CS345 : Algorithms II
Semester I, 2020-21, CSE, IIT Kanpur

Assignment 7

Deadline : 11:55 PM, 22nd November 2020.

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.
- **Grading policy:**
  There will be no penalty for submission based on the time of submission as long as the submission is before the deadline.
- 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 2 problems in this assignment - Difficult and Easy. Each carries equal marks. You have to submit the solution of exactly one of them.
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. You may use the results proved in the class. But, if you wish to use any homework problem in your solution, you must provide its solution as well.
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, and the roll numbers of the 2 students of the group. For example, if you are submitting the solution of a difficult problem, you should name the file as **D_Rollnumber1_Rollnumber2.pdf**. If you are submitting the solution for the easy problem, the name should be **E_Rollnumber1_Rollnumber2.pdf**.
6. **Each student of a group** has to upload the submission file separately unlike in the past when only one submission file per group had to be uploaded. 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: **shreyasa@cse.iitk.ac.in** for all queries related to the submission of this assignment. Avoid sending any such queries to the instructor.

Job Scheduling using Max-Flow

Suppose you are managing a collection of processors and must schedule a sequence of jobs over time.

The jobs have the following characteristics. Each job j has an arrival time a_j when it is first available for processing, a length ℓ_j which indicates how much processing time it needs and a deadline d_j by which it must be finished. (We'll assume $0 \leq \ell_j \leq d_j - a_j$.) Each job can be run on any of the processors, but only on one at a time; it can also be preempted and resumed from where it left off (possibly after a delay).

Moreover, the collection of processors is not entirely static either: you have an overall pool of k possible processors; but for each processor i there is an interval of time $[t_i, t'_i]$ during which it is available; it is unavailable at all other times.

Given all this data about job requirement and processor availability, you would like to decide whether the jobs can all be completed or not. Give a polynomial time algorithm that either produces a schedule completing all jobs by their deadline or reports (correctly) that no such schedule exists. You may assume that all the parameters associated with the problem are integers.

Example: Suppose we have 2 jobs J_1 and J_2. J_1 arrives at time 0, is due at time 4, and has length 3. J_2 arrives at time 1, is due at time 3, and has length 2. We also have 2 processors P_1 and P_2. P_1 is available between times 0 and 4; P_2 is available between times 2 and 3. In this case, there is a schedule that gets both jobs done.

- At time 0, we start J_1 on processor P_1.
- At time 1, we preempt J_1 to start J_2 on P_1.
- At time 2, we resume J_1 on P_2.(J_2 continues processing on P_1.)
- At time 3, J_2 completes by its deadline. P_2 ceases to be available, so we move J_1 back to P_1 to finish its remaining one unit of processing there.
- At time 4, J_1 completes its processing on P_1.

Notice that there is no solution that does not involve preemption and moving off jobs.

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.

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.