Design and Analysis of Algorithms

1 Semester, 2024-25, Indian Statistical Institute, Bangalore

End Term Examination ; marks 100, Time Limit - 3 hours

November 6, 2024

1. A Greedy Algorithm. [5+10+5=20]

   A set of tasks $T = \{t_1, t_2, \ldots, t_n\}$ is given. Each task is an interval $t_i = (s_i, f_i)$ the start and finish times respectively of the task ($s_i < f_i$).

   A set of tasks is feasible if and only if no two task intervals intersect. For example the task set $\{(1, 10), (5, 20), (15, 20)\}$ is infeasible, however the task set $\{(1, 10), (10, 15), (18, 20)\}$ is feasible.

   The problem is to find the largest possible subset $S \subseteq T$ of the given task set which is feasible.

   The greedy approach is as follows: Initially S is empty, i.e., $S = \{\}$. At each iteration we find among the remaining tasks, the task with minimum finish time, which when added to S keeps S feasible. We repeat this until we can add no more tasks.

   (a) Show that above greedy approach solves the problem and finds a feasible set S with largest possible size.

   (b) Write the algorithm as pseudo code.

   (c) Analyse the running time for your algorithm.

2. Longest Path in a DAG. [15+5=20]

   You are given a Directed Acyclic Graph $G(V, E)$ where each edge has an associated non-negative length.

   (a) Give the pseudo-code of a linear time algorithm (including any other sub-algorithms it may use) to determine the longest directed path in the given graph (the length of a path is the sum of the lengths of the edges in the path).

   (b) Justify the running time.

3. HITTINGSET is NP-Complete. [5+15=20]

   The HITTINGSET problem is defined as follows: You are given the following: A set S of n items, a collection of m subsets of S : $C_1, C_2, .., C_m$, and thirdly an integer K. The problem is to determine if there is some subset $X \subseteq S$ of *at most* K so that X has at least one element in common with each subset C_i in the given collection.

   (a) Show why HITTINGSET is in NP.

   (b) Prove that HITTINGSET is NP-Hard by giving a polynomial time reduction from the vertex cover problem.
   Hint: Note the strong similarity between this problem and vertex cover problem, where m edges are like subsets of size 2 of the n vertices of the vertex set.

4. Stable Supplier-Consumer Flow Problem [15+5=20]

You are given a directed graph $G(V, E)$ with n vertices(or nodes) and m edges. Each edge is marked with a positive integer called the edge capacity. The edge capacity indicates the maximum units that can flow on a (directed) edge.

Additionally, some nodes are also are marked with a non-zero integer.

For a node v if this integer, d_v is positive, it indicates the node wishes to supply d_v units into the network (it is a *supplier* node). This means that the total flow on edges of G coming out of v needs to be greater than the total flow on edges coming into v by the amount d_v.

Similarly, if the node is marked with an integer $-d_v$ (i.e., a negative integer) it indicates the node wishes to consume d_v units from the network(it is a *consumer* node).

If a node has no associated integer, then it is merely a transit node, i.e., its total inflow must equal its total outflow in any flow.

(a) Show how to use the standard (single-source single-sink) $s - t$ network flow problem as a subroutine, to determine if there exists a flow in the network that can support all supplier and consumer node requirements while honoring flow constraints on the edges. Write the pseudo code.

(b) What is the running time of the overall algorithm (in terms of n and m), assuming you use the Edmond-Karp BFS idea for the $s - t$ network flow problem.

5. Subset Sum Problem using dynamic programming. [14+2+4=20]

You are given a set of n items whose weights are represented in an array $W[1..n]$ and you are given a integer K. All the weights and K are positive integers.

You are asked to determine if there exists a subset of the items such that their weights add up exactly to K. For example if $W = [1, 8, 4, 6]$ and $K = 7$ the answer is Yes, because $W[1] + W[4] = 7$. However, if $K = 3$ the answer is No, as no subset of the four items have weights that add up to exactly 3.

This problem is an NP-Complete problem.

A recursive way to approach a solution is using the inclusion-exclusion idea: If there exists such a subset, then either it is ***with*** the n^{th} item or ***without*** the n^{th} item whose weight is $W[n]$. At least one of these is true.

(a) Write pseudo-code for a dynamic programming solution to this problem.
Hint: Use the idea above to formulate a dynamic programming solution, where the i^{th} row , w^{th} column entry in the matrix is a **boolean** value representing if there exists a subset of the first i items (with with weights $W[1..i]$) such that their weights add up exactly to w. Consider what to do when $w = 0$ or $i = 0$.

(b) Analyze the running time of your algorithm. Why is it not considered to be a polynomial time algorithm?

(c) Show the result of the algorithm on the example given with $K = 7$. Only show the state of the matrix after each row is filled.