Modern Algorithm Design (Monsoon 2024)
Homework 2

Deadline: 30th September, 2024, 11:59 pm (IST)

Release Date: 16th September, 2024

1. (*10 points*) This problem concerns running time optimizations to the Hungarian algorithm for computing minimum-cost perfect bipartite matchings. Recall the $O(mn^2)$ running time analysis from lecture: there are at most n augmentation steps, at most n price update steps between two augmentation steps, and each iteration can be implemented in $O(m)$ time.

   (a) By a phase, we mean a maximal sequence of price update iterations (between two augmentation iterations). The naive implementation in lecture regrows the search tree from scratch after each price update in a phase, spending $O(m)$ time on this for each of up to n iterations. Show how to reuse work from previous iterations so that the total amount of work done searching for good paths, in total over all iterations in the phase, is only $O(m)$.

   (b) The other non-trivial work in a price update phase is computing the value of δ (the minimum magnitude by which the prices can be updated without violating any of the invariants). This is easy to do in $O(m)$ time per iteration. Explain how to maintain a heap data structure so that the total time spent computing δ over all iterations in the phase is only $O(m \log n)$. Be sure to explain what heap operations you perform while growing the search tree and when executing a price update.

   [This yields an $O(mn \log n)$ time implementation of the Hungarian algorithm.]

2. (*10 points*) We used low-stretch trees to approximate APSP on general graphs. We explore this connection further via the k-point-facility problem: Given a graph $G = (V, E)$ with positive edge lengths and $k \in \mathbb{Z}_{\geq 0}$. Define $d_G(u, v)$ to be the length of the shortest path between u and v in G according to these edge-lengths. For a set $C \subseteq V$, define

$$d_G(v, C) := \min_{c \in C} d_G(v, C)$$

   The k-point-facility problem asks you to find a set $C \subseteq V$ with $|C| = k$ to minimize $\Phi_G(C) := \sum_{v \in V} d_G(v, C)$.

   (a) Design a poly(k, n) time algorithm to solve the k-point-facility problem on an edge-weighted path of n vertices.

   (b) Given an algorithm to solve k-point-facility optimally on trees, show that the algorithm that samples a tree T from (random) low-stretch tree distribution with stretch α, solves k-point-facility on T to get C_T, and outputs this set C_T , ensures that the expected cost $\mathbb{E}_T[\Phi_G(C_T)] \leq \alpha \cdot OPT$.

   (c) Show that if you perform $L := O(\frac{\log n}{\epsilon})$ independent runs of the above algorithm to get sets $C_1, C_2, \cdots, C_L$, and return the set with the least $\Phi_G(C_i)$ value from among these (call it C^*), then $\mathbb{P}r[\Phi_G(C_T) > (1 + \epsilon)\alpha \cdot OPT] \leq 1/\text{poly}(n)$.

   (d) Show that the expected weight of a low-stretch tree is at most $O(\alpha)$ times the MST.

   (e) Extend your algorithm from (a) to solve k-point-facility on an edge-weighted tree.
   (Hint: first solve it on a binary tree. Then show how to reduce the problem to general trees)

3. (*20 points*) Given an undirected graph $G = (V, E)$ with edge weights w_e, a subgraph H is a γ-*distance emulator* with stretch $\gamma \geq 1$ if for every edge $(x, y) \in E$,

$$d_H(x, y) \leq \gamma \cdot d_G(x, y).$$

   γ-*distance emulator*s are similar in spirit to low stretch spanning trees except that they might have many more edges than a spanning tree.

(a) Show that for all $x, y \in V$, even if (x, y) is not an edge, $d_G(x, y) \leq d_H(x, y) \leq \gamma \cdot d_G(x, y)$.

Clearly, a trivial distance emulator is the graph itself, that is $H = G$, but ideally we want a much sparser H.

(b) *Construction 1.* Sample $t = 4 \log n$ trees $T_1, T_2, \cdots, T_t$ from an α-stretch (randomized) low-stretch *spanning* tree, i.e., a low-stretch tree whose distribution is on spanning trees of the graph with the same edge lengths. Let H be the union of all these edges.

   i. Show that for any fixed edge $(x, y) \in E$,

$$\mathbb{P}r[d_H(x, y)] \geq 2\alpha d_G(x, y)] \leq 2^{-t}.$$

   (Hint: for any single value of i, bound $\mathbb{P}r[d_{T_i}(x, y) \geq 2\alpha d_G(x, y)]$)

   ii. Given that, for any graph, there always exists a low-stretch spanning tree distribution with stretch $\alpha = O(n \log n \log \log n)$. Use this to show that with probability $1 - \frac{1}{n^2}$, the graph H is an $O(\log n \log \log n)$-*distance emulator* with $O(n \log n)$ edges.

(c) *Construction 2.* A simple greedy algorithm approach is following.

   i. Define the *girth* of graph G to be smallest number of edges on any cycle in G. We first show that any graph G with m edges and n nodes, and girth *strictly more than* g must have $m \leq O(n + n^{1+1/\lfloor g/2 \rfloor})$.

      A. The average degree of G is $\overline{d} := \frac{2m}{n}$. Show that there exists a subset $S \subseteq V$ such that the induced subgraph $H := G[S]$ has minimum degree at least $\overline{d}/2$.
      [Hint: drop some low-degree vertices]

      B. For this graph H and any vertex $v \in H$, show that the number of distinct vertices reachable within $\lfloor g/2 \rfloor$ hops from v is at least $(\overline{d}/2 - 1)^{\lfloor g/2 \rfloor}$.

      C. Prove that the number of edges m in the original graph G satisfies $m \leq O(n + n^{1+1/\lfloor g/2 \rfloor})$.

   ii. The algorithm is a variant of Kruskal's algorithm for $\alpha \geq 1$. Consider the edges of G in increasing order of lengths $e_1, e_2, \cdots, e_m$. Initialize $H_0 = \emptyset$. When considering edge $e_i = (x, y) \in E$, if the current distance $d_{H_{i-1}}(x, y) \leq \alpha d_G(x, y)$, then discard e (i.e., set $H_i \leftarrow H_{i-1}$), else take it (i.e., set $H_i \leftarrow H_{i-1} \cup \{e_i\}$).

      A. Show that if we set $\alpha = n - 1$, then you will get Kruskal's algorithm. Also, observe that by construction, the graph H at the end of the process is an $(n - 1)$-*distance emulator*. (In fact, an $(n - 1)$-stretch spanning tree.)

      B. If we set $\alpha = O(\log n)$, use (c) with $g = \Theta(\log n)$ to show the final graph H is a $O(\log n)$-*distance emulator* with $O(n)$ edges.