1. Minimum Cost Branching

(a) Directed, weighted graph.
(b) Branching out of \( r \): tree rooted at \( r \), edges directed from parent to child.
(c) Equivalent to: each vertex except \( r \) has exactly one edge pointing to it, undirected versions of edges do not form cycles.
(d) Goal: weights on all edges \( u \rightarrow v \), minimum weight branching.
(e) Each vertex \( u \neq r \) pick minimum weight edge into it. \( \min_v w(v \rightarrow u) \).
(f) If these edges form a tree, we’re done.
(g) Otherwise, there is cycle.
(h) Claim: we use at most 1 edge not on this cycle.
(i) Proof: assume otherwise.
   i. Let this cycle be \( v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_l \rightarrow v_1 \).
   ii. Let \( v_i \) be the vertex furthest from \( r \) such that \( v_{i-1} \rightarrow v_i \) is not in the tree.
   iii. \( v_i \) cannot have path to \( v_{i-1} \), since otherwise the vertex before \( v_{i-1} \) without its predecessor on the cycle is further from \( r \). (this predecessor can’t be \( v_i \) by assumption of \( \geq 2 \) edges missing from the cycle.
   iv. Replacing edge into \( v_i \) with \( v_{i-1} \rightarrow v_i \) leads to tree with no worse weight.
(j) Can view cycle as single vertex.
(k) If we use \( u \rightarrow v_i \) as only edge into cycle, can’t use \( v_{i-1}v_i \). Total cost \( \sum_i w(v_j \rightarrow v_{j+1} - w(v_{i-1} \rightarrow v_i) \).
(l) First term constant, second term can incorporate into weight of \( u \rightarrow v_i \).
(m) Replace \( w(u \rightarrow v_i) \) with \( w(u \rightarrow v_i) - w(v_{i-1} \rightarrow v_{i+1}) \).

2. Speedup

(a) Goal: \( \tilde{O}(m) \) time algorithm.
(b) Can view adjusting weights as assigning weights \( y_v \) to all vertices \( v \neq r \).
(c) Need to detect cycles, and maintain minimum weight edge into a vertex.
(d) Merging can be handled using union-find.
(e) Start from some vertex, follow the minimum weight edge until either get into \( r \), or cycle.
(f) Later case: shrink cycle, can keep on walking.
(g) Need to merge edge sets of vertices along some cycle.
(h) Simple algorithm using binary heaps: insert all elements from smaller set into bigger. \( O(\log n) \) overhead.
(i) Mergeable heaps useful here. Need extra tag indicating modifications done via. \( y_u \), aka. \(-w(v_{i-1}, v_i)\).
3. Matroid Intersection

(a) Informally: set of subsets of S where greedy works very well.

(b) Examples:
   i. Partition matroid: each partition \( S = S_1 \ldots S_n \), one element from each.
   ii. Edges have no cycles, aka. spanning tree.

(c) Minimum cost branching: intersecting partition matroid with spanning tree matroid.

(d) Bipartite matching: intersecting partition matroid with partition matroid.

(e) Can also intersect spanning tree matroid with spanning tree matroid.

(f) Will use trees as representative for general matroids, algorithm here is readily extendable.

(g) Two edge lists for two graphs \( G \) and \( H \), want maximum subset \( I \) so that \( e_I, f_I \) are both forests.

(h) Let edges be \( e_1 \ldots e_m, f_1 \ldots f_m \). Want \( I = \{i_1 \ldots i_{n-1}\} \) such that \( e_I \) and \( f_I \) are both spanning trees.

(i) For a subset \( U \), let \( rank(G)(U) \) be maximum number of elements from \( U \) that can be picked so no cycles exist in \( G \) (and define \( rank(H)(U) \) similarly).

(j) Claim: \( I \) does not exist iff there is a partition of \( S = [1,n] \) into \( U, S \setminus U \) such that \( rank(G)(U) + rank(H)(S \setminus U) < n - 1 \).

(k) One direction easy, can’t pick more than \( rank(G)(U) \) from \( U, rank(H)(U) \) from \( S \setminus U \).

(l) Sanity check: bipartite matching, same as König’s theorem.

(m) Other direction: define bipartite graph on \( S \setminus I \) and \( I \). Call a set ok if it has no cycles.

(n) Edge \( i \rightarrow j \) (\( i \in S \setminus I, j \in I \)) exists if \( f_{i \cap i-j} \) is ok in \( H \).

(o) Define \( X_G = \{i \in S \setminus I, e_{I+i} \text{ is ok in } G\}, X_H = \{i \in S \setminus I, f_{I+i} \text{ is ok in } H\} \)

(p) If path from \( X_G \) to \( X_H \) (they can overlap too), can augment \( I \) (this is more tricky than usual augmenting paths, need to augment along the shortest path).

(q) Let \( U \) be everything reachable from \( X_G \).

(r) Claim: \( rank(G)(U) \leq |I \cap U| \).
   i. Since \( f_I \) has no cycles, \( f_{I \cap U} \) is also ok.
   ii. Suppose we can add 1 more edge, \( f \in U \setminus I \) to \( I \cap U \) so \( f_{I \cap U+i} \) is ok in \( H \).
   iii. Since \( U \cap X_H = \emptyset, f_{I+i} \) has a cycle.
   iv. Since \( f_I \) is ok and \( f_{I \cap U+i} \) is also ok, exist \( j \in I \setminus U \) such that \( f_{I+i+j} \) is ok. (Cycle formed by adding \( i \) can’t all be in \( I \))
   v. \( i \in I \cap U \), so \( j \) should be in \( U \) as well. Contradiction with choice of \( U \).

(s) Claim: \( rank(G)(S \setminus U) \leq |I \setminus U| \).
   i. Since \( e_I \) has no cycles, \( e_{I \setminus U} \) is also ok.
   ii. Suppose \( e_{I \setminus U+i} \) is ok in \( G \) for some \( i \in (S \setminus U) \setminus (I \setminus U) \).
   iii. Note \((S \setminus U) \setminus (I \setminus U) = (S \setminus I) \setminus U \).
   iv. Since \( X_G \subset U, e_{I+i} \) has a cycle.
v. Since $e_I$ is ok and $e_I \setminus U^+_i$ is also ok, exist $j \in U$ such that $e_{I^+ i-j}$ is ok.
vi. $i$ is reachable as well, contradiction with choice of $U$.

(v) Adding gives overall bound.
(w) Intersecting three matroids is NP hard.