DESIGN AND ANALYSIS OF ALGORITHMS

Year:II-II(2023 batch) **Banch:CSE-D**

MODULE-1 MODULAR QUESTION BANK

1. A tromino is a group of three unit squares arranged in an L-shape. Consider the following tiling problem: The input is an m × m array of unit squares where m is a positive power of 2,with one forbidden square on the array. The output is a tiling of the array that satisfies the following conditions:

    - Every unit square other than the input square is covered by a tromino.
    - No tromino covers the input square.
    - No two trominos overlap.
    - No tromino extends beyond the board.

    Design a divide-and-conquer algorithm that solves this problem and also find the time complexity for the algorithm. And Analyze your algorithm and give the results using order notation.

2. Design a divide-and-conquer algorithm for the Towers of Hanoi problem. The Towers of Hanoi problem consists of three pegs and n disks of different sizes. The object is to move the disks that are stacked, in decreasing order of their size, on one of the three pegs to a new peg using the third one as a temporary peg.
   The problem should be solved according to the following rules:
   (1) When a disk is moved, it must be placed on one of the three pegs;
   (2) Only one disk may be moved at a time, and it must be the top disk on one of the pegs;
   (3) A larger disk may never be placed on top of a smaller disk.
   Show for your algorithm that S (n) = 2n − 1. (Here S (n) denotes the number of steps (moves), given an input of n disks.)
   (4) Analyze your algorithm and give the results using order notation.

3. When the list gets large, quicksort is clearly the fastest *comparison* sorting algorithm (hence the name). However, when the lists are small enough, quicksort runs slower that some of the $\Theta(n^2)$ algorithms. This might not seem important until you note that when sorting a large list with quicksort, many *many* small sublists must be sorted. While the savings on sorting one small list with a faster algorithm is negligible, sorting hundreds of small lists with a faster algorithm can make a difference in the overall efficiency of the sort. For this, you will combine quicksort with another sorting algorithm to build the fastest possible sorting algorithm. You have several options --
   Use quicksort until the list gets small enough, and then use another sort or insertion sort to sort the small lists

- Use quicksort to "mostly" sort the list. That is, use quicksort to sort the list until a cut-off size is reached, and then stop. The list will now be mostly sorted, and you can use insertion sort on the entire list to quickly complete the sorting (not unlike the strategy used in Shell Sort)
- Use some other method of your own devising.

What does ``small enough'' mean? You can try a percentage of the list (say, 5% or 10%), or an absolute number (8 elements, 10 elements, 15 elements, 100 elements, etc), or something else of your choosing. You should also test to ensure that you have the most efficient algorithm possible. You should also be sure that your hybrid quicksort has reasonable performance on all lists -- most notably, it should be efficient on sorted and inverse sorted lists as well as random lists. Try various methods for choosing the pivot element, to try to get the best possible behavior.

4. **Bridge Crossing Puzzle:** There are four people who want to cross a rickety bridge; they all begin on the same side. You have 17 minutes to get them all across to the other side. It is night, and they have one flashlight. A maximum of two people can cross the bridge at one time. Any party that crosses, either one or two people, must have the flashlight with them. The flashlight must be walked back and forth; it cannot be thrown, for example. Person 1 takes 1 minute to cross the bridge, person 2takes 2 minutes, person 3 takes 5 minutes, and person 4 takes 10 minutes. A pair must walk together at the rate of the slower person's pace.

   Consider the generalization of the bridge crossing in which we have n > 1 people whose bridge crossing times are t1, t2,...,tn. All the other conditions of the problem remain the same:at most two people at a time can cross the bridge (and they move with the speed of the slower of the two) and they must carry with them the only flashlight the group has.

   a. Design a greedy algorithm for this problem and find how long it will take to cross the bridge by using this algorithm.
   b. Does your algorithm yield a minimum crossing time for every instance of the problem?
   c. If it does—prove it;
   d. if it does not—find an instance with the smallest number of people for which this happens.
   e. Analyze your algorithm and give the results using order notation.

5. A telecom company (FCC) has a huge pile of requests from radio stations in the India to transmit on radio frequency 88.1 FM. The FCC is happy to grant all the requests, provided that no two of the requesting locations are within Euclidean distance 1 of each other (distance 1 might mean, say, 20 miles). However, if any are within distance 1, reject the entire set of requests. Suppose that each request for frequency 88.1 FM consists of some identifying information plus (x, y) coordinates of the station location. Assume that no two requests have the same x coordinate, and likewise no two have the same y coordinate. The input includes two sorted lists, Lx of the requests sorted by x coordinate and Ly of the requests sorted by y coordinate.

a) Suppose that the map is divided into a square grid, where each square has dimensions ½X 1/2 Why must the FCC reject the set of requests if two requests are in, or on the boundary of, the same square?
b) Design an efficient algorithm for the FCC to determine whether the pile of requests contains two that are within Euclidean distance 1 of each other; if so, the algorithm should also return an example pair. For full credit, your algorithm should run in O (n log n) time, where n is the number of requests. Hint: Use divide-and- conquer, and use Part (a).
c) Describe how to modify your solution for Part (b) to determine whether there are three requests, all within distance 1 of each other. For full credit, your algorithm should run in O (n log n) time, where n is the number of requests.

6. Consider the problem of scheduling n jobs of known durationst1, t2,...,tn for execution by a single processor. The jobs can be executed in any order, one job at a time.

a) Design an algorithm to find a schedule that minimizes the total time spent by all the jobs in the system. (The time spent by one job in the system is the sum of the time spent by this job in waiting plus the time spent on its execution.)
b) Design a greedy algorithm for this problem.
c) Does the greedy algorithm always yield an optimal solution?
d) Analyze your algorithm and give the results using order notation.

7. A thief breaks into a store and wants to fill his knapsack of capacity K with goods of as much value as possible. The thief's knapsack can hold 100 gms and has to choose from:
30 gms of gold dust at Rs 1000 /gm
60 gms of silver dust at Rs 500/gm
30 gms of platinum dust at Rs 1500/gm
20 gms of copper dust at Rs 400/gm
25 gms of bronze dust at 250/gm
a) Design a greedy algorithm that finds the optimal solution for the given problem.
b) what is the maximum possible solution to get more profit?
c) Analyze the algorithm by finding time and space complexity for n number of items

8. An independent set of an undirected graph G = (V, E) is a subset $W \subseteq V$ such that there does not exist any edge $e = \{u, v\} \in £$ with both end points in W (that is, $|e \cap W| \leq 1$). An undirected graph G = (V, E) is called an interval graph if every vertex $v \in V$ can be associated with some interval $I_v = [a, b)$ in R such that there is an edge e between u and v if and only if $I_u \cap I_v = \emptyset$. Let G be an interval graph and $\{I_v : v \in V\}$ be the intervals associated with its vertices.

From the Independent set of undirected graph compute a maximum weighted independent set of a binary tree. Your algorithm will be given as input a pointer to the root of a binary tree. Each node v has a weight w(v) and two pointers left(v) and right(v) leading to the left and right children of v respectively; if a child is absent, the corresponding pointer is NULL. The weight of an independent set in a tree is the sum of

the weights of the nodes in it. A maximum weight independent set is an independent set whose weight is maximum.

a) Derive a recursive relation for the weights of a maximum weight independent set of various subtrees.

b) Design an efficient algorithm to compute the weight of a maximum weight independent set of the input tree and Analyze your algorithm and give the results using order notation.

9. Design an algorithm that multiplies two 8 × 8 matrices using Strassen's Matrix Algorithm. Determine the following cases:

   a) number of addition operations
   b) number of Multiplications operations
   c) number of Subtractions operations
   d) total number of arithmetic operations

   Analyze your algorithm with normal multiplication for n=8 and give the results using order notation.

10. Consider an undirected graph G = (V, E) with nonnegative edge weights We ≥ 0. Suppose that you have computed a minimum spanning tree of G, and that you have also computed shortest paths to all nodes from a particular node s ∈ V.
    Now suppose each edge weight is increased by 1: the new weights are $W'_e = We + 1$

    a) Design an algorithm to calculate the cost of minimum spanning tree?
    b) Determine the minimum number of spanning trees?
    c) Suppose Prisms algorithm is run on this graph. In what order are the edges added to the MST?
    d) Compute the time complexity of proposed algorithm.