Advanced Algorithms Deadline: October 17, 2017, 10:15 am

Graded Homework 1

Lecturer: Mohsen Ghaffari

`NOTE: The following exercises provide 14 points. Receiving 10 points counts as a full score on this homework, and the rest will be viewed as a bonus.`

1 Number of Minimum Edge Cuts (1 point)

In the first exercise, we saw that an n-node graph has at most $\binom{n}{2}$ minimum cuts, i.e., minimum-cardinality set of edges whose removal disconnects the graph. Given a number $n \geq 3$, construct a graph that has exactly $\binom{n}{2}$ distinct minimum cuts.

2 Vertex Cuts (3 points)

Consider an arbitrary n-node graph $G = (V, E)$ that is k-vertex-connected, i.e., the graph remains connected after removing any set of k vertices. Define a random subset $S \subseteq V$ by including each node $v \in V$ in the set S with probability $p = \frac{C \log n}{\sqrt{k}}$, independent of other nodes, for a desirably large constant $C > 10$. Prove that with high probability, the subgraph $G[S]$ induced by the sampled set S is connected.

3 Shortcutting Hop Diameter (2 points)

Consider an arbitrary weighted n-node graph $G = (V, E, w)$ where $w : E \to \mathbb{R}^+$ indicates the weight of the edges. For two nodes $s, t \in V$, let $dist(s, t)$ denote the smallest total weight among all paths that connect s and t. Moreover, define the *shortest-path hop-distance* of s and t to be the smallest h such that there is a path with weight $dist(s, t)$ and only h hops, connecting s and t. We define the Shortest-Path Hop-Diameter (SPHD) of the graph to be the maximum shortest-path hop-distance for any pair of vertices s and t.

In general, a graph might have a large SPHD, even if all nodes are within few hops of each other. Working with graphs of smaller SPHD is much easier, in many computational settings. In this problem, we see how we can add a small number of weighted edges to the graph to decrease the SPHD, while preserving pair-wise distances. Prove that there exists a set of $\tilde{O}(n)$ weighted edges, whose addition to the graph G reduces the SPHD to at most $3\sqrt{n}$.

Hint: Think of a node pair s and t whose shortest-path hop-distance is greater than $\sqrt{n}$, and one shortest path between them. This path is a heavy object, in the lingo of what we did in the second lecture, and thus randomly sampling a few nodes should hit it.

4 Max Cover (2 points)

Given a bipartite graph $G = (L \sqcup R, E)$ for an edge-set $E \subseteq L \times R$, devise an approximation algorithm for the objective of finding a subset $S \subseteq L$ such that the size of the neighborhood $N(S)$ of S is maximized, subject to the constraint $|S| \leq k$.

Note: You can achieve a constant approximation. What is the best constant that you get?

5 Vertex Cover via DFS (2 points)

In this problem, we study a different 2-approximation algorithm for minimum cardinality vertex cover. Let $G = (V, E)$ be a graph and let T be a Depth First Search tree of G. Define S to be the set of all non-leaf vertices in the tree T. Prove that

(A) S is a vertex cover of G, and

(B) $|S| \leq 2|OPT|$, where OPT denotes the optimal (i.e., minimum cardinality) vertex cover of G.

6 Parallel Greedy for Hitting Set (4 points)

Given a bipartite graph $G = (L \sqcup R, E)$ for an edge-set $E \subseteq L \times R$. The objective is to find the smallest cardinality set $S \subseteq L$ such that every node in R has a neighbor in S. In the class, we discussed the sequential greedy algorithm for this problem. This algorithm, despite its extreme simplicity, can be prohibitively complex in a number of modern computational settings, mainly because it has to update the graph after every step of adding one element to the hitting set; i.e., it seems inherently sequential.

In this question, we develop an algorithm that builds the set S incrementally, in only polylogarithmic number of independent iterations. Consider an intermediate point of time in the algorithm, and suppose that S is the current hitting set. Consider the remaining graph H, which is the induced subgraph by $(L \setminus S) \sqcup (R \setminus N(S))$, that is, these are L-nodes that are not in S already, and the R-nodes that remain to be hit. Let $\Delta_{H|L}$ denote the maximum degree in the remaining graph H among nodes in the left side L. We call a subset $S' \subseteq (L \setminus S)$ *good* if $\frac{|N(S') \setminus N(S)|}{|S'|} \geq \alpha \cdot \Delta_{H|L}$, for some $\alpha \in (0, 1]$.

(A) Consider an algorithm that repeatedly finds good subsets S', with respect to the remaining graph in that repetition, and updates $S \rightarrow S \cup S'$, until the remaining graph becomes empty. Prove that this algorithm achieves an approximation of $O(\frac{\log n}{\alpha})$.

Now, we discuss how to find good subsets in a manner that exhausts the graph in only a few iterations. Let T be the set of nodes in the left side of the remaining graph H whose degree is at least $\Delta_{H|L}$. Consider the sub-graph H' of H induced by the vertices T and their neighbors. Let $\Delta_{H'|R}$ denote the maximum degree on the right side of H'.

(B) Suppose we mark each node of T with probability $\frac{1}{2\Delta_{H'|R}}$. Prove that, with at least some constant probability, the set S'_T of marked T-vertices satisfies the following two conditions: (1) The set S'_T is a good subset for some constant α, (2) The set S'_T hits at least a constant fraction of the vertices on the right side of H' that have degree at least $\Delta_{H'|R}/2$.

(C) Complete the above ideas to an algorithm that in $O(\log^4 n)$ iterations of randomly marking appropriately chosen subsets in the remaining subgraph, computes an $O(\log n)$ approximation of the minimum-cardinality hitting set.