Advanced topics in TCS

Exercise sheet 6.
Insertion-deletion Graph Streams

Christian Konrad

This exercise sheet addresses a framework for designing insertion-deletion streaming algorithms for various graph problems including Maximum Matching, Minimum Vertex Cover, and Minimum Dominating Set. In exercise 1, we will describe the framework applied to the Maximum Matching problem and prove properties about the framework. We will then extend the framework to other problems in the subsequent exercises. Exercise two is difficult, and exercise three is optional (and very interesting!).

Question 1. Framework applied to Maximum Matching

Denote by V the vertex set of the input with $|V| = n$ and assume that V is known prior to the processing of the insertion-deletion graph stream. Let $\alpha \leq 1$ be a parameter related to the desired approximation guarantee. The algorithm Insertion-Deletion-Match proceeds as follows:

1. **Pre-processing:** Arbitrarily partition V into sets $V_1, V_2, \ldots, V_{n\alpha}$ such that $|V_i| = \frac{1}{\alpha}$, for every i (for simplicity we assume that $n\alpha$ and $\frac{1}{\alpha}$ are integers)
2. **While processing the stream:** For every pair $(i,j) \in \{1,\ldots,n\alpha\}^2$ with $i \leq j$, compute one arbitrary edge with one endpoint in V_i and the other endpoint in V_j if there is one. Denote this edge by e_{ij} and if no such edge exists let $e_{ij} = \perp$.
3. **Post-processing:** Let $F = \{e_{ij} \ : \ (i,j) \in \{1,\ldots,n\alpha\}^2$ with $i \leq j$ and $e_{ij} \neq \perp\}$. Let $M \leftarrow$ Greedy(F) be the matching obtained by running the Greedy matching algorithm on an arbitrary ordering of the edges F and **return** M

Algorithm 1. Insertion-Deletion-Match

1. Argue that step 2 can be implemented using the l_0-samplers discussed in the lecture (slide 6 in lecture 13).
2. How many l_0-samplers overall does the algorithm use?

3. Suppose that our aim is to achieve that none of the l_0-samplers fails with probability at least $1 - \frac{1}{n}$. How do we have to set the individual error probabilities δ in the l_0-samplers in order to achieve this?

4. Use the two previous exercises to bound the space requirements of the resulting algorithm.

5. Prove that the approximation factor of the algorithm is at least $\alpha/2$. One way to do this is to use a charging scheme: First, charge every edge of a fixed maximum matching M^* in G to a sampled edge e_{ij}. Then, if a charged edge e_{ij} is not included in the matching M then transfer the charge of e_{ij} to the edge that prevented e_{ij} from being added to M^*. The result then follows by bounding the maximum charge that an edge of M has received.

Question 2. Extending the Framework to Minimum Vertex Cover (difficult!)

A *vertex cover* in a graph $G = (V, E)$ is a subset of vertices $V' \subseteq V$ such that every edge $e \in E$ has at least one endpoint in V', i.e., if $e = uv$ then either $u \in V'$, $v \in V'$, or both $u, v \in V'$ holds. A *minimum vertex cover* is one of smallest size. Observe that the Minimum Vertex Cover problem is a minimization problem as opposed to Maximum Matching, which is a maximization problem.

Use the previous framework (i.e., the idea of working with vertex groups rather than individual vertices) to obtain an α-approximation (for $\alpha > 1$) algorithm for Minimum Vertex Cover. We say that a vertex cover I is an α-approximation if:

$$|I| \leq \alpha |OPT| \ ,$$

where OPT is a minimum vertex cover. What are the space requirements of your algorithm? Can you prove that it is correct?

Question 3. Extending the Framework to Minimum Dominating Set (optional)

A *dominating set* in a graph $G = (V, E)$ is a subset of vertices $D \subseteq V$ such that every vertex outside D is adjacent to a vertex in D, i.e., for every $v \in V \setminus D$, $\Gamma(v) \cap D \neq \varnothing$, where $\Gamma(v)$ denotes the neighborhood of v. A minimum dominating set is one of smallest size.

Adapt the framework such that an α-approximate dominating set can be computed in space $\tilde{O}(n^2/\alpha)$, where the $\tilde{O}(.)$ notation supresses poly-logarithmic factors.