Quiz 1 (BTech CS) **Advanced Algorithms (CS 315)**
Questions: 6, **Marks:** 25

Instructions: Only YOUR notebook allowed (No electronic devices, textbooks, printed/photocopys).

Part A: Answer one question

1. (5 marks) Show that the solution to the recurrence equation $T(n) = 2T(\sqrt{n}) + log\, n$ is $O(log\,(n)\, log(log\, n))$. [Hint: Change variable to $m = log\ n$.]

2. (5 marks) Give asymptotic upper bound for $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + n$. You can assume $T(n)$ is constant for sufficiently small n. Make your bound as tight as possible.

Part B - Algorithmic strategies: Answer one question

3. (10 marks) (Dynamic programming) Assume you have one machine and a set of n jobs $a_1, \ldots, a_n$ to process on that machine. Each job a_j has processing time t_j, profit p_j and deadline d_j. The machine can process only one job at a time and for all j, job a_j should run uninterruptedly for t_j consecutive time units. If job a_j is completed by its deadline d_j, you receive a profit p_j but if it is completed after the deadline you receive 0 profit. Assume all processing times are integers between 1 and n. Give the following

    1. Give an algorithm to find the schedule which gives you maximum profit.
    2. What is the running time of your algorithm?

4. (10 marks) (Greedy) Many machine learning applications require lot of computation time. Assume you are running an ML company and you have a supercomputer and lots of ordinary PCs.

    You need to run a program which is broken down into n distinct jobs, labeled $j_1, j_2, \ldots, j_n$ which can be performed independently of one another. Each job consists of two stages: first it needs to be preprocessed on the supercomputer and then it needs to be finished on one of the PCs. Let's say that job j_i needs p_i time on the supercomputer, followed by f_i time on an ordinary PC.

    You can assume that there are n PCs available and hence jobs can be run on the PCs simultaneously. But the supercomputer can only work on a single job at a time. The system manager needs to schedule the jobs on the supercomputer. As soon as the first job is finished in the supercomputer, it can be handed off to a PC for finishing; when the second job is done on the supercomputer, it can be processed in the PC regardless of whether the first job is finished or not (since there are n PCs we can do the PC processing in parallel) and so on.

    Your aim is to find out a schedule (an ordering of the jobs) for the supercomputer so that all the processing is finished on the PCs as early as possible. Give the following

    1. Give a polynomial time algorithm for outputting the schedule. Analyse the running time of the algorithm.
    2. Show that there is no schedule better than the output of your algorithm.

Part C - Approximation Algorithms: Answer one question

5. (10 marks) Metric Travelling salesman problem (metric TSP): You are given a complete undirected graph (that is a graph with undirected edges between all pairs of vertices) with non-negative edge costs. Your objective is to find a minimum cost cycle visiting every vertex exactly once. You can also assume that the edge costs satisfy triangle inequality, that is the weight of an edge (u, v) is less than or equal to the weight of any path from u to v. Consider the following algorithm which solves the metric TSP when given an input graph G.

   1. Find a minimum spanning tree T of G.
   2. Double every edge of T (this gives an Eulerian graph - a graph where there is a cycle which visits every edge exactly once).
   3. Find an Eulerian path E on this graph - that is, E visits every edge exactly once.
   4. Output the path that visits vertices of G in the order of their fist appearance in E.

   Question: Argue that the above algorithm gives a 2-approximation.

6. (10 marks) In the knapsack problem, you are given (a) n items of cost $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$ respectively and (b) a weight W. Your objective is to identify a set of items such that the cost is maximized but the total weight of the items is less than or equal to W. We also assume that $w_i \leq W$ for all $i \leq n$. Consider the following greedy algorithm for the knapsack problem.

   1. We initially sort all the items in order of non-increasing ratio of value to weight. That is $v_1/w_1 \geq v_2/w_2 \geq \cdots \geq v_n/w_n$.
   2. If $\sum_{i=1}^{n} w_i \leq W$ Return the set $\{1, 2, \ldots, k\}$.
   3. Set $v^* = max\{v_1, \ldots, v_n\}$. Let i^* be such that $v_{i^*} = v^*$.
   4. Find $k \leq n$ such that $\sum_{i=1}^{k} w_i \leq W$ but $\sum_{i=1}^{k+1} w_i > W$.
   5. If $v^* \geq \sum_{i=1}^{k} v_i$, Return $\{i^*\}$.
   6. Return $\{1, \ldots, k\}$.

   Question: Argue that the above algorithm gives a $\frac{1}{2}$-approximation.