

Preliminary Examination

Samples

Discrete Structures and Algorithms

Problem #1

Suppose that you live in Manhattan near subway station A. You have two friends: one who lives in the Bronx and another one who lives in Brooklyn. You like your friends equally well and prefer to spend equal time with them. When you go to Brooklyn, you catch a train that comes from Grand Central Station. When you go to Bronx you catch a train that goes to Grand Central Station. Since you like your friends equally well, you take the train that arrives first. You arrive at subway station A every Saturday at a random time. Trains arrive every 10 minutes in each direction. Nevertheless, after ten Saturdays you discover that you spent 9 Saturdays with your friend in Brooklyn and only one Saturday with your friend in the Bronx. Explain how that has happened.

Problem #2

Consider the towers of Hanoi problem. Suppose that you have three towers A, B and C. Suppose that tower A has n disks stacked on it in decreasing order of disk diameters. In one move you may take a disk from one tower and put it on one of the other towers. Prove that it requires $2^n - 1$ steps to transfer the disks from tower A to tower C in such a way that at no time is a disk of smaller diameter covered by a disk of larger diameter.

Problem #3

How many ways are there to assign 25 students to 5 faculty advisors so that every faculty has at least one advisee?

Problem #4

We call a graph $G = \langle V, E \rangle$ **bipartite** if and only if V can be decomposed into two subsets $V = V_1 \cup V_2$ so that the intersection of V_1 and V_2 is empty and, for every edge $e = (v_1, v_2)$, v_1 belongs to V_1 and v_2 belongs to V_2 . Design an algorithm that, given a graph, decides whether the graph is bipartite

ALGORITHMS

Problem #5:

Let $T[1..n]$ be a sorted array of distinct integers, some of which may be negative. Give an algorithm that can find an index i such that $1 \leq i \leq n$ and $T[i] = i$, provided such an index exists. Your algorithm should take a time in $O(\log n)$ in the worst case.

Problem #6:

Suppose that in the 0-1 knapsack problem, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give an efficient algorithm to find an optimal solution to this variant of the knapsack problem, and argue that your algorithm is correct.

Problem #7:

Suppose an oracle has given you a magic computer, C , that when given any Boolean formula B in CNF will tell you in one step if B is satisfiable or not. Show how to use C to construct an actual assignment of satisfying Boolean values to the variables in any satisfiable formula B . How many calls do you need to make to C in the worst case in order to do this?

Problem-#8

Show an algorithm to solve the Tower of Hanoi problem. (ii) Show that it can be performed by $2^N - 1$ moves using a recursive algorithm. (iii) Professor Shonku thinks it is not possible to improve the recursive solution's efficiency by re-using sub-solutions. Try such a solution and argue in favor or against Prof. Shonkul.

Hint: Tower of Hanoi problem: There are N disks will all different sizes, and three pegs each can hold them in a stack. The restriction is that a smaller disk must not be below a larger disk. Initially, all disks are stacked accordingly (largest one at the bottom smaller one on top) on one peg. Monks have to move all of them in a second peg without violating the restriction. Monk can use the third peg as a temporary stacking peg.