This HW is going out a week before classes start so that you can get a feel for the course asap, and prepare accordingly. It’s a short HW, with a short deadline (end of week #1 of classes), so that we can get to the real material for the course (and HW1) soon.

These problems are solvable using ideas we cover in the first 15 lectures of our undergraduate Algorithms course, plus a basic probability course. (There is not much here to test your Linear Algebra skills, you could try this problem set from Waterloo to see how you do.) Unless specified otherwise, all algorithms should run in poly-time. We’re not asking you to optimize your runtimes, but in general please do so when possible (and reasonable); as algorithm designers, it’s a good habit to strive for optimality.

Please solve the (non-exercise) problems without collaboration. You may discuss the exercises with others. Submission details (and corrections) will appear on the webpage, please check it regularly.

Exercises

Exercises are for fun and edification, please do not submit.

1. (a) Let \(x_1, x_2, \ldots, x_n\) be a random permutation of the numbers \([n] := \{1, 2, \ldots, n\}\). You scan the numbers from left to right. You have a buffer \(B\) of size \(b\) where \(1 \leq b \leq n\), initially containing \(b\) copies of \(\infty\). When you see \(x_i\), if \(B\) contains a number bigger than \(x_i\), then drop the largest number in \(B\), and add in \(x_i\) to \(B\). What is the expected total number of numbers you add to \(B\) during the scan?

(b) Given a cycle \(C_n\) of length \(n \geq 3\) whose nodes are numbered 0 through \(n - 1\), you start an unbiased random walk at node 0. At each step, when you are at node \(i\), you go to \(i + 1\) with probability \(1/2\), and to \(i - 1\) with probability \(1/2\). (Both numbers are considered modulo \(n\), of course.) You continue this walk until you visit all the nodes. What is the probability that the last node to be visited is node \(n - 1\)?

(c) An airplane in Politiesville has \(n\) seats, and \(n\) passengers assigned to these seats. The first passenger to board gets confused, and sits down at a uniformly random seat. The rest of the passengers do the following: when they board, if their assigned seat is free they sit in it, else (being too polite) they choose a uniformly random empty seat and sit in it. (a) What is the probability that the last person to board sits in their assigned seat? (b) What is the expected number of people who board to find their assigned seat occupied?

(d) How many times do we have to sample from a uniform distribution of numbers between 1 and 10 until we see a 1 conditioned on the fact that all numbers we see up to that point are either 1 or 2? (Hint: the answer is not 2.)

(e) You want to sample uniformly at random from the set of all \(n\)-bit strings that are balanced, i.e., that contain exactly \(n/2\) 0’s and \(n/2\) 1’s. You do the following: take a uniform random sample \(\omega\) from the set of all \(n\)-bit strings. output \(\omega\) if it is balanced, else reject, and sample again. Show that the expected number of times you sample an \(\omega\) is \(O(\sqrt{n})\). (Hint: Stirling’s formula.) You should also try to show the expected number of samples is \(\leq n + 1\), by arguments from first principles.
2. (a) Given a tree $T = (V, E)$ and a set of nodes $X \subseteq V$ that contains all the leaves, prove that the average degree of nodes in $X$ is at most 2.

(b) For a graph $G$, consider two edge-weight functions $w_1$ and $w_2$ such that

$$w_1(e) \leq w_1(e') \iff w_2(e) \leq w_2(e')$$

for all edges $e, e' \in E$. Show that $T$ is an MST wrt $w_1$ iff it is an MST wrt $w_2$. (In other words, only the sorted order of the edges matters for the MST.)

(c) Suppose graph $G$ has integer weights in the range $\{1, \ldots, W\}$, where $W \geq 2$. Let $G_i$ be the edges of weight at most $i$, and $x_i$ be the number of components in $G_i$. Then show that the MST in $G$ has weight exactly $n - W + \sum_{i=1}^{W-1} x_i$.

3. You are given a bipartite graph $G = (U, V, E)$ with $|U| = |V| = n$ and maximum degree $\Delta$. Parts (b) and (c) are slightly more challenging.

(a) Give an algorithm to color the edges of $G$ with $2\Delta$ colors so that the edges incident to every vertex have distinct colors.

(b) Give an algorithm to color the edges of $G$ with $\Delta$ colors so that the edges incident to every vertex have distinct colors. (Hint: matchings.)

(c) Let $\Delta'$ be the smallest power of 2 such that $\Delta' \geq \Delta$. Given an $O(n\Delta)$-time algorithm to color the edges with $\Delta'$ colors. (Hint: can you halve the degree at each step?)

4. Given directed graph $G = (V, E)$, a cycle cover is a collection of node-disjoint cycles that contain each vertex of the graph. Give an efficient algorithm for this problem that either outputs a cycle cover or reports that none exists.

**Problems**

1. **(Better Early Than Late)** Consider $n$ points $x_1 \leq x_2 \leq \ldots \leq x_n$ on the real line $\mathbb{R}$, some positive and some negative. You start at the origin, and traveling back and forth on the line, you visit all these points in some order. The delay of point $x_i$, denoted $D_i$, is the total distance you travel before you reach $x_i$. Your goal is to minimize the sum of delays $\sum_{i=1}^{n} D_i$.

   E.g., if the points are at $x_1 = -10, x_2 = -1, x_3 = 2, x_4 = 9$, visiting them in the order $x_2, x_1, x_3, x_4$ would incur total delay $1 + 10 + (1 + 10 + 2) + (10 + 10 + 9) = 62$, whereas visiting them in the order $x_2, x_3, x_4, x_1$ would incur $1 + (1 + 1 + 2) + (1 + 1 + 9) + (1 + 1 + 9 + 9 + 10) = 46$.

   Give an algorithm to find the order that minimizes the total delay. Your algorithm should run in time $\text{poly}(n)$.

2. **(How Many Elements...)** Let $U$ be a universe of elements, with $|U| = n$. Consider a sequence $S = \{a_1, a_2, \ldots, a_m\}$ of $m$ items, with each $a_i \in U$. These may not be all distinct, so suppose there are $D \leq \min(m, n)$ distinct elements in $S$. For simplicity, assume $D$ is a power of 2, and $D \geq 128$. Define $L = \log_2 m$.

   Define $U^0 = U$, and for each $i \in \{1, \ldots, L\}$, let $U^i$ be obtained by independently picking each item from $U^{i-1}$ with probability $1/2$. Observe that $U^i \subseteq U^{i-1}$. Now $S^i$ is the subsequence of $S$ that retains only the elements in $U^i$, so $S^0 = S$.

   (a) Let $X^i$ be a random variable (r.v.) denoting the number of distinct elements in the subsequence $S^i$. Define the event $\mathcal{E}_i := \{|X^i - \mathbb{E}(X^i)| < \frac{\mathbb{E}(X^i)}{4}\}$. Show that

   $$\Pr[\mathcal{E}_i] \geq 1 - \frac{16}{\mathbb{E}(X^i)}.$$
(b) Let \( i^* \) be the level such that \( E(X^{i^*}) = 128 \). Define the event \( F \) to be \( \mathcal{E}_{i^*-1} \cap \mathcal{E}_{i^*} \cap \mathcal{E}_{i^*+1} \).

Show that \( \mathbb{P}[F] \geq 1/2 \).

(c) Consider the following algorithm: Find any level \( i \) for which \( X^i \in [96, 160) \), and output the estimate \( 2^i \cdot X^i \). (If there are no such levels, output zero.)

Show that this estimate lies in \([3/4, 5/4]D\) with probability at least 1/2.

3. (Spick and Span) The column span of an \( n \times d \) matrix \( A \), with \( n > d \), is the set of vectors \( y \in \mathbb{R}^n \) for which \( y = A \cdot x \), for some \( x \in \mathbb{R}^d \). A vector is a positive vector if all its coordinates are non-negative and at least one is strictly positive. Given matrix \( A \in \mathbb{R}^{n \times d} \), write a polynomial-sized linear program whose solution is some positive vector in \( A \)'s column span, and which is infeasible if there is no such vector.

4. (The Utility Company) You are given a universe \( U \) of \( n \) elements, and a collection of subsets \( S = \{S_1, S_2, \ldots, S_m\} \), with each \( S_i \subseteq U \). A subset \( S \in S \) is covered by set \( V \subseteq U \) if \( S \subseteq V \). The utility of a set \( V \subseteq U \) is defined as

\[
\text{util}(V) := \frac{\text{number of sets in } S \text{ covered by } V}{|V|}.
\]

(a) Use an \( s-t \) min-cut algorithm to find a set \( V \) with largest utility. (Hint: Can you solve the problem when you know the value \( \lambda^* \) of the optimal utility? Then how would you remove this assumption?)

(b) Now consider a variant where given \((U, S)\) and two values \((k, \ell)\), the goal is to find a set \( V \) of size exactly \( k \), that covers at least \( \ell \) subsets. Show that this problem is NP-hard, by reduction from a problem in the collection \{SET COVER, CLIQUE, 3-COLORING\}.