The Cut-Matching Game

R. Kyng & M. Probst Gutenberg *Problem Set 8— Monday, April 22th*

The exercises for this week will not count toward your grade, but you are highly encouraged to solve them all. This exercise sheet has exercises related to week 9 and 10. We encourage you to start early so you have time to go through everything.

To get feedback, you must hand in your solutions by 23:59 on May 5th. Both hand-written and LaTeX solutions are acceptable, but we will only attempt to read legible text.

Exercise 1: Proof that $\mathbf{\Pi}^t$ is doubly-stochastic

In chapter 13, we stated that the matrix $\mathbf{\Pi}^t$ is doubly-stochastic. We now ask you to prove it.

Solution. Since each $\boldsymbol{p}_i^t$ is a probability distribution, it is clear that $\sum_i \boldsymbol{p}_i^t = 1$.

We prove that the matrix is left-stochastic by induction on t. For $t = 0$, we have that the matrix $\mathbf{\Pi}^0 = \boldsymbol{I}$ which is trivial. For the step $t \mapsto t+1$, we observe for each $(i,j) \in M_{t+1}$, we have that $\boldsymbol{p}_i^{(t+1)} = \boldsymbol{p}_j^{(t+1)} = \frac{1}{2}(\boldsymbol{p}_i^t + \boldsymbol{p}_j^t)$, we must have for each ℓ, that

$$\boldsymbol{p}_{i\mapsto\ell}^{(t+1)} + \boldsymbol{p}_{j\mapsto\ell}^{(t+1)} = 2 \cdot \frac{1}{2}(\boldsymbol{p}_{i\mapsto\ell}^t + \boldsymbol{p}_{j\mapsto\ell}^t) = \boldsymbol{p}_{i\mapsto\ell}^t + \boldsymbol{p}_{j\mapsto\ell}^t$$

Thus, summing over all rows using the matching M_{t+1}, we get that the column sum is preserved for each component ℓ.

Exercise 2: Run-time Analysis of the Cut-Matching Algorithm

Analyze the run-time of the Cut-Matching algorithm to establish a total run-time of $O(\log^2 n) \cdot T_{max-flow}(G) + \tilde{O}(m)$[1]. You may assume that computing a random vector $\boldsymbol{r}$ takes $O(n)$ time, and you may use without proof that using link-cut trees you can find a path decomposition in $\tilde{O}(m)$ time.

Solution. Each invocation of FindBiPartition$(\cdot)$ can be implemented by finding a random vector $\boldsymbol{r}$ in $O(n)$ time, and then computing $\boldsymbol{u} = \mathbf{\Pi}^t \boldsymbol{r}$ in time $O(t \cdot n)$ time by simulating the random walk along the found edges explicitly. In particular, letting $\boldsymbol{M}_i$ be the operator corresponding to the effect on the random walk by the matching M_i. Then $\mathbf{\Pi}^t = \boldsymbol{M}_t \circ \boldsymbol{M}_{t-1} \circ \cdots \circ \boldsymbol{M}_1$. Now, letting $\boldsymbol{u}_0 = \boldsymbol{r}$, we can compute iteratively $\boldsymbol{u}_i = \boldsymbol{M}_i \boldsymbol{u}_{i-1}$ until we derive $\boldsymbol{u}_t$ (which is clearly equal $\boldsymbol{u}$). Each iteration $\boldsymbol{M}_i \boldsymbol{u}_{i-1}$ takes $O(n)$ time since $\boldsymbol{M}_i$ just averages the values of components i and j for each edge $(i,j) \in M_i$. Finally, finding the partition once $\boldsymbol{u}$ is computed can be done by sorting components of $\boldsymbol{u}$ in $\tilde{O}(n)$ time.

[1] Technically, we run the max flow procedure on graphs G with two additional vertices and n additional edges. You should ignore this subtlety as it will not impact the asymptotics for any max-flow algorithm known.

Finally, each of the $\Theta(\log^2 n)$ iterations of SparsityCertifyOrCut invokes FindBiPartition($\cdot$) once and then constructs the new flow problem in time $O(m)$ and invokes a max-flow procedure. Doing a path decomposition takes $\tilde{O}(m)$ time. This establishes the claimed run-time.

Exercise 3: Bounding the Diameter of an Expander

Let G be a connected, unweighted graph that is a ϕ-expander with regard to conductance. Prove that the diameter of G is $O(\log m/\phi)$. You may find it useful that $\phi \leq 1$ and that $e^x < 1 + x + x^2$ for $x < 1.79$.

Hint: Fix any pair of vertices s, t in G. Then think about running BFS explorations simultaneously from s and t.

Solution. Let $B(s, d)$ be the (closed) ball around s of radius d. Let $E(B(s, d))$ be the internal edge set of $B(s, d)$.

Observe that since G is connected, we have that $|E(B(s, 0))| \geq 1$. Further, for each $d \geq 0$ where $|E(B(s, d))| \leq m/2$, we have by the definition of a ϕ-expander that

$$|E(B(s, d), V \setminus B(s, d))| \geq \phi \cdot |E(B(s, d)|.$$

It follows that $|E(B(s, d + 1)| \geq (1 + \phi)|E(B(s, d)|$. Let d_{max} be the largest integer such that $|E(B(s, d_{max}))| \leq m/2$. It is clear from our discussion above that $d_{max} \leq 2 \log m/\phi$ since otherwise

$$|E(B(s, d_{max}))| > (1 + \phi)^{2 \log m/\phi} \geq (1 + \phi/2 + (\phi/2)^2)^{2 \log m/\phi} \geq e^{\log m} = m$$

which gives a contradiction to the fact that the number of edges in G is m.

Finally, follow the same argument from t. We have that both s, t must have their balls of radius $2 \log m/\phi$ containing more than $m/2$ edges respectively. But this implies that they intersect in an edge. Thus, there is a path from s to t of length at most $4 \log m/\phi$.

Exercise 4: ASZ Vertex Sparsifiers

In this exercise, we consider a very simple way of obtaining a vertex sparsifier due to Andoni, Stein and Zhong. Recall that a vertex sparsifier of $G = (V, E)$ is a graph $H = (V', E', \boldsymbol{w})$ with less vertices than G that preserves some property of G with respect to a terminal set $T \subset V, V'$. In this exercise, the property we aim to preserve is the pairwise distance between vertices in T. Notice that the vertex sparsifier is a weighted graph, whereas we assume $G = (V, E)$ to be unweighted. Similarly, edge sparsifiers of unweighted graphs are often also weighted, for example in the case of cut/spectral sparsification. We assume that all shortest paths in G are unique, and that the maximum degree in G is Δ.

We now describe the construction of H given G and T. First, we add all the terminal vertices in T to V'. Then, we sample every vertex with probability $100 \log n/k$ and add all the sampled vertices to V'.To describe the edge set, we denote with $B_G(v, V')$ the largest ball around v that does not contain any vertex in V'.

We then describe a path collection $\mathcal{P}$. For every vertex u, $v \in B(u, V')$ and edge $(v, x) \in E$ add the path $\pi_G(u, v) \oplus (v, x)$ to $\mathcal{P}$ where $\pi_G(u, v)$ denotes the shortest path in between u and v in G. Finally, we say that the pivot $p(v) \in V'$ of each vertex v is the closest vertex to v in V'. We then let $\tilde{\mathcal{P}} = \{\pi_G(p(u) \oplus P \oplus \pi_G(v, p(v))|P \in \mathcal{P} \text{ is a } u - v \text{ path}\}$ be the set of paths projected to the closest endpoints in V'. Finally, the edge set E' is constructed by adding an edge (u, v) of length $|P|$ for every $u - v$ path in $\tilde{\mathcal{P}}$. This concludes the description of the graph H.

1. Show that $|B(v, V')| \leq k$ with high probability for all v.
2. Argue that the expected number of vertices in V' is in $\tilde{O}(|T| + n/k)$.
3. Show that the size of the set $|\mathcal{P}| \leq O(\Delta \cdot n \cdot k)$ with high probability.
4. Show that for every pair $u, v \in V'$, we have $\text{dist}_G(u, v) \leq \text{dist}_H(u, v) \leq 4\,\text{dist}_G(u, v)$.

Solution.

1. Directly follows from the fact that there are at most n^2 shortest paths between vertices. Therefore, any shortest path (segment) of length k is hit with high probability.
2. Directly follows since every vertex is sampled with probability $\tilde{O}(1/k)$.
3. For every vertex there are up to k vertices in its ball. Each of these are incident to Δ edges.
4. The first inequality directly follows from the construction. Then, consider the shortest path $\pi_G(u, v)$ between the vertices u and v in G. We then construct a sequence as follows: $u = x_1$. Then, we let x_i be the first vertex on the path $\pi_G(u, v)$ that is no longer in the ball $B(x_i, V')$. Finally, we end the sequence with $x_l = v$ for some l. Notice that this means that the shortest path between x_i and x_{i+1} is in $\mathcal{P}$. Consider the path $p(x_1), \ldots, p(x_l)$ in H. The length of this path is then given by $\text{dist}_G(u, v) + 2\sum_{i=1}^{l} \text{dist}(x_i, p(x_i))$. But $\text{dist}(x_i, p(x_i)) \leq \text{dist}(x_{i-1}, x_i)$ for $i = 2, \ldots, l-1$ and $\text{dist}(x_i, p(x_i)) = 0$ for $i = 1, l$. Therefore the claimed approximation follows.

   We have $p(u) = u$ and $p(v) = v$. Then, consider the sequence of pivots $u = p(x_1), p(x_2), \ldots, p(x_l) = v$. We will now argue that this is a short path in H.

Exercise 5: Distance Oracles

Recall the notion of a cluster in the context of Thorup-Zwick distance oracles for a graph $G = (V, E)$. For every vertex w, we let $C(w) = \{v \in V|\,\text{dist}(v, w) < \text{dist}(v, p(v))\}$ i.e. the cluster of w contains all vertices that are closer to w than to their pivot. As previously, we assume that the distances (and thus shortest paths) are unique. Furthermore, assume that the distance $\text{dist}(v, p(v))$ of vertex v to its pivot is known for every $v \in V$.

1. Show that for a vertex w we can compute the cluster $C(w)$ by running a modified version of Dijkstra in time $\tilde{O}(|E(C(w))|)$.

Hint: First show that for a vertex $t \in C(w)$, all the vertices on its shortest path $\pi(t, w)$ to w are also in $C(w)$. Then, show that running the version of Dijkstra that does not relax edges from vertices u for which $\text{dist}(u, p(u)) \leq \text{dist}(w, u)$ finds the whole cluster $C(w)$.

Solution. We first show that for a vertex $t \in C(w)$, all the vertices on its shortest path $\pi(t, w)$ to w are also in $C(w)$. For the sake of contradiction, assume that there is a vertex t' on the path $\pi(t, w)$ such that $\text{dist}(t', p(t')) \leq \text{dist}(w, t')$. We then have $\text{dist}(t, p(t)) \leq \text{dist}(t, w) - \text{dist}(t', w) + \text{dist}(t', p(t')$ by triangle inequality (and because the pivot of a vertex is closer to it than any other pivot). Since we have $\text{dist}(t', w) \geq \text{dist}(t', p(t')$ because t' is not in $C(w)$, we obtain $\text{dist}(t, p(t)) \leq \text{dist}(t, w)$ in contradiction to t being in $C(w)$.

Given the first part of the hint, we conclude that the whole shortest path tree of the cluster routed at w is contained in the cluster, and therefore running Dijkstra while only relaxing vertices that are closer to w than to their pivot suffices.