CS218 Design and analysis of algorithms **Jan-Apr 2024**

Exercises: Greedy, Dynamic Programming

Greedy algorithms

1. Let $G(V, E)$ be a graph with edge weights and $V = V_1 \cup V_2$ be a partition of vertices. Let C be the set of cut edges connecting V_1 with V_2, i.e.,

   $$C = \{(u, v) : u \in V_1, v \in V_2\}.$$

   Let e^* be the minimum weight edge in C. Prove that there must exists a minimum weight spanning tree containing e^*.

2. Let $G(V, E)$ be a graph with edge weights and let v_1 be an arbitrary vertex. Let e^* be the minimum weight edge incident on v_1. Prove that there must exists a minimum weight spanning tree containing e^*.

3. Suppose you are given a set of n course assignments today, each of which has its own deadline. Let the i-th assignment have deadline d_i and suppose to finish the i-th assignment it takes ℓ_i time. Given that there are so many assignments, it might not be possible to finish all of them on time. If you finish an assignment at time t_i which is more than its deadline d_i, then the difference $t_i - d_i$ is called the lateness of this assignment (if $t_i < d_i$ then the lateness is zero). Since you want to maintain a balance among courses, you want that the maximum lateness over all assignments is as small as possible. You want to find a schedule for doing the assignments which minimizes the maximum lateness over all assignments.

   Example: $d_1 = 20, \ell_1 = 10$, $d_2 = 40, \ell_2 = 20$, $d_3 = 60, \ell_3 = 30$. If the assignments are done in order $(1, 3, 2)$ then the maximum lateness will be 20 (for assignment 2). If the assignments are done in order $(1, 2, 3)$ then the maximum lateness will be 0.

   Can you show that a greedy algorithm will give you an optimal solution?

   - Greedy Strategy 1: Do the assignments in increasing order of their lengths (ℓ_i).
   - Greedy Strategy 2: Do that assignment first whose deadline is the closest.
   - Greedy Strategy 3: Do that assignment first for which $d_i - \ell_i$ is the smallest.

   Strategies 1 and 3 don't work. Give examples to show that they don't work.

   Strategy 2 works. Prove that it works correctly as follows. Consider any particular schedule for the assignments. Pick any two assignments A_i and A_j which are adjacent, that is, A_j is done immediately after A_i. If $d_i \leq d_j$ do then nothing, but if $d_i > d_j$ then swap the positions of A_i and A_j. Argue that after this step, the maximum lateness cannot increase.

   Repeatedly use the same argument to show that increasing order of deadlines is an optimal schedule.

4. Consider another variant. Now, all assignments are equally long, so let's say each takes a unit time to finish. The ith assignment has deadline d_i and a reward r_i. You get the reward only if you finish the assignment within the deadline, otherwise the reward is zero. Design an algorithm to find the maximum possible reward you can get.

5. Hard problem. Consider another variant. For each assignment, you know its deadline d_i and the time ℓ_i it takes to finish it. Suppose you get zero marks for finishing an assignment after its deadline. So, either you should do the assignment within the deadline or not do it at all. How will you find the maximum number of assignments possible within their deadlines.

6. Given a set of intervals, you need to assign a color to each interval such that no two intersecting intervals have the same color. Design an efficient algorithm find a coloring with minimum number of colors. To put the problem in another way, given arrival and departure times of trains at a station during the day, what is the minimum number of platforms that is sufficient for all trains.

7. Given a list of n natural numbers $d_1, d_2, \ldots, d_n$, we want to check whether there exists an undirected graph G on n vertices whose vertex degrees are precisely $d_1, d_2, \ldots, d_n$ (that is the ith vertex has degere d_i) and construct such a graph if one exists. G should not have multiple edges between the same pair of vertices and should not have self-loop (edges having same vertex as the two endpoints).

   Example 1: (2, 1, 3, 2) . The graph with the set of edges $(v_1, v_3), (v_1, v_4), (v_2, v_3), (v_3, v_4)$ has this degree sequence.

   Example 2: (3, 3, 1, 1) . There is no graph on four vertices having these degrees.

8. Suppose you are an advertisement company who wants to advertise something to all n people in the city. You know that each of these n people will come to the city center on Sunday for some interval of time. You have acquired these time intervals for all people through some unethical means. You cannot put ads at the city center, but you can pay people to carry your ad on them (maybe by wearing a t-shirt). Assume that if X is carrying the ad, then anyone whose time interval intersects with the time interval of X will see the ad (of course, X will also see the ad). You want to choose minimum number of people to whom you should pay so that everyone sees the ad. Design an algorithm for this and prove its correctness.

9. Suppose you have n objects and you are given pairwise distances between those objects $\{d_{i,j} : 1 \leq i < j \leq n\}$. For a clustering of these objects into k clusters, say $S_1 \cup S_2 \cup \cdots \cup S_k = \{1, 2, \ldots, n\}$, we define the spacing of this clustering is the minimum distance between any two objects that are in different clusters. Formally, the spacing is

   $$\min\{d_{i,j} : 1 \leq i < j \leq n, i \text{ and } j \text{ belong to different clusters }\}.$$

   We want to find a clustering with k clusters that maximizes the spacing. Can you do this in $O(n \log n)$ time? Think first about the $k = 2$ case.

Dynamic Programming

Flavour 1: where the subproblems are on suffixes/prefixes of the input.

10. Work out the remaining details of the problem of finding a disjoint intervals that maximizes the total length. Argue that the running time is $O(n \log n)$.

11. The algorithm we described in the class only computed the optimal cost. Update the pseudocode to also output an optimal set of intervals.

12. Design an $O(n \log n)$ time algorithm for the weighted version of the above problem: each interval has a weight and we want to select a disjoint subset of intervals that maximizes the total weight.

13. Design an $O(n \log n)$ time algorithm for the counting version of the above problem, that is, we want to find the number of possible disjoint subsets of intervals.

14. You are going on a car trip from city A to city B that will take multiple days. On the way, you will encounter many cities. You plan to drive only during the day time and on each night you will stay in one of the intermediate cities. Suppose you can drive at most d kilometers in a day. You are given the distances of the intermediate cities from city A, say, $d_1, d_2, \ldots, d_n$. And distance of B from A is d_{n+1}. You are also given the costs of staying in various cities for one night, say, $c_1, c_2, \ldots, c_n$. Find a travel schedule, that is, in which all cities you should do a night stay, such that your total cost of staying is minimized.

15. We are given a directed graph, where each edge goes from a lower index vertex to a higher index vertex. Want to find the longest path from vertex 1 to vertex n.

16. Given a sequence of numbers, find the longest increasing subsequence in $O(n \log n)$ time.

17. Given an array of integers, you want to find a subset with maximum total sum such that no two elements in the subset are adjacent. For example, for the array $\{6, 4, 3, 2, 1, 5\}$, the desired subset is $\{6, 3, 5\}$ with total sum 14. Design an $O(n)$-time algorithm for this problem, where n is the length of the array.

Flavour 2: where the subproblem is on a substring of the input.

18. The naive algorithm to multiply two matrices of dimensions $p \times q$ and $q \times r$ takes time $O(pqr)$. Suppose we have four matrices A, B, C, D which are $2 \times 4, 4 \times 3, 3 \times 2, 2 \times 5$ respectively. If you multiply $ABCD$ in the order $(AB)(CD)$, it will take time $2 \times 4 \times 3 + 3 \times 2 \times 5 + 2 \times 3 \times 5 = 84$, on the other hand if you multiply in the order $A((BC)D)$, it will take time $4 \times 3 \times 2 + 4 \times 2 \times 5 + 2 \times 4 \times 5 = 104$.

    Given matrices $A_1, A_2, \ldots, A_n$ with dimensions $p_1 \times p_2, p_2 \times p_3, \ldots, p_n \times p_{n+1}$, design an algorithm to find the order in which you should multiply $A_1 A_2 \cdots A_n$ which minimizes the multiplication time.