Design & Analysis of Algorithm
Home Assignment-I

Prof. Bibhudatta Sahoo

January 2021

All algorithms are to be written in pseudocode notation. Use a long exercise book for the hard copy submission of the home assignment. The Last date of submission of the assignment is 30 January 2021

Problem 1: Suppose that, in a divide-and-conquer algorithm, we always divide an instance of size n of a problem into 10 sub-instances of size n/3, and the dividing and combining steps takes a time $\Theta(n^2$. Write a recurrence equation for the running time $T(n)$, and solve the equation for T(n).

Problem 2: What is the asymptotic running time of quickselect, using a median-of-median-of-three partitioning strategy?

Problem 3: Show that quickselect with median-of-median-of-seven partitioning is linear. Why is median-of-median-of-seven not used in the proof?

Problem 4: Show how to multiply two complex numbers $x = a + ib$ and $y = c + id$ using only three multiplications.

Problem 5: Consider a set of n intervals $[a_i, b_i]$ that cover the unit interval, that is,$[0, 1]$ is contained in the union of the intervals. (a) Describe an algorithm that computes a minimum subset of the intervals that also covers $[0, 1]$. (b) Analyze the running time of your algorithm.

Problem 6: Design a linear-time algorithm to rearrange elements of a given array of n integers so that all its negative elements precede all its positive elements. The algorithm should not use more than constant extra memory, i.e. it should not use an additional array. Implement your algorithm.

Problem 7: You are facing a wall that stretches infinitely in both directions. There is a door in the wall, but you know neither how far away nor in which direction. You can see the door only when you are right next to it. Design an algorithm that enables you to reach the door by walking at most $O(n)$ steps where n is the (unknown to you) number of steps between your initial position and the door.

Problem 8: A very large department has a mix of 100 professors: some are honest and hard-working, while others are deceitful and do not like students. The honest professors always tell the truth, but the deceitful ones sometimes tell the truth and sometimes lie. You can ask any professor the following question about any other professor: professor Y, is professor X honest? Professor Y will answer either with yes or no. Design an algorithm that, with no more than 198 questions, would allow you to figure out which of the 100 professors are honest. It is known that there are more honest than dishonest professors.

Problem 9: Design a reasonably efficient algorithm for solving the following problem and determine its efficiency class. You are given n telephone bills and m checks sent to pay the bills ($n = m$). Assuming that telephone numbers are written on the checks, find out who failed to pay. (For simplicity, you may also assume that only one check is written for a particular bill and that it covers the bill in full.)

Problem 10: There are **n** black, **m** green, and **k** brown chameleons on a deserted island. When two chameleons of different colors meet they both change their color to the third one (e.g., black and green chameleons become brown). Design and implement an algorithm that for each choice of n, m, and k will decide whether it is possible that after some time all chameleons on the island are the same color. If you think that it is always possible, check the case $n = 1$,$m = 3$, and $k = 5$.

Problem 11: Sometimes, there are many shortest paths between nodes (i.e. multiple paths that are at least as short as any other path). Show how to compute the number of shortest paths from a specific node v to each other node. Your algorithm should have the same running time as Dijkstras algorithm.

Problem 12: Consider a possible DIVIDE-AND-CONQUER approach to

finding a minimal spanning tree in a connected, weighted, graph G. Suppose that we recursively divide the vertices of G into two disjoint subsets V_1 and V_2. We then find a minimal spanning tree T_1 for V_1 and a minimal spanning tree T_2 for V_2. Finally we find a minimum weight edge e connecting T_1 and T_2. We then let T be the graph obtained by combining T_1, T_2, and e.

(a) Is T always a spanning tree?
(b) If T is a spanning tree, is it always a minimal spanning tree?

Problem 13: Considering a 0/1 Knapsack problem, explain the application of Dynamic Programming, Backtracking and Greedy algorithm to found an optimal solution. Give the three separate representation of problem space and compare the time complexity of the algorithms under the said paradigm.

Problem 14: Given a sorted array of distinct integers $A[1...n]$, you want to find out whether there is an index i for which $A[i] = i$. Given a dive-and-conquer algorithm that runs in time *O(log n)*.

Problem 15: You are interested in analysing some hard-to-obtain data from two separate databases. Each database contains n numerical values, so there *2n* values total-and you may assume that no two values are the same. You'd like to determine the median of this set of *2n* values, which we will define here to be the n^{th} smallest value. However, the only way you can access these values is through quarries to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the k^{th} smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible. Give an algorithm that finds the median value using at most *O(log n)* queries.

Problem 16: Assume you are working in the census department of a small town where the number of records, about 3000, is small enough to fit into the internal memory of a computer. All the people currently living in this town were born in the United States. There is one record for each person in this town. Each record contains (a) the state in which the person was born, (b) county of birth, and (c) name of person. How would you produce a list of all persons living in this town? The list is to be ordered by state. Within each state the persons are to be listed by their counties, the counties being arranged in alphabetical order. Within each county, the names are also listed in alphabetical order. Justify any assumptions you make.

Problem 17: Let T be a tree with root v. The edges of T are undirected.

Each edge in T has non-negative length. Write an algorithm to determine the length of the shortest paths from $v$to the remaining vertices of T. Your algorithm should have complexity $O(n)$, where n is the number of vertices in T.

Problem 18: Let G be a directed, acyclic graph with n vertices. Assume that the vertices are numbered 0 through $n-1$ such that all edges are of the form $< i, j >$. where $i < j$. Assume that the graph is available as a set of adjacency lists and that each edge has a length (which may be negative) associated with it. Write an algorithm to determine the length of the shortest paths from vertex 0 to the remaining vertices. The complexity of your algorithm should be $O(n+e)$, where e is the number of edges in the graph.

Problem 19: Let's consider a long, quiet country road with houses scattered very sparsely along it. (We can picture the road as a long line segment,, with an eastern endpoint and a western endpoint.) Further, let's suppose that despite the bucolic setting, the residents of all these houses are avid cell phone users. You want, to place cell phone base stations at certain points along the road, so that every house is within four miles of one of the base stations. Give an efficient algorithm that achieves this goal, using as few base stations as possible.

Problem 20: Bucket sort assumes that keys are randomly distributed over some interval say $[a, b]$. The interval $[a, b]$ is divided into k equal subintervals, called buckets. (For example, if we were sorting names, the buckets might correspond to the first letter of the last name.) Bucket sort begins by putting each element in the array to sort in the appropriate bucket. It then sorts each bucket using an appropriate sorting algorithm. Show that at this point, if we enumerate the items in the first bucket (in sorted order) followed by the items in the second bucket (again, in sorted order) and so on, the entire array is sorted. Making appropriate assumptions (what are they?), what is the expected run-time cost of your algorithm?

Problem 21: Consider distinct items $x_1, x_2, ..., x_n$ with positive weights $w_1, w_2, ..., w_n$ such that $\sum_{i=1}^{n} w_i = 1.0$. The *weighted median* is the item x_k that satisfies $\sum_{x_i<x_k} w_i < 0.5$ and $\sum_{x_j>x_k} w_j \leq 0.5$. (a) Show how to compute the weighted median of n items in worst-case time $\bigcirc(nlogn)$ using sorting. (b) Show how to compute the weighted median in worst-case time $\bigcirc(n)$ using a linear-time median algorithm.

Problem 22: How would you implement **mergesort** without using recursion? Determine the running time of **mergsort** for, *(i)* sorted input, *(ii)*

reverse-order input, and *(iii)* random input.

Problem 23: Using the **quicksort** implementation discussed in the class, determine the running time of **quicksort** for, *(i)* sorted input, *(ii)* reverse-order input, *(iii)* random input, and *(iv)*when all keys are equal.

Problem 24: Suppose you are given a sorted list of n elements followed by $f(n)$ randomly ordered elements. How would you sort the list if, *(i)* $f(n) = \bigcirc(1)$, *(ii)*$f(n) = \bigcirc(log\mathbf{n})$,and *(iii)* $f(n) = \bigcirc(\sqrt{n})$?

Problem 25: Develop a non-deterministic algorithm of complexity $f(n) = \bigcirc(n)$ to determine whether there is a subset of the n numbers x_i, $1 \leq i \leq n$ that sums to W.

Problem 26: Show that *partition problem* reduces to *0/1 knapsack decision problem* ?

Problem 27: Demonstrate empirically how randomized **quicksort** compares with the deterministic **quicksort**.

Problem 28: Give a Monte-Carlo algorithm for finding the majority elements in an array?

Problem 29: Develop an algorithm that finds the k^{th} smallest element of a sequence that is presented to you one element at a time in an order you cannot control. You have only space $O(k)$ available. This models a situation where voluminous data arrives over a network or at a sensor. *(a)*Refine your algorithm so that it achieves a running time $O(nlogk)$. *(b)* Refine the algorithm and its analysis further so that your algorithm runs in average-case time $O(n)$ if $k = O(n/logn)$. Here, **average** means that all orders of the elements in the input sequence are equally likely.

Problem 30: Explain how to insert k new elements into a sorted list of size n in time $\bigcirc(k\mathbf{log}\ k + n)$.
