# CS7800: Advanced Algorithms — Fall 2025
# Homework 6: Due Friday, December 5, 2025

**Jonathan Ullman**

## Assigned Problems (Collected and Graded)

***Problem 1*** Children are constantly getting too big for their clothes and toys as they grow up. But since you never know when a future child, a friend's child, or even a grandchild might want them, you decide to put all the toys in boxes in the attic instead of throwing them out. However, it's difficult to climb the steep ladder to the attic, so you need to make sure none of the boxes are too heavy, yet you still want to use the fewest boxes possible.

You figure this is the sort of problem that algorithms can help with. You decide that you can carry a box as long as it weighs less than $C$ for some $C \geq 0$. You also have $n$ items with weights $w_1, \ldots, w_n \in [0, M]$. You'd like to find a way to allocate the items to $B$ boxes such that no box has total weight more than $C$ and $B$ is as small as possible. Unfortunately this problem turns out to be NP-hard (you don't have to prove this fact), so you decide to try the following greedy approximation algorithm where you consider the items in order, put as many as you can in the first box, then you put as many items as you can in the second box, and so on. More formally,

```
• Let $i = 1$ and $B = 0$
• While $i \leq n$:
    – Let $B = B + 1$ and open box $B$
    – Let $j$ be the largest value such that $w_i + \cdots + w_{i+j} \leq C$
    – Put items $i, \ldots, i + j$ in box $B$ and close box $B$
    – Let $i = i + j + 1$
```

In this question you will prove that this greedy algorithm is a 2-approximation algorithm. That is, if the smallest number of boxes that suffices to fit all $n$ items is $OPT$ then the greedy algorithm uses $B \leq 2 \cdot OPT$ boxes.

***1.1*** Let $f_k$ be the amount of weight that the greedy algorithm puts into box $k$ for $k = 1, \ldots, B$. Prove that, $f_k + f_{k+1} > C$ holds for every $k = 1, \ldots, B - 1$.

***1.2*** Using 2.1, prove that $\sum_{k=1}^{B} f_k \geq \frac{1}{2}C(B - 1)$.

***1.3*** Complete the proof that the greedy algorithm uses $B \leq 2 \cdot OPT$ boxes.

***Problem 2*** In class we saw how to use linear programming to obtain an $O(\ln n)$-approximation to SetCover, but we didn't have the tools from randomized algorithms to complete the analysis, so you will now fill in the details yourselves!

Recall that in SetCover we have a domain $\{1, \ldots, n\}$ and sets $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ and our goal is to find the smallest number of sets whose union covers the whole domain. The first step in our algorithm was to solve the following *fractional set cover linear program*:

$$\begin{aligned}\min_{x_1,\dots,x_m} \quad & \sum_{j=1}^{m} x_i \\ \text{s.t.} \quad & \sum_{j: i\in S_j} x_j \geq 1 \quad \forall i = 1,\dots,n \\ & 0 \leq x_j \leq 1 \quad \forall j = 1,\dots,m\end{aligned}$$

Let $x^\star$ be the optimal solution to the LP. Let $\textsc{Opt}$ be the size of the optimal set cover and $\text{LP}\textsc{Opt} = \sum_{j=1}^{m} x_j^\star$. Note that $\text{LP}\textsc{Opt} \leq \textsc{Opt}$. Thus, our goal will be to round the LP to find a set cover $C$ whose size is

$$|C| \leq O(\ln n)\cdot \text{LP}\textsc{Opt} \leq O(\ln n)\cdot\textsc{Opt}.$$

In class we used the following *randomized rounding algorithm* to obtain a small set cover.

- For $r = 1,\dots,R$ :
  - Let $C_r = \emptyset$
  - For $j = 1,\dots,m$ :
    * Add $S_j$ to $C_r$ with probability $x_j^\star$, independently from all other iterations
- Return $C = C_1 \cup C_2 \cup \cdots \cup C_R$

The next few questions will guide you through proving the following theorem: We can set $R = O(\ln n)$ so that the following are true: (1) the expected size of $C$ is at most $O(\ln n)\cdot\textsc{Opt}$, and (2) the probability $C$ fails to be a cover is at most $1/n$.

***2.1*** Prove that for every round $r \in \{1,\dots,R\}$, $\mathbb{E}(|C_r|) = \text{LP}\textsc{Opt}$.

***2.2*** Prove that $\mathbb{E}(|C|) \leq R\cdot\text{LP}\textsc{Opt}$.

***2.3*** Prove that for every round $r \in \{1,\dots,R\}$, and every element $i \in \{1,\dots,n\}$

$$\mathbb{P}\left(i \text{ is not covered by } C_r\right) \leq 1/e.[1]$$

***2.4*** Prove that for every element $i \in \{1,\dots,n\}$

$$\mathbb{P}\left(i \text{ is not covered by } C\right) \leq 1/e^R.$$

***2.5*** Prove that

$$\mathbb{P}\left(C \text{ is not a cover of } \{1,\dots,n\}\right) \leq n/e^R.$$

***2.6*** Conclude that we can set $R = K\ln n$ for some constant $K$ so that the theorem above is true.

***Problem 3*** In this problem you will build a streaming algorithm for a very simple problem: counting the number of 1's in a string of bits. You are given a stream of bits $x_1,\dots,x_n$ and would like to compute $s = \sum_{i=1}^{n} x_i$. Since the sum can be as large as $n$, computing it exactly requires $\log_2 n$ bits of space. In this problem we will see a simple algorithm that *approximates* the sum with space complexity proportional to $\log\log n$.

Consider the following algorithm that outputs an estimator of $s$, denoted $\tilde{s}$:

[1]**Hint:** You may find it useful to use the fact that $(1-x) \leq e^{-x}$ for every $x \geq 0$.

- Let $X = 0$
- For $i = 1, \dots, n$: if $x_i = 1$ increment $X$ to $X + 1$ with probability $2^{-X}$, otherwise do nothing
- Return $\tilde{s} = 2^X - 1$

First we will analyze the space required to compute $\tilde{s}$. Note that the number of bits required to store $X$ at any given time is $\log_2 X$, so we will try to argue that $\log_2 X$ does not get too big. Then we will show that $\tilde{s}$ is a good estimate of the number of $s$

***3.1*** Prove that $\mathbb{P}(\log_2 X \geq k) \leq s/2^{2^k}$ and conclude that the expected number of bits required to store the value $X$ is $O(\log \log s)$.[2]

***3.2*** Prove by induction that $\mathbb{E}(2^X) = s + 1$, so $\mathbb{E}(\tilde{s}) = s$. You may find it helpful to use the notation $X_t$ to refer to the value of $X$ after seeing the value 1 in the stream $t$ times.

***3.3*** Prove that $Var(2^X) = O(s^2)$ where $Var$ denotes the *variance*.

***3.4*** Prove that when $U, V$ are *independent* random variables $Var(U + V) = Var(U) + Var(V)$.

***3.5*** Suppose we run $\tilde{s}$ in parallel $k$ times on the same inputs *independently* to obtain $\tilde{s}_1, \dots, \tilde{s}_k$, and then return the average $\bar{s}_k = \frac{1}{k} \sum_{j=1}^{k} \tilde{s}_j$. Compute the mean and variance of $\bar{s}$.

***3.6*** Using Chebyshev's inequality, conclude that for some constant $k$ that does not depend on $s$ or $n$, $\mathbb{P}(|\bar{s}_k - s| > s/100) \leq 1/100$.

Putting these steps together, we have proven that we can estimate $s$ to within $\pm 1\%$ relative error using just $O(\log \log s)$ space in expectation.

[2]**Hint:** You may find the following fact useful: for a non-negative integer random variable $\mathbb{E}(X) = \sum_{i=1}^{\infty} \mathbb{P}(X \geq i)$.