Assignment I

Design and Analysis of Algorithms

1. Let f be a monotonically decreasing function with positive integer valued input and integer valued output. (Thus $f(1) > f(2) > f(3).....$) You are given a "black box" function $int\ ComputeF(int)$ that returns the value of $f(i)$ if a positive integer i is given as the input. We want the **smallest** (positive) integer value n such that $f(n) < 0$. Design an algorithm to find the value of n with only $O(\log n)$ calls to the function $ComputeF$.

2. Suppose a, b are two integers, we say a positive integer d is a greatest common divisor, $GCD(a, b)$ if $d|a, d|b$ and for any other $e|a, e|b$, it is also true that $e|d$. Let $a > b$. Show that $GCD(a, b) = GCD(a-xb, b)$ for any integer x. Clearly, if $b|a$, $GCD(a, b) = b$. Using these observations, choosing appropriate value for x, show that $GCD(a, b)$ can be computed with at most $2\log_2 a$ recursive calls. (Hint: Show that $a - xb$ can be made always less than $\frac{a}{2}$ by appropriate choice of x.)

3. Let G be a graph. Fix any vertex v. Consider the following:

   $MST(G, v)$
   . Pick the nearest neighbour w of v // break ties arbitrarily
   . $Print(v, w)$
   . $G' = Contract(G, (v, w))$ // Let the contracted vertex be called x
   . $MST(G', x)$

   Prove that the algorithm correctly computes a minimum spanning tree in the graph G. As was done in the case of Kruskal's algorithm, eliminate the contraction step and show how the algorithm can be implemented in $O((n + e)\log n)$ complexity.

4. A shop offers you a special Onam gift. You can buy items in the shop for reduced prices. Let us assume that the items listed for profit purchase are $1, 2, 3, ..., n$. purchase of item i offers you a profit p_i. But item i has weight w_i. Though you have lot of money with you to spend, your car can carry only W Kg. Design an $O(nW)$ algorithm that allows you to decide on which items to purchase so as to maximize your profit. (Items in the shop can't be taken fractionally - as is the case with washing machines or refrigerators) Give an $O(n \log n)$ greedy strategy for solving the problem if items can be taken fractionally (as is the case with items like rice and sugar) [Hint: Consider profit/weight ratio]

5. In the class, we studied the problems of finding the maximum independent set in an interval graph as well as optimal colouring of an interval graph. Here is a recursive algorithm design for finding the maximum independent set in a general graph. Given graph $G(V, E)$, let $V = \{v_1, v_2, .., v_n\}$. suppose we pick the v_1 in our optimal independent set, then we can't pick any of its neighbours. and we must look for a maximum independent set in reduced graph with v_1 and all its neighbours removed. On the other hand, if we do not pick v_1, Then, the problem reduces to finding the maximum independent set $G \setminus v_1$. Using these observations, design a divide and conquer algorithm for solving the problem. Show that your algorithm needs exponential running time.

6. This question attempts to find an exponential time algorithm for finding the minimum number of colours needed for a proper colouring of a graph G (Chromatic number). Suppose G has n vertices. If G is a complete graph, clearly n colours are required. Otherwise, let u, v be two non-adjacent vertices in G. Let G_1 be the graph obtained by adding the edge (u, v) to G and G_2 be the graph obtained by contracting the edge (u, v) in G_1. Argue that the chromatic number of G is the minimum of the chromatic number of G_1 and G_2. Design a recursive algorithm based on this observation and prove that the algorithm terminates and outputs the correct answer. Show that the running time of the algorithm is exponential.

7. Suppose we want to construct a binary search tree of keys $\{1, 2, ...n\}$. We know in advance that key i will be searched f_i times. Since several binary search trees are possible with keys $\{1, 2, ..n\}$, we would like to find one that minimizes the search cost. The cost of searching an item i in a BST T is

$f_i.d_i(T)$ where $d_i(T)$ denotes the depth of the item i in the tree T. (For example, if $n = 3$, Let $f_1 = 1, f_2 = 2, f_3 = 3$, then if the tree has 2 at the root, the cost will be $1 \times 2 + 2 \times 1 + 3 \times 2$.). To find the optimal solution, suppose element i is at the root, and if $C(1, i-1)$ is the cost of the optimal left subtree containing 1 to i and $C(i+1, n)$ is the cost of the optimal right subtree with items $i+1$ to n, then, show that the cost of the tree will be $C(1, i-1) + C(i+1, n) + \sum_{j=1}^{n} f_j$. Develop a recursive solution for the problem and use dynamic programming to reduce the complexity to $O(n^3)$.

8. Suppose we are given a set of airports $\{1, 2, ..n\}$. The fare for a direct flight from i to j is given by $c(i, j)$. We would like to get a flight sequence that optimizes the travel cost from city i to city j for all values of i, j (possibly by taking several hops instead of taking a direct flight). To achieve this, consider the following strategy. Let $C_k(i, j)$ denote the optimal cost to fly from i to j without landing on any air port numbered greater than k. (Example, $C_2(4, 5)$ computes the best way to travel from 4 to 5 where you may chose to take hops to either 1 or 2 as intermediate steps, but not to any air port with number exceeding 2.) Clearly $C_0(i, j) = c(i, j)$ as direct flight is the only option permitted here. Moreover $C_n(i, j)$ is nothing but the optimal cost which we want to compute. The following outlines a way to compute $C_{k+1}(i, j)$ if we know the value of $C_k(r, s)$ for all r, s.

   - Show that $C_{k+1}(i, j) = \min\{C_k(i, j), C_k(i, k+1) + C_k(k+1, j)\}$
   - write a recursive program to compute $C_n(i, j)$.
   - Using an array to store $C_k(i, j)$ values, design a dynamic programming strategy to compute $C_{k+1}(i, j)$. Iterating over k show that we can solve the problem in $O(n^3)$ time. (Note: The array used for storing $C_k(i, j)$ values can be re-used to store $C_{k+1}(i, j)$ in the next iteration over k).

9. In the airport question posed above, suppose we change our objective to start from airport 1, visit every airport exactly once and come back to vertex 1, Let us see what can be done. A straightforward method would be to generate all permutations of $\{1, 2, .., n\}$ and take the minimum cost route. This leads to an $\Omega(n!)$ solution. The following outlines a way of doing this more efficiently, though using exponential time. Let S be a set k nodes (airports) such that $1 \in S$. Suppose i is another node in S, $i \neq 1$. Suppose we want to find the best way to start from 1, visit all nodes in S and finally land at i. Let $Cost(S, i)$ denote the cost of such an optimal sequence of hops in S. we have two cases:

   - S contains only 1 and i (i.e., $k = 2$). In this case $C(S, i) = c(1, i)$ (the only possibility).
   - If S contains nodes other than 1 and i, then for each $j \in S \setminus \{1, i\}$, we can compute $C(S \setminus i, j) + c(j, i)$, and infer the value of $C(S, i)$.

   Show that this strategy solves the problem in $O(2^n poly(n))$ complexity.