Design and Analysis of Algorithms
Indian Institute of Technology Kanpur

Worked out Assignment on *Greedy Paradigm*

An Important guideline

- It is only through the assignments that one learns the most about the algorithms and data structures. For the current assignment, a sequence of hints are given. Try to look at a hint only after trying on your own for sufficiently long period of time.
- Ideally, one should look at one hint per day and then spend around 20-25 minutes to materialize the hint. The best way - go for a stroll after dinner pondering over the hint(s).
- You should judge your performance by the number of hints you indeed use – more hints you use, less is your performance.
- The onus of learning from a course lies first on you. So act wisely while working on this assignment.

Problem Statement

1. Suppose we are given a set of points $P = \{p_1, p_2, \ldots, p_n\}$, together with a distance function d on the set $P; d$ is simply a function on pairs of points in P with the properties that $d(p_i, p_j) = d(p_j, p_i) > 0$ if $i \neq j$, and that $d(p_i, p_i) = 0$ for each i.

   We define a hierarchical metric on P to be any distance function τ that can be constructed as follows. We build a rooted tree T with n leaves, and we associate with each node v of T (both leaves and internal nodes) a height h_v. These heights must satisfy the properties that $h(v) = 0$ for each leaf v, and if u is the parent of v in T, then $h(u) \geq h(v)$. We place each point in P at a distinct leaf in T. Now, for any pair of points p_i and p_j, their distance $\tau(p_i, p_j)$ is defined as follows. We determine the least common ancestor v in T of the leaves containing p_i and p_j, and define $\tau(p_i, p_j) = h_v$.

   We say that a hierarchical metric τ is consistent with our distance function d if, for all pairs i, j, we have $\tau(p_i, p_j) \leq d(p_i, p_j)$.

   Give a polynomial-time algorithm that takes the distance function d and produces a hierarchical metric τ with the following properties.

   (i) τ is consistent with d, and

   (ii) if τ' is any other hierarchical metric consistent with d, then $\tau'(p_i, p_j) \leq \tau(p_i, p_j)$ for each pair of points p_i and p_j.

τ is called the optimal hierarchical metric.

If we think of a brute force approach to compute the optimal hierarchical metric, what can be possible search space? There can be many tree structure possible on a set of n-leaves. How to search for the tree structure that corresponds to the optimal hierarchical metric ? This indicates the difficulty of the problem. At the same time, it also gives a direction...

Can we state properties of the tree that corresponds to the optimal hierarchical metric ?

Proceed along similar (not same) lines as our approach for solving the optimal prefix coding.

Let T be the tree corresponding to an optimal hierarchical metric. We can always obtain another optimal hierarchical metric whose tree structure will be such that each internal node has $\underline{\text{exactly}}$ 2 children. Try to establish this property.

Getting inspired from our solution for the optimal prefix coding, think of ways to reduce the given problem instance to a smaller instance of the same problem. This is the most important step. Try really really hard on your own before the most important hint on the next page.

Let (p_1, p_2) be the pair of points with the minimum distance among all pairs of points from P. Replace these two points by another point, say p'. How will you define distance of p' from the remaining $(n-2)$ points ? Spend some time to get a better understanding of this smaller instance.

State and prove a suitable theorem that relates the solution of the original instance and the smaller instance.

How will you retrieve the optimal metric of the original instance from the optimal metric of the smaller instance ?

Note: It is possible to design an algorithm for this problem that takes a totally different approach (relating it to MST). However, the aim of the solution sketched above is to make you more comfortable and confident in the use of the generic strategy for designing and analysing a greedy algorithm.