CS345 : Algorithms II
Semester I, 2016-17, CSE, IIT Kanpur

Assignment 2

Deadline : 6:00 PM, Friday, 26 August 2016

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 two 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 aim of the second exercise is to make you realize that one must think with a free and unconditioned mind to solve a problem. Though we discuss some well known algorithm paradigms (divide and conquer, greedy strategies, dynamic programming), our problem solving approach must not become conditioned to apply these paradigms mechanically. There are algorithms for many fundamental problems that can not be put into any of these paradigms. Through the second exercise, you will be amazed to see that there is a such a simple algorithm for such a nontrivial problem. Enjoy ...

1 Greed is not always a curse!

Attempt exactly one of the two problems.

1.1 Jobs on supercomputer

(marks=25)

There are n jobs. There are n PCs and one supercomputer. Each job consists of two stages: first it needs to be preprocessed on the supercomputer, and then it needs to be finished on one of the PCs. Let us say that job J_i needs p_i seconds of time on the supercomputer, followed by f_i seconds of time on a PC. Since there are n PCs available, the finishing of the jobs can be performed in parallel - all jobs can be processed at the same time on their respective PCs. However, the supercomputer can only work on a single job at a time. So we need to find out the order in which to feed the jobs to the supercomputer. Our aim is to minimize the completion time of the schedule which is defined as the earliest time at which all jobs will have finished processing on the PCs. Design a greedy strategy/algorithm that finds the order in which the jobs should be scheduled on the supercomputer so that completion time achieved is as small as possible.

1.2 Hierarchical metric

(marks=40)

There is a complete graph G on n vertices $V = \{v_1, \ldots, v_n\}$. There is a mapping $d : V \times V \to R^+$ such that $d(v_i, v_j) \geq 0$ for each $i \neq j$. Further $d(v_i, v_i) = 0$ for all i.

Consider a rooted tree T whose leaves are V and each internal node x has a label $h(x) \geq 0$. For any two vertices $v_i, v_j \in V$, their distance in T is defined as $h(x)$ where x is the lowest common ancestor of v_i and v_j in T. T is said to be consistent with G if distance between each $v_i, v_j \in V$ in tree T is less than or equal to $d(v_i, v_j)$. Aim is to compute a tree T^* for a given graph G satisfying the following conditions.

- T^* is consistent with G
- For any other tree T' consistent with G, the following condition must hold for each $v_i, v_j \in V$:
  "distance between v_i and v_j in T' is less than or equal to their distance in T^*."

You need to design a greedy algorithm to compute tree T^* using the generic strategy we discussed in the class for greedy algorithms.

1. Provide complete details of the greedy step and the smaller instance G' obtained by applying the greedy step.
2. State the theorem that relates $T^*(G)$ with $T^*(G')$ and give a short proof for it.
3. Describe the fastest possible implementation of the algorithm based on the greedy step described above.

2 Miscellaneous problems

Attempt exactly one of the following two problems.

2.1 Minimum spanning tree

(*marks=20*)

Given an undirected weighted graph $G = (V, E)$ on $n = |V|$ vertices and $m = |E|$ edges, consider the following algorithm which executes in phases and computes its minimum spanning tree T. Initially $\mathcal{G} = G$, $T = \emptyset$.

While $\mathcal{G}$ has more than one vertex **do**

1. Each vertex in $\mathcal{G}$ selects least weight edge incident on it. Let $\mathcal{F}$ be the set of edges thus selected.
2. $\mathcal{F}$ will partition the set of vertices of $\mathcal{G}$ into connected components. Let $C_1, ..., C_i$ be these components. Construct a new graph G' as follows : Each component C_i corresponds to a single vertex in G'. The edges in G' will be only those edges from $\mathcal{G}$ whose endpoints belong to distinct components. The graph G' thus constructed may be a multigraph since there may be multiple edges between its vertices. We transform G' to a simple graph by keeping only the least weight edge from the set of multiple edges between each pair of vertices.
3. $\mathcal{G} \leftarrow G'$; $T \leftarrow T \cup \mathcal{F}$.

Give a short proof of correctness of the above algorithm and present an $O(m \log^2 n)$ time implementation of this algorithm.

2.2 A novel algorithm

(*marks=40*)

Let $G = (V, E)$ be an undirected, unweighted and connected graph on $n = |V|$ vertices. We wish to compute a subgraph $H = (V, E_S)$, $E_S \subseteq E$ such that for each pair $u, v \in V$, the distance between them in H is at most 3 times the distance in G. To achieve this objective, it suffices if for each edge $(u, v) \in E, (u, v) \notin E_S$, there exists a path of at most 3 edges in H connecting u and v. The following is the skeleton of an algorithm to compute such a subgraph H of size $O(n^{3/2})$.

Fill in the blanks appropriately. You must also present the fastest implementation of the algorithm along with a short proof of correctness.

Algorithm 1: An inspirational algorithm

```
E_S ← ∅;
while _____ do
    Pick any vertex v from G;
    if degree(v)≤ √n then
        Add all edges incident on v to E_S;
        remove v and all edges incident on v from G;
    else
        _____ ;
        _____ ;
        _____ ;
return E_S;
```

Hint (if you like): The Else part of the If statement also involves similar activities as then part.