CS345 : Algorithms II
Semester I, 2021-22, CSE, IIT Kanpur

Assignment II

Deadline : 11:55 PM, 2nd October 2021.

Most Important guidelines

- It is only through the assignments that one learns the most about the algorithms and data structures. You are advised to refrain from searching for a solution on the net or from a notebook or from other fellow students. Remember - **Before cheating the instructor, you are cheating yourself**. The onus of learning from a course lies first on you. So act wisely while working on this assignment.
- Refrain from collaborating with the students of other groups. If any evidence is found that confirms copying, the penalty will be very harsh. Refer to the website at the link: https://cse.iitk.ac.in/pages/AntiCheatingPolicy.html regarding the departmental policy on cheating.

General guidelines

1. There are three problems in this assignment: 2 Difficult problems and 1 Easy. Each difficult problem carries 100 marks, and the easy one carries 70 marks. Attempt **only** one of the 3 problems.

2. You are strongly discouraged to submit the scanned copy of a handwritten solution. Instead, you should prepare your answer using any text processing software (LaTex, Microsoft word, ...). The final submission should be a single pdf file.

3. You need to justify any claim that you make during the analysis of the algorithm. But you must be formal, concise, and precise.

4. If you are asked to design an algorithm, you may state the algorithm either in plain English or a pseudocode. But it must be formal, complete, unambiguous, and easy to read. You must not submit any code (in C++ or C, python, ...).

5. **Naming the file**:
   The submission file has to be given a name that reflects the information about the assignment number, version attempted (difficult/moderate/esay), and the roll numbers of the 2 students of the group. For example, you should name the file as **D_2_Rollnumber1_Rollnumber2.pdf** if you are submitting the solution for the difficult version of the 1st assignment. In a similar manner, **E_2_Rollnumber1_Rollnumber2.pdf** if you are submitting the solution for the easy problem of the 2nd assignment.

6. Both students of a group have to upload the identical copy of their final submission. Be careful during the submission of an assignment. Once submitted, it can not be re-submitted.

7. Deadline is strict. Make sure you upload the assignment well in time to avoid last minute rush.

8. Contact TA at the email address: kbhanja@cse.iitk.ac.in for all queries related to the submission of the assignment. Avoid sending any such queries to the instructor.

Difficult

1. A large collection of mobile wireless devices can naturally form a network in which the devices are the nodes, and two devices x and y are connected by an edge if they are able to directly communicate with each other (e.g., by a short-range radio link). Such a network of wireless devices is a highly dynamic object, in which edges can appear and disappear over time as the devices move around. For instance, an edge (x, y) might disappear as x and y move apart from each other and lose ability to communicate directly.

   In a network that changes over time, it is natural to look for efficient ways of maintaining a path between certain designated nodes. There are two opposing concerns in maintaining such a path: we want paths that are short, but we also do not want to have to change the path frequently as the network structure changes. That is, we would like a single path to continue working, if possible, even as the network gains and loses edges. Here is a way we model this problem.

   Suppose we have a set of nodes V, and at a particular point in time there is a set E_0 of edges among the nodes. As the nodes move, the set of edges changes from E_0 to E_1, then to E_2, then to E_3, and so on, to an edge set E_b. For $i = 0, 1, 2, ..., b$, let G_i denote the graph (V, E_i). So if we were to watch the structure of the network on the nodes V as a "time lapse", it would look precisely like the sequence of graphs $G_0, G_1, \ldots, G_b$. We will assume that each of these graphs G_i is connected.

   Now consider two particular nodes $s, t \in V$. For an $s - t$ path P in one of the graphs G_i, we define the *length* of P to be simply the number of edges in P, and we denote this by $\ell(P)$. Our goal is to produce a sequence of paths $P_0, \ldots, P_b$ so that for each i, P_i is an $s - t$ path in G_i. We want the paths to be relatively short. We also do not want there to be too many *changes* - points at which the identity of the path switches. Formally, we define *changes*$(P_0, \ldots, P_b)$ to be the number of indices i $(0 \leq i \leq b - 1)$ for which $P_i \neq P_{i+1}$.

   Fix a constant $K > 0$. We define the *cost* of the sequence of paths $P_0, \ldots, P_b$ to be

$$\textit{cost}(P_0, P_1, \ldots, P_b) = \sum_{i=0}^{b} \ell(P_i) + K \cdot \textit{changes}(P_0, P_1, \ldots, P_b)$$

   Give a polynomial time algorithm to find a sequence of paths $P_0, \ldots, P_b$ of minimum cost.

2. We have already seen Bellman Ford algorithm which solves single source shortest paths (SSSP) problem in $O(mn)$ time for directed graphs with potentially negative weighted edges but no negative cycle. Consider a related problem of finding shortest paths between every pair of vertices in such a graph. This problem is commonly known as the All-Pair Shortest Paths (APSP) problem. One can trivially solve APSP problem in $O(mn^2)$ time by executing Bellman Ford algorithm on each of the n vertices. However, there exists a better and novel algorithm to solve APSP problem which takes $O(n^3)$ time. Interestingly, it is also based on the Dynamic Programming technique, though applied in a manner which is totally different from Bellman Ford algorithm. We now provide some notations in the following paragraph which will guide you towards the desired algorithm.

   Recall that the vertices in a graph are labeled (numbered) from 1 to n. Let $P^k(i,j)$ be the set all paths from vertex i to vertex j whose intermediate vertices have label at most k. Let $p^k(i,j)$ be the shortest path from the set $P^k(i,j)$ and $\delta^k(i,j)$ be the corresponding distance (length of the shortest path). Try to explore answers to the following questions : What would $\delta^0(i,j)$ mean ? What would $\delta^n(i,j)$ mean ? How would $p^k(i,j)$ look like for any $0 < k$? Use the insight from the answers of the above questions to derive a recurrence for $\delta^k(i,j)$. Interestingly, the final algorithm consist of a few nested for-loops.

   As a part of the assignment, you have to achieve the following two objectives.

   (a) Design an algorithm which processes a given directed graph in $O(n^3)$ time and outputs a $n \times n$ matrix D storing all-pairs distances in the graph. The space used by the algorithm has to be $O(n^2)$. As a hint, note that the code of this algorithm uses just three nested for-loops and is very short and simple.

   (b) Extend the above algorithm to output an $O(n^2)$ size data structure using which any shortest path can be reported in optimal time, that is, the time of the order of the number of edges on the shortest path. You must also describe the corresponding algorithm to report any shortest path using the data structure.

   An important note: Instead of searching for the solution of the above problem elsewhere, make a sincere attempt to solve it on your own. Take this problem as an opportunity to improve your skills of dynamic programming; you will realize that it will be a very satisfying experience.

Easy

There are n jobs: $J_1, \ldots, J_n$. Job J_i has a deadline d_i and a required processing time t_i, and all jobs are available to be scheduled starting at time t_i, and all jobs are available to be scheduled starting at time s. For a job i to be done, it needs to be assigned a period from $s_i \geq s$ to $f_i = s_i + t_i$, and different jobs should be assigned nonoverlapping intervals. As usual, such an assignment of times will be called a schedule.

Each job must either be done by its deadline or not at all. We will say that a subset J of jobs is schedulable if there is a schedule for the jobs in J so that each of them finishes by its deadline. Your problems is to select a schedulable subset of maximum possible size and give a schedule for this subset that allows each job to finish by its deadline.