# CS7800: Advanced Algorithms. Fall 2017
# Homework 9

Instructor: Jonathan Ullman TA: Albert Cheu

Due Fri December 1 at 11:59pm
(Email to neu.cs7800@gmail.com)

## Homework Guidelines

**Collaboration Policy.** Collaboration on homework problems is permitted, however it will serve you well on the exams if you solve the problems by yourself. If you choose to collaborate, you may discuss the homework with at most 2 other students currently enrolled in the class. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is strictly forbidden.

*You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem.* You must also identify your collaborators. If you did not work with anyone, indicate that on your submission. If asked, you must be able to explain your solution to the instructors.

**Preparing and Submitting Solutions.** You must type your solutions using LaTeX. Please use an 11-pt or larger font. Please submit both the source and PDF files using the naming conventions `lastname_hw9.tex` and `lastname_hw9.pdf`. Your name must be on the first page of the PDF.

**Solution guidelines** For problems that require you to provide an algorithm, you must give the following: (1) a precise description of the algorithm in English and, if helpful, pseudocode, (2) a proof of correctness, (3) an analysis of running time. You may use any facts from class in your analysis and you may use any algorithms from class as subroutines in your solution.

You should be as clear and concise as possible in your write-up of solutions. Communication of technical material is an important skill, so clarity is as important as correctness. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand. Points might be subtracted for solutions that are too long.

**Problem 1** (Sampling in Data Streams)**.**

Imagine you are Amazon and you would like to monitor your transactions to find items that have suddenly become popular. Fortunately for you, you do millions of transactions every day. Unfortunately, with that many transactions, even storing them in one place is impossible.

A *data stream* is a long sequence of items $x_1, x_2, \ldots$ that you can only read once. A data stream algorithm must process each item very quickly, and with too little memory to store the stream of items, or even a significant fraction of the items. Streaming algorithms typically look like this:

```
ProcessStream(stream S):
  Repeat:
    x_i = next item in the stream
    [do something really simple and fast with x_i]
  Until the stream stops
  Output [something interesting]
```

Figure 1: A caricature of a streaming algorithm.

**(a)** One interesting thing to do with a data stream is to sample a uniformly random item. If you knew that the stream had length $n$, you could simply choose a number $i$ randomly from $\{1, \ldots, n\}$ and then output the item $x_i$ when it arrives. Design an algorithm that chooses a uniformly random sample from the stream *without knowing the length of the stream in advance.* Your algorithm should use $O(1)$ time and $O(1)$ space to process each item (assuming each stream element is $O(1)$ size).[1]

**(b)** Assume that the items in the stream are numbers, and we want to find the *median* of the items in the stream. Being the median is a very global property, so it is difficult to find the median without storing a lot of the stream. Using the algorithm from **(a)**, we can obtain a set of $m$ independent and uniformly random samples from the stream (with replacement). Suppose we take the median $x^*$ of these $m$ samples. Prove that there exists a constant $m_0$ such that if $m \geq m_0$, then with probability at least .99, $x^*$ is an .01-*approximate median* of the items in the data stream. By .01-approximate median, we mean that if the stream has length $n$, then there are at least $.49n$ items in the stream smaller than $x^*$ and at least $.49n$ items in the stream larger than $x^*$. You may assume all items in the stream are distinct.

[1] **Hint:** store a single item that will be your sample, after each item arrives, decide whether to keep the same sample or throw it out and replace it with the item that just arrived.

**Problem 2** (Independent Set in Sparse Graphs, 10 points)**.** In class we saw that the independent set problem is *NP*-complete, and in fact it is *NP*-hard to even find an approximately largest independent set. In this problem we will see that, using randomization, we can find reasonably large independent sets in *sparse graphs*. Let $G = (V, E)$ be an undirected, unweighted graph. Suppose that there are very few edges, specifically $|E| \leq 2n$.

**(a)** Suppose we choose a random subset $S \subseteq V$ as follows: independently for every node $v \in V$, include $v \in S$ with probability 1/4. What is the expected size of $S$, $\mathbb{E}[|S|]$?

**(b)** Let $X$ be the number of edges *inside* $S$ (i.e. the number of edges connecting pairs of nodes in $S$). What is the expectation of $X$, $\mathbb{E}[X]$?

**(c)** Suppose you make $S$ into an independent set as follows: for every edge $e$ inside $S$, remove one of its endpoints if it hasn't been done already. The set of remaining nodes $S'$ is an independent set. Show that $\mathbb{E}[|S'|] \geq n/8$.

**(d)** Deduce from part **(c)** that $G$ contains an independent set of size at least $n/8$.

**(e)** Deduce from part **(c)** that $\mathbb{P}[|S'| \geq n/16] \geq 1/15$.[2]

[2] **Hint:** Use Markov's inequality: for any non-negative random variable $X$, $\mathbb{P}[X > t\mathbb{E}[X]] \leq 1/t$.