COL351: Analysis and Design of Algorithms	Announced on: Aug 24, 2026 (Monday)

Tutorial Sheet 6

Problems marked with ($\star$) will not be asked in the tutorial quiz.

1. You are given n jobs, each with *length* ℓ_j and *deadline* d_j. Define the *lateness* λ_j of a job j in a schedule σ as the difference $C_j(\sigma) - d_j$ between the job's completion time and deadline, or as 0 if $C_j(\sigma) \leqslant d_j$.

   Which of the following greedy algorithms produces a schedule that minimizes the *maximum* lateness? Feel free to assume that there are no ties. In each case, either provide a counterexample to show incorrectness or a brief argument (2-3 sentences) to show correctness.

   a) Schedule the jobs in increasing order of length ℓ_j.

   b) Schedule the jobs in increasing order of slack $d_j - \ell_j$.

   c) Schedule the jobs in increasing order of deadline d_j.

2. How will your answer to Problem 1 change if, instead of maximum lateness, the goal is to minimize the *total* lateness $\sum_{j=1}^{n} \lambda_j$.

3. You are given as input n jobs, each with a start time s_j and a finish time t_j. Two jobs *conflict* if they overlap in time—if one of them starts between the start and finish times of the other. The goal is to select the maximum-size subset of jobs with no conflicts. (For example, given three jobs consuming the intervals $[0, 3]$, $[2, 5]$, and $[4, 7]$, the optimal solution consists of the first and third jobs.) The plan is to design an iterative greedy algorithm that, in each iteration, irrevocably adds a new job j to the solution so far and removes from future consideration all jobs that conflict with j.

   Which of the following greedy algorithms is guaranteed to compute an optimal solution? Feel free to assume that there are no ties. In each case, either provide a counterexample to show incorrectness or a brief argument (2-3 sentences) to show correctness.

   a) At each iteration, choose the remaining job with the earliest start time.

   b) At each iteration, choose the remaining job with the earliest finish time.

   c) At each iteration, choose the remaining job that requires the least time, i.e., with the smallest value of $t_j - s_j$.

   d) At each iteration, choose the remaining job with the fewest number of conflicts with other remaining jobs.

4. Given a list of n natural numbers $d_1, d_2, \ldots, d_n$, show how to decide in polynomial time whether there exists a simple undirected graph $G = (V, E)$ whose node degrees are precisely the numbers $d_1, d_2, \ldots, d_n$. That is, if $V = (v_1, v_2, \ldots, v_n)$, then the degree of v_1 should be d_1, the degree of v_2 should be d_2, and so on.

5. Consider the following change-making problem: The input to this problem is an integer L. The output should be the minimum cardinality collection of coins required to make L shillings of change, that is, you want to use as few coins as possible. The coins are worth $1, 5, 10, 20, 25$, and 50 shillings. Assume that you have an unlimited number of coins of each type. Formally prove or disprove that the greedy algorithm that takes as many coins as possible from the highest denominations correctly solves the problem. So, for example, to make a change for 234 Shillings, the greedy algorithm would require ten coins: four 50 shilling coins, one 25 shilling coin, one 5 shilling coin, and four 1 shilling coins.

6. Consider another change-making problem: The input to this problem is again an integer L, and the output should again be the minimum cardinality collection of coins required to make L nibbles of change (that is, you want to use as few coins as possible). Now the coins are worth $1, 2, 2^2, 2^3, \ldots, 2^{1000}$ nibbles. Assume that you have an unlimited number of coins of each type. Prove or disprove that the greedy algorithm that takes as many coins of the highest value as possible solves the change-making problem.

   Combined hint for Problems 5 and 6: The greedy algorithm is correct for one of the two problems and is incorrect for the other. For the problem where greedy is correct, you could use the following proof strategy: Assume, to reach a contradiction, that there is an input I on which greedy is not correct. Let $\texttt{OPT}(I)$ be a solution for input I that is better than the greedy output $G(I)$. Show that the existence of such an optimal solution $\texttt{OPT}(I)$ that is different than greedy is a contradiction. What you can conclude from this is that for every input, the output of the greedy algorithm is the unique optimal/correct solution.

7. Suppose there are n agents and m items. The value of agent i for item j is given by a nonnegative integer $v_{i,j}$. An agent's value for a set of items is the sum of its values for individual items in that set. The goal is to partition the m items among the n agents in a *fair* manner.

   Denote an allocation by $A := (A_1, A_2, \ldots, A_n)$, where A_i is the subset of items assigned to agent i. We require that for any $i \neq k$, $A_i \cap A_k = \emptyset$ (i.e., items are not shared between bundles) and $\cup_i A_i$ is the entire set of items (i.e., no item is left unallocated). An allocation is deemed fair if, for any pair of agents i and k, the value derived by agent i from its bundle A_i is "within an item" of the value it derives from agent k's bundle A_k; specifically, for every pair of agents i and k and for every item $j \in A_k$, we have that $v_i(A_i) \geqslant v_i(A_k \setminus \{j\})$, where $v_i(S)$ denotes the value of agent i for a subset S of items.

   Design a polynomial-time algorithm for computing a fair allocation when the agents have *identical* valuations, i.e., item j is valued at $v_j \geqslant 0$ by every agent (though, for distinct items j and j', the values v_j and $v_{j'}$ may differ).[1]

8. How would your solution to the previous problem change if v_j is allowed to be an arbitrary integer (i.e., negative, zero, or positive)?

   Due to the presence of negative values, we need to redefine fairness. An allocation is deemed fair if, for any pair of agents i and k, the value of bundle A_i is "within an item" of the value of bundle A_k; specifically, for every pair of agents i and k:

   - for every item $j \in A_i$ such that $v_j < 0$, we have $v(A_i \setminus \{j\}) \geqslant v(A_k)$, and
   - for every item $j \in A_k$ such that $v_j \geqslant 0$, we have $v(A_i) \geqslant v(A_k \setminus \{j\})$.

   Your task is to design a polynomial-time algorithm for computing a fair allocation.

9. ($\star$) Imagine you have a set of n course assignments given to you today. For each assignment i, you know its deadline d_i and the time ℓ_i it takes to finish it. With so many assignments, it may not be possible to finish all of them on time. If you finish an assignment after its deadline, you get zero marks. Therefore, you must either complete the assignment by the deadline or not at all. How can you determine the maximum number of assignments you can complete within their deadlines?

10. ($\star$) Byomkesh has been assigned to go on a mission to the mansion of his arch nemesis, Anukul. To limit exposure, he has decided to travel via an underground sewer network. He has a map of the sewer, consisting of n bidirectional *pipes* that connect to one another at *junctions*. Each junction connects to at most four pipes, and every junction is reachable from every other junction via the sewer network. Every pipe is marked with its positive integer length. Some junctions are marked as containing identical motion sensors, any of which can detect Byomkesh if his distance to that sensor (as measured along the pipes in the sewer network) is too close. Unfortunately, Byomkesh does not know the sensors' sensitivity.

    Describe an $O(n \log n)$ time algorithm to find a path along pipes from a given entrance junction to the junction below Anukul's mansion that stays as far from motion sensors as possible.

    Hint#1: Byomkesh does not know the sensitivity of the motion sensors. If the sensitivity is too high, there may not be a feasible path to Anukul's mansion. On the other hand, if the sensitivity is zero, the network's connectivity would imply that such a path certainly exists. Think about the *maximum* sensitivity for which such a path still exists. Can you help Byomkesh discover this quantity?

[1] If you can show the existence of a fair allocation without the identical valuations assumption, please meet Rohit.

Hint#2: Some junctions have sensors, while other junctions can be detected from the ones with sensors, and still others may be out of range (and therefore “safe”). Specifically, for a fixed (unknown) sensitivity level of the sensors, a junction is safe if its shortest distance from every junction containing a motion sensor is strictly greater than the sensitivity level. The desired path (if it exists) should only use the safe junctions.

Hint#3: Consider adding an *auxiliary* vertex to your graph if it helps.