CS345 : Algorithms II
Semester I, 2017-18, CSE, IIT Kanpur

Assignment 4

Deadline : 6:00 PM on 31st October, 2017

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. **Before cheating the instructor, you are cheating yourself**. The onus of learning from a course lies first on you and then on the quality of teaching of the instructor. So act wisely while working on this assignment.
- There are three exercises in this assignments. Each exercise has two problems- one *easy* and one *difficult*. Submit exactly one problem per exercise. It will be better if a student submits a correct solution of an easy problem that he/she arrived on his/her own instead of a solution of the difficult problem obtained by hints and help from a friend. Do not try to be so greedy :-).
- The answer of each questions must be formal, complete, and to the point. Do not waste your time writing intuition or idea. There will be penalty if you provide any such unnecessary details.

1 Dynamic Programming

Attempt one of the following problems.

Mobile network problem

(*25 marks*) 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 $\mathit{changes}(P_0, \ldots, P_b)$ 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

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

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

Job scheduling

(*20 marks*) 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.

1. Prove that there is an optimal solution J (i.e., a schedulable set of maximum size) in which the jobs in J are scheduled in increasing order of their deadlines.

2. Assume that all deadlines d_i and the required times t_i are integers. Give an algorithm to find an optimal solution. Your algorithm should run in time polynomial in the number of jobs n, and the maximum deadline $D = \max_i d_i$.

2 Max flow algorithms

Attempt any one of the following problems.

Analysis of Ford Fulkerson Algorithm

(*marks=30)*

Recall the Ford-Fulkerson algorithm we discussed in the class. There is a directed graph on n vertices and m edges where each edge has unit capacity. There is also a degnated source vertex s and a designated sink vertex t. The distance from s to t is $O(n^{2/3})$. Prove rigorously that the Ford-Fulkerson algorithm will take $O(mn^{2/3})$ time to compute the maximum flow from s to t.

Non-terminating example of Ford-Fulkerson algorithm

(*marks=20)*

Recall the example of 6 node flow-network (discussed briefly in Lecture 23) on which it is possible that Ford-Fulkerson algorithm will never terminate. Moreover, even asymptotically, the flow computed will be less than max-flow possible. The detailed sketch of this example is given at the end of the Lecture 23. Provide complete details of the analysis.

3 Max flow Application

An atmospheric science experiment

Your friends are involved in a large scale atmospheric experiment. They need to get good measurements on a set S of n different conditions in the atmosphere (such as the ozone level at various places), and they have a set of m balloons that they plan to send up to make these measurement. Each balloon can make at most two measurements.

Unfortunately not all balloons are capable of measuring all conditions, so for each balloon $i = 1, \ldots, m$, they have a set of S_i of conditions that balloon i can measure. Finally, to make the results more reliable, they plan to take each measurement from at least k different balloons. (Note that a single balloon should not measure the same condition twice). They are having trouble figuring out which conditions to measure on which balloon.

1. (*marks = 17*)

   Give a polynomial-time algorithm that takes the input to an instance of this problem (the n conditions, the sets S_i for each of the m balloons and the parameter k) and decides whether there is a way to measure each condition by k different balloons, while each balloon only measures at most two conditions.

2. (*marks = 8*)

   There is one more constraint that has to be followed. Each of the balloons is produced by one of three different sub-contractors involved in the experiment. A requirement of the experiment is that there be no conditions for which all k measurements come from balloons produced by single sub-contractor. Explain how to modify your polynomial time algorithm for part (a) into a new algorithm that decides whether there exists a solution satisfying all the conditions from (a) plus a new constraint about the subcontractors described above.