# 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 $M \leftarrow \textsc{Greedy}(F)$ 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).

   *Solution:*

Let $f \in \{0,1\}^{\binom{n}{2}}$ be the vector described by the dynamic graph stream in the algorithm. In step 2, for each pair $(i,j)$, let $f_{ij}$ be the vector which corresponds to the entries of $f$ that represent the edges with endpoints in both $V_i$ and $V_j$. Observe that processing only the items of the stream whose edges have endpoints in both $V_i$ and $V_j$ simulates a graph stream that describes $f_{ij}$.

Using an $l_0$-sampler on these implicitly defined streams for each $(i,j)$ gives a randomly sampled non-zero entry from $f_{ij}$ to get the desired edge $e_{ij}$, if one exists.

2. How many $l_0$-samplers overall does the algorithm use?

   *Solution:*

   Step 2 requires an $l_0$-sampler for each pair $(i,j) \in \{1,2,...,\alpha n\}^2$ with $i \leq j$. Hence,

   $$\binom{\alpha n}{2} + \alpha n = \frac{\alpha n(\alpha n - 1)}{2} + \alpha n = \frac{\alpha n(\alpha n + 1)}{2} = \Theta((\alpha n)^2)$$

   $l_0$-samplers are used by the algorithm overall.

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 $\delta$ in the $l_0$-samplers in order to achieve this?

   *Solution:*

   Observe that

   $$\Pr[\text{none of the } l_0 \text{ samplers errs}] = 1 - \Pr[\text{at least one } l_0 \text{ sampler errs}] \ .$$

   Therefore, it suffices to find some error probability $\delta$ such that $\Pr[\text{at least one } l_0 \text{ sampler errs}]$ is at most $\frac{1}{n}$. By the union bound,

   $$\Pr[\text{at least one } l_0 \text{ sampler errs}] \leq \sum_{i=1}^{\frac{\alpha n(\alpha n+1)}{2}} \delta = \frac{\alpha n(\alpha n+1)\delta}{2} \leq \frac{1}{n}.$$

   Therefore, we can choose any $\delta$ such that

   $$\delta \leq \frac{2}{\alpha n^2(\alpha n+1)}.$$

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

*Solution:*

(a) The arbitrary partition of $V$ into sets $V_1, \ldots, V_{n\alpha}$ can be stored using $O(n \log(n\alpha))$ bits (indicating for each vertex to which group it belongs).

(b) $\Theta(\alpha^2 n^2)$ $l_0$-samplers are used, each of which requires $O(\log^2 n \log \frac{1}{\delta}) = O(\log^3 n)$ bits of space with error probability $\delta = n^{-5}$. Overall, this step requires $O(\alpha^2 n^2 \log^3 n)$ bits of space.

(c) Running the GREEDY matching algorithm on the edges outputted by the $l_0$-samplers finds the matching $M$. Since $M$ is a subset of the edges $F$, the space required for $M$ is smaller than the space required for storing the $l_0$-samplers.

The algorithm thus requires $O(\alpha^2 n^2 \log^3 n + n \log(\alpha n))$ bits of space.

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.

*Solution:*

As in the statement of the algorithm, let $F$ denote the set of edges $e_{ij}$ and let $M$ be the matching computed by GREEDY on $F$. We will prove that $|M| \geq \frac{\alpha}{2}|M^*|$ with the following charging argument:
Let $e = uv \in M^*$ be an edge with $u \in V_i$ and $v \in V_j$. Then, we charge edge $e$ to the edge $e_{ij}$ (with a charge of "1" unit). Observe that, in doing so, overall we have charged $|M^*|$ units of charge to the edges in $F$. Next, consider the matching $M$ computed by GREEDY. For every charged edge $e_{ij} \in F \setminus M$, we transfer its charge to the at most two edges in $M$ that prevented $e_{ij}$ from being added to $M$. We have now established that a charge of at least $|M^*|$ is located at the edges in $M$.
Last, we bound the maximum charge received by an edge in $M$. Consider thus an edge $e_{ij} \in M$. This edge can only receive charge from optimal edges with endpoints in $V_i$ or $V_j$. Since there are at most $2/\alpha$ such optimal edges, the maximum charge of an edge is therefore $2/\alpha$. Hence:

$$\frac{2}{\alpha}|M| \geq |M^*| \ ,$$

which implies the result.

## 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 $\alpha$-approximation (for $\alpha > 1$) algorithm for Minimum Vertex Cover. We say that a vertex cover $I$ is an $\alpha$-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?

*Solution:*

Please see Section 4.3. of this paper:

`https://arxiv.org/pdf/2005.11116.pdf`

The result in this paper is stated as an $n^\epsilon$-approximation and proves a space bound of $O(n^{2-2\epsilon} \log n)$ for such an approximation. Denoting the approximation factor by $\alpha$ and setting $\alpha = n^\epsilon$, the result implies a space bound of $O(n^2 \log n / \alpha^2)$.

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