15-850: Advanced Algorithms **CMU, Spring 2023**

HW #0 (out Monday Jan 9, 2023) **Due:** Monday Jan 23, 2023

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 HW with a short deadline (start of week #2 of classes).

These problems are solvable using ideas we cover in the first 15 lectures of our undergraduate Algorithms course, plus basic probability and linear algebra courses. You can find links to resources for these topics on the course webpage. 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 problems without collaboration. Submissions will be via gradescope, and the link will appear on the course webpage and on Piazza. Also, changes, corrections, and clarifications will also appear on Piazza, so please check it regularly.

Please submit solutions to two problems from $\{1, 2, 3\}$ and one from $\{4, 5\}$.

1. **(The Centers of Attraction.)** For a path $P = (V, E)$ with positive edge lengths, define $d_P(u, v)$ to be the length of the subpath between u and v in P according to these edge-lengths. For a set C, define
$$d_P(v, C) := \min_{c \in C} d_P(v, c)$$
to be the distance of v to its closest center in C. Given a path $P = (V, E)$ with n nodes, and an integer $k \in \mathbb{Z}_{\geq 0}$, you want to pick a set C of k "centers" from P such that
$$\textstyle\sum_{v \in V} d_P(v, C)$$
is minimized. Give an algorithm that runs in time poly(k, 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. We'd like to create an algorithm which can estimate D (approximately), without storing all of S. For simplicity, assume D is a power of 2, and $D \geq 128$. Define $L = \log_2 m$.

   Our algorithm uses a set of sub-universes $\{U^0, U^1, \ldots, U^L\}$. 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 $U^i \subseteq U^{i-1}$. We will also define a set of subsequences $\{S^0, S^1, \ldots, S^L\}$. Let S^i be the subsequence of S that retains only the elements in U^i. Note that $S^0 = S$.

   Our algorithm considers the number of distinct elements in each of the subsequences S^i. Let X^i be a random variable (r.v.) denoting the number of distinct elements in subsequence S^i.

   (a) First, we will prove that X^i is likely to take a value near its mean. Define the event $\mathcal{E}_i := \{|X^i - \mathbf{E}(X^i)| < \frac{\mathbf{E}(X^i)}{4}\}$. Show that
   $$\mathbf{Pr}[\mathcal{E}_i] \geq 1 - \tfrac{16}{\mathbf{E}(X^i)}.$$

   (b) Because D is a power of 2 and $D \geq 128$, there must exist some i^* such that $\mathbf{E}(X^{i^*}) = 128$. Define the event $\mathcal{F}$ to be $\mathcal{E}_{i^*-1} \cap \mathcal{E}_{i^*} \cap \mathcal{E}_{i^*+1}$. Show that $\mathbf{Pr}[\mathcal{F}] \geq 1/2$.

(c) Now, we are ready to define our estimation 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 $[\frac{3}{4}D, \frac{5}{4}D]$ with probability at least 1/2.

(A variant of this algorithm can give an estimate in the range $[(1-\varepsilon)D, (1+\varepsilon)D]$, instead of in the range $[(1-1/4)D, (1+1/4)D]$.)

3. **(It's So Nice, We Ran it Twice)** Recall that Dijkstra's algorithm computes the single-source shortest-path (SSSP) correctly for directed graphs with non-negative edge-lengths. For graphs with negative-length edges, we use typically the Bellman-Ford or Floyd-Warshall algorithms. Let us explore what happens if we use Dijkstra's algorithm instead. Assume that the graph does not have negative-length cycles.

(a) Show an example of a graph with negative edge-lengths where Dijkstra's algorithm returns the wrong shortest-path distance from the source s. For your reference, we give Dijkstra's algorithm here.

Algorithm 1: Dijkstra's Algorithm

$D(s) = 0$, $D(v) = \infty$ for all $v \neq s$
run **Dijkstra-Iteration**

Algorithm 2: Dijkstra-Iteration

unmark all nodes
while *not all vertices marked* **do**
    $u \leftarrow$ unmarked vertex with least label $D(u)$
    mark u
    forall *its out-edges* (u, v) **do** $D(v) \leftarrow \min\{D(v), D(u) + \ell(u, v)\}$
end

(b) Now suppose we iterate through Dijkstra's algorithm K times (shown formally as under). Considr any node v such that the shortest-path from s to node v contains at most $K-1$ negative-length edges. Show that the final value of the label $D(v)$ equals the length of this shortest s-v path.

Algorithm 3: K-Fold Dijkstra's Algorithm

$D(s) = 0$, $D(v) = \infty$ for all $v \neq s$
for $i = 1, 2, \ldots, K$ **do**
    run **Dijkstra-Iteration**
end

4. **(A Matter of Degree.)** You are given a directed graph $G = (V, E)$ where $V = [n]$, and two vectors $a, b \in \mathbb{Z}^n_{\geq 0}$. You want to find a subgraph $H = (V, E')$ of G such that vertex i has a_i edges entering it and b_i edges leaving it.

(a) Give an algorithm that finds such a subgraph in polynomial time (or reports that such a graph does not exist).

(b) Now the requirements change, and you need to find a *strongly connected* subgraph H of G with the same constraints. Show that the problem is NP-hard, by reduction from a problem in the collection {Clique, 3-Coloring, Hamilton Cycle}. *(A strongly-connected digraph is one where there exists an x-y path for each pair of vertices x, y.)*

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

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

(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 λ^* of the optimal utility? Then how would you remove this assumption?)

(b) Now consider a variant where given $(U, \mathcal{S})$ and two values (k, ℓ), the goal is to find a set V of size exactly k, that covers at least ℓ subsets. Show that this problem is NP-hard, by reduction from a problem in the collection {Set Cover, Clique, 3-Coloring}.