = \theta(g)$)
(a)$f(n) = n - 100$, $g(n) = n - 200$
(b) $f(n) = 100n + logn$, $g(n) = n + (logn)^2$
(c) $f(n) = n^{1.5}$, $g(n) = nlog^2n$
(d) $f(n) = log2n$, $g(n) = log3n$
(e) $f(n) = n!$, $g(n) = 2^n$

3 Problem

Solve the following recurrences and give upper bounds.
(a)$4T(n/2) + n^2logn$
(b)$8T(n/2) + nlogn$
(c)$2T(\sqrt{n}) + n$

4 Problem

Given a graph G, check whether can we color the vertices of a graph G using two colors such that no two adjacent vertices have the same color. Which algorithm will you apply for this problem and what would be the complexity?

5 Problem

Find the minimum number of moves for Tower of Hanoi problems:
(a)There are four towers, numbered from 0 to 3. A stack of n white disks, is initially on tower 0, and a stack of n black disks, is initially on tower 2. The destination towers for the white stack and the black stack are towers 2 and 0, respectively. In other words, the goal is to interchange the two stacks, using towers 1 and 3 as temporary holding towers.
(b) If we duplicate every disk to create a stack of 2n disks with two of each size and considering the 3 tower from 0 to 2. Initially, stack of 2n disks is on tower 0. Transfer the 2n disks from tower 0 to tower 2 using tower 1.

6 Problem

Suppose you are given an array $A[1...n]$ of distinct sorted integers that has been circularly shifted k positions to the right. For example, [31, 40, 4, 11, 24, 28] is a sorted array that has been circularly shifted k = 2 positions, while [24, 28, 31, 40, 4, 11] has been shifted $k = 4$ positions. We can obviously find the largest element in A in $O(n)$ time. Give an $O(logn)$ algorithm to find the largest number in A.

7 Problem

Given two prime numbers, P1 and P2 , your task is to reach from P1 to P2 by changing only one digit at a time and also the number you get after changing a digit must also be a prime. For example suppose that P1 is 1033 and P2 is 8179 (both primes), we can reach from P1 to P2 in following way :-
$1033 -> 1733 -> 3733 -> 3739 -> 3779 -> 8779 -> 8179$.

The first digit must be non-zero that is you can not go from 4-digit to 3-digit numbers. Note that there may be multiple way to get to destination but you have to find the way in minimum number of steps.

For example the minimum number of steps to get from P1 to P2 in following examples are as follows :-
1033 8179 ANWSER:- 6
1373 8017 ANWSER:- 7
1033 1033 ANWSER:- 0

Hint: Apply BFS

8 Problem

Suppose you are managing a collection of processors and must schedule a sequence of jobs over time. The jobs have the following characteristics. Each job j has an arrival time a_j when it is first available for processing, a length j which indicates how much processing time it needs, and a deadline d_j by which it must be finished. (Assume $0 < j <= d_j - a_j$.) Each job can be run on any of the processors, but only on one at a time; it can also be preempted and resumed from where it left off (possibly after a delay) on another processor. Moreover, the collection of processors is not entirely static either: You have an overall pool of k possible processors; but for each processor i , there is an interval of time $[t_{i1}, t_{i2}]$ during which it is available; it is unavailable at all other times. Given all this data about job requirements and processor availability, decide whether the jobs can all be completed or not.

Give a polynomial-time algorithm that either produces a schedule completing all jobs by their deadlines or reports (correctly) that no such schedule exists. You may assume that all the parameters associated with the problem are integers.

Example. Suppose we have two jobs J_1 and J_2 . J_1 arrives at time 0, is due at time 4, and has length 3. J_2 arrives at time 1, is due at time 3, and has length 2 . We also have two processors P_1 and P_2. P_1 is available between times 0 and 4 ; P_2 is available between times 2 and 3. In this case, there is a schedule that gets both jobs done.

- At time 0 , we start job J_1 on processor P_1 .
- At time 1 , we preempt J_1 to start J_2 on P_1 .
- At time 2 , we resume J_1 on P_2 . (J_2 continues processing on P_1 .)
- At time 3 , J_2 completes by its deadline. P_2 ceases to be available, so we move J_1 back to P_1 to finish its remaining one unit of processing there.
- At time 4 , J_1 completes its processing on P_1.

Notice that there is no solution that does not involve preemption and moving of jobs.
Hint: Apply Greedy