15-850: Advanced Algorithms **CMU, Spring 2024**
HW #1 (out Monday Jan 29, 2024) **Due:** Friday Feb 9, 2024

Some problems may have sub-parts that are exercises, do not submit their answers. For this problem set, it's OK to work with others. (Groups of 2, maybe 3 if you really must.) That being said, please think over the problems yourself before you talk to others. *Please cite all sources you use, and people you work with.*

Please submit a solution to one of $\{1, 2\}$ and one of $\{3, 4\}$. If you submit solutions to more problems, please indicate which solutions you would like us to grade, else we will grade the first one among $\{1, 2\}$ and the first one among $\{3, 4\}$.

---

1. **(Generalizing MSTs to Matroids and the Greedy Algorithm)** Given a finite universe U of elements, a collection of subsets $\mathcal{I} \subseteq 2^U$ is *subset-closed* if $B \in \mathcal{I}$ and $A \subseteq B \subseteq U$ implies $A \in \mathcal{I}$. A set system $(U, \mathcal{I})$ is called an *independence system* if **(I1)** $\varnothing \in \mathcal{I}$, and **(I2)** $\mathcal{I}$ is subset-closed. (We call a set A *independent* if $A \in \mathcal{I}$.)

   A *matroid* is an independence system that satisfies the additional property:

   (I3) if $A, B \in \mathcal{I}$ and $|A| < |B|$ there is an element $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$.

   This means any (inclusion-wise) maximal independent set is also a maximum-cardinality independent set. The *rank* of a matroid is the cardinality of any maximal (and hence *maximum*) independent set in the matroid.

   (a) (Do not hand in) Show that the following set systems are matroids:

      i. *r-uniform matroid:* $\mathcal{I}$ contains all subsets of U with cardinality at most r.

      ii. *partition matroid:* U is the disjoint union of sets $U_1, U_2, \ldots, U_\ell$ (denoted $U = U_1 \uplus U_2 \uplus \ldots \uplus U_\ell$), and $\mathcal{I} := \left\{ A \subseteq U \,\middle|\, |A \cap U_i| \leq r_i \,\forall i \in [\ell] \right\}$,

      iii. *graphic matroid:* U is the set of edges of graph $G = (V, E)$ and $\mathcal{I}$ contains all acyclic subsets of edges (i.e., forests),

      iv. *linear matroid:* $U = \mathbb{F}^r$ for some field $\mathbb{F}$, and $\mathcal{I}$ contains every set of vectors in U that is linearly independent over the field $\mathbb{F}$.

      Show that the ranks of these matroids are r, $\sum_i r_i$, $|V| - 1$, and r respectively.

   (b) (Do not hand in) Verify that the following independence systems do not form matroids in general (by giving counterexamples):

      i. U is the set of edges of an undirected bipartite graph $G = (V, E)$, and $\mathcal{I}$ contains all matchings in G (i.e., subsets of edges such that no two edges share a common endpoint).

      ii. U is the set of arcs of a *directed* graph $G = (V, A)$ and $\mathcal{I}$ contains all subsets of arcs that do not contain a directed cycle.

   (c) (Do not hand in) An (inclusion-wise) maximal independent set is called a *base*. Show that the cardinality of any two bases in a matroid is the same.

   Two last definitions: a *cycle* (or *circuit*) of the matroid is a minimal *dependent* set. A *cut* is a minimal set that intersects every base.[1] Verify that for graphic matroids, these notions correspond to the natural graph-theoretic notions of cycles and cuts.

[1]By minimal set with some property, we mean inclusion-wise minimal, i.e., no strict subset has the property.

Given element weights $w : U \to \mathbb{R}$, the *min-weight base* problem seeks to find a base $B \in \mathcal{I}$ of minimum total weight $w(B) := \sum_{e \in B} w(e)$. (Assume distinct edge weights.)

(d) Show that there is a unique min-weight base.

(e) Prove the *cut rule* (pick a cut and color the lightest edge in the cut blue) and the *cycle rule* (pick any cycle and color the heaviest edge on the cycle red) for matroids. I.e., applying these rules ensures that blue edges belong to the min-weight base, and the red edges do not belong to it. (Hint: think about the graphic matroid case, and how to generalize it.) (Hint: show that if a cut K and a cycle C intersect, they must intersect in at least two elements.)

(f) Consider the natural generalization of Kruskal's algorithm:

> Let $S \leftarrow \varnothing$. Consider elements in U in increasing order of weight. When considering an element e, add it to S if $S \cup \{e\}$ is independent.

Prove this solves the min-weight base problem. (You may assume an oracle that given $S \subseteq U$ answers the membership query "is $S \in \mathcal{I}$?".)

(g) Show that for an independence system $(U, \mathcal{I})$, if the greedy algorithm correctly solves the min-weight base problem for all settings of weights, then the independence system satisfies **(I3)** and hence is a matroid.

Hence, in a very specific sense, the greedy algorithm and matroids are equivalent.

2. **Triangle Self-Reduction.** *This problem is more challenging than #1 to solve, but requires less writing.* Recall the definition of a *subcubic* algorithm: it runs in time $O(n^{3-\epsilon})$ for some constant $\epsilon > 0$. In class, we showed that the All-Pairs Shortest Paths (APSP) problem admits a subcubic algorithm if and only if the Negative Weight Triangle problem admits a subcubic algorithm. In this exercise, we augment the equivalence to the Negative Weight Triangle *Detection* problem, a seemingly even easier problem with a binary (yes/no) output.

   In particular, this exercise asks you to show subcubic equivalence between the following two problems:

   (a) Negative-Weight Triangle: given a weighted, directed graph with weights in the range $[-n^{O(1)}, n^{O(1)}]$, output three arcs that form a negative-weight triangle, or conclude that a negative-weight triangle does not exist.

   (b) Negative-Weight Triangle Detection: given a weighted, directed graph with weights in the range $[-n^{O(1)}, n^{O(1)}]$, output YES if there is a negative-weight triangle, and NO otherwise.

   Clearly, an algorithm that solves Negative-Weight Triangle can also solve Negative-Weight Triangle Detection. Your goal is to show the converse: if there is a subcubic algorithm for negative-weight triangle detection, then there is a subcubic algorithm for negative-weight triangle. (Hint: split the vertex set into roughly equal parts.)

3. **(Yao's $O(m \log\log n)$-time MST Algorithm)** The idea behind this algorithm: *in Boruvka's algorithm we scan all the edges in the graph in each pass, and we should avoid this repetition.* Assume that G is a connected simple graph, and edge weights are distinct.

   (a) Suppose for each vertex, the edges adjacent to that vertex are stored in increasing order of weights. Show a slight variant of Boruvka's algorithm with runtime $O(m + n \log n)$.

(b) (*k-partial sorting*) Given a parameter k and a list of N numbers, give an $O(N \log k)$-time algorithm that partitions this list into k groups $g_1, g_2, \dots, g_k$ each of size at most $\lceil N/k \rceil$, so that all elements in g_i are smaller than those in g_{i+1}, for each i.

(c) Adapt your algorithm from part (a) to handle the case where the edges adjacent to each vertex are not completely sorted but only k-partially-sorted. Ideally, your run-time should be $O(m + \frac{m}{k} \log n + n \log n)$.

(d) Use the two parts above (setting $k = \log n$), preceded by some additional rounds of Boruvka, to give an $O(m \log \log n)$-time MST algorithm.

4. **Short on Average.** Given a directed graph $G = (V, A)$ with possibly negative edge-weights $\{w_a\}_{a \in A}$, define the *weight-ratio* of a directed cycle C to be

$$\rho(C) = \frac{\sum_{a \in C} w_a}{|C|}.$$

We want to find a cycle C with minimum weight-ratio. Denote this by $\rho^*(G) := \min_{\text{cycles } C} \rho(C)$. (You may assume that edge-weights are integers in the range $[-M, M]$.)

(a) (Nothing to submit.) Observe that you can check if a graph has a negative-weight cycle, using Bellman-Ford-Moore. (If all vertices are reachable from s this is immediate, else what will you do?)

(b) Show how to use Bellman-Ford-Moore to find a cycle of zero weight assuming there is no negative-weight cycle.

(c) Observe:

$$\rho^*(G) = \max\{\alpha \in \mathbb{Q} \mid G \text{ with arc weights } (w_a - \alpha) \text{ has no negative cycle } \}.$$

Use this observation to compute $\rho^*(G)$, and also find a cycle C of this weight ratio. Your algorithm should run in time $O(mn(\log M + \log n))$ time.

Caveat: binary search over a range of K integers takes $O(\log K)$ time. But binary search over rationals or reals may not terminate. Can we stop the binary search once it reaches a certain precision, and recover the minimum weight-ratio cycle anyway?