COS 330: Great Ideas in Theoretical Computer Science

Fall 2025

Problem Set 6

Module: New Models of Computation

Below is a reminder of key aspects of the PSet:

- The only goal of this PSet is to help you develop your problem-solving skills in preparation for the exams. Your performance on this PSet will not directly contribute to your grade, but will indirectly improve your ability to do well on the exams.
- Because your performance does not directly impact your grade, you may use any resources you like (collaboration, AI, etc.) to help you complete the PSet.
- We strongly suggest taking a serious stab at the PSet alone, to help self-evaluate where you're at. But, we also suggest collaborating with friends, visiting office hours, asking on Ed, and/or using AI tools to help when stuck. Even when able to complete the entire PSet on your own, you may still find any of these methods useful to discuss the PSet afterwards.
- Throughout the PSet, we've included some general tips to help put these into broader context. Exams will not have these, and future PSets may have fewer.
- We strongly suggest treating this like any other PSet, and writing up your solutions as if you were handing them in for a grade. At minimum, we very strongly suggest writing up sufficiently many solutions to discuss with your Coach.

Aligning Expectations

Recall that the 🗎 symbol implies that the following problem is an "exam-style" problem We highly recommend that you write-up a full solution to this problem.

Problem 1 : Online Degree-Bounded Edge Selection 🗎

In this problem, you are given a fixed vertex set V with $|V| = n$. Edges in E arrive one by one in an online fashion: $e_1, e_2, \ldots$, where $e_t = \{u_t, v_t\} \subseteq V$. When edge e_t arrives, you must immediately and irrevocably decide whether to accept or reject it.[1]

[1] To be extra clear, the game proceeds as follows. You know V, and therefore know $|V| = n$. You do not know anything about E, and you do not know $|E|$. Then, you see $e_1 = (u_1, v_1)$, and must decide whether to accept or reject e_1 (and this decision is final forever). Only after making your decision, then you will see $e_2 = (u_2, v_2)$, and must decide whether to accept or reject e_2.

Constraints. At all times, for all $v \in V$, the set of accepted edges must not have more than two edges adjacent to v. Specifically, let E_{ALG} denote the set of all accepted edges. $\deg_{E_{\text{ALG}}}(v)$ denotes the number of edges in E_{ALG} that are adjacent to v. Your algorithm must maintain:

$$\deg_{E_{\text{ALG}}}(v) \leq 2 \quad \text{for all } v \in V.$$

Objective. Maximize the number of accepted edges. That is, maximize $|E_{\text{ALG}}|$.

Throughout this problem, we'll let E_{OPT} be an optimal offline solution: a maximum-size set of edges in E such that every vertex has degree at most 2.

Finally, this problem will consider the following greedy algorithm: when edge $e_t = \{u_t, v_t\}$ arrives, accept it if and only if

$$\deg_{E_{\text{ALG}}}(u_t) \leq 1 \quad \text{and} \quad \deg_{E_{\text{ALG}}}(v_t) \leq 1.$$

Otherwise, reject it. That is, accept edge e if and only if it is feasible to add e to the current E_{ALG} without violating the constraints.

(a) Show that the greedy algorithm is maximal: there does not exist any $e \in E$ such that it is feasible to add e to the final set E_{ALG}. That is, prove that for every edge $e = \{u, v\} \notin E_{\text{ALG}}$, at least one of u or v has degree exactly 2 in the final set E_{ALG}.

(b) Let $\text{Full}(E_{\text{ALG}})$ denote the nodes in V with degree exactly 2 in E_{ALG}. Prove that $|E_{\text{ALG}}| \geq |\text{Full}(E_{\text{ALG}})|$.

(c) Prove that the number of edges in E_{OPT} that the greedy algorithm rejects is at most $2 \cdot |\text{Full}(E_{\text{ALG}})|$. That is, prove that $|E_{\text{OPT}} \setminus E_{\text{ALG}}| \leq 2 \cdot |\text{Full}(E_{\text{ALG}})|$.

(d) Prove that the greedy algorithm is 3-competitive. That is, prove:

$$|E_{\text{ALG}}| \geq \frac{1}{3}|E_{\text{OPT}}|.$$

(e) Prove that no deterministic online algorithm can always achieve a competitive ratio of 1.

That is, for every deterministic online algorithm, design an instance where that algorithm does strictly worse than $|E_{\text{OPT}}|$.

Problem 2 : Streaming with Multi Passes

Consider a stream of n elements $a_1, a_2, \ldots, a_n$, where each element $a_i \in \{0,1\}^\ell$ (i.e., each element is an ℓ-bit string)[2]. The most frequent element (MFE) is an element that appears most often in the stream. For example, if $n = 7$, $\ell = 2$, and the stream is $00, 01, 10, 00, 11, 00, 01$, then the MFE is 00 (which appears 3 times).

You process the stream in passes: in each pass, you see the elements in order $a_1, a_2, \ldots, a_n$, and can maintain any memory between passes. Your goal is to compute the MFE using as few passes and as little memory as possible.

[2]For simplicity, you can assume that $2^\ell \gg n$, say $2^\ell > n^2$.

(a) Design an algorithm that computes the MFE in one pass using $O(n\ell)$ bits of memory.

(b) Design an algorithm that computes the MFE using $O(\ell)$ bits of memory and $O(2^\ell)$ passes through the stream.

(c) Design an algorithm that computes the MFE using $O\left(\frac{n}{p} \cdot \ell\right)$ bits of memory and $2p$ passes through the stream, for any parameter p with $1 \leq p \leq n$. You may assume you have access to a hash function h that maps each ℓ-bit string to elements in $\{1, 2, \ldots, p\}$ uniformly, or in other words for any $i_1, i_2 \in \{1, 2, \ldots, p\}$ we have $|\{x : h(x) = i_1\}| \approx |\{x : h(x) = i_2\}|$.

(Hint: Use the hash function to partition elements into p groups of about n/p elements each.)