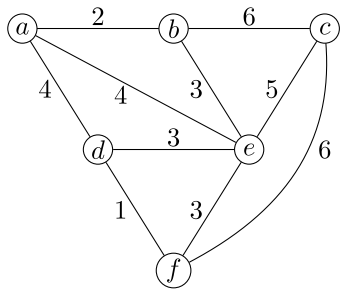
Problem 1 (20 points). Run Kruskal's algorithm on the following undirected graph: give the order of edges that are added to the MST (whenever you have a choice, always choose the smallest edge in lexicographic order); for each edge added, give a cut (i.e., the certificate) that justifies its addition does not break optimality.

Solution. The order of edges that are added to the MST is as below (followed by the certificate cut):
(d,f); one possible certificate cut is $(\{d\}, V \setminus \{d\})$
(a,b); one possible certificate cut is $(\{a\}, V \setminus \{a\})$
(b,e); one possible certificate cut is $(\{a,b\}, V \setminus \{a,b\})$
(d,e); one possible certificate cut is $(\{a,b,e\}, V \setminus \{a,b,e\})$
(e,c); one possible certificate cut is $(\{c\}, V \setminus \{c\})$.



Problem 2 (20 points). Run Prim's algorithm on the above undirected graph: give the order of vertices that are added to the MST (whenever you have a choice, always choose the smallest vertex in alphabetic order); before adding each vertex to the MST, give the *key* (i.e., priority) value for all vertices in the priority queue.

Solution. The order of vertices that are added to the MST is a,b,e,d,f,c (see table below; each row gives the key of each element in priority queue and the corresponding *prev*).

Set S	a	b	c	d	e	f
{}	0/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
a		2/a	∞/nil	4/a	4/a	∞/nil
a,b			6/b	4/a	3/b	∞/nil
a,b,e			5/e	3/e		3/e
a,b,e,d			5/e			1/d
a,b,e,d,f			5/e			
a,b,e,d,f,c						

Problem 3 (20 points). Design an efficient algorithm for the *maximum spanning tree* problem, i.e., given an undirected graph $G = (V,E)$ with edge weight $w(e)$ for any $e \in E$, to compute a spanning tree T of G such that $\sum_{e \in T} w(e)$ is maximized.

Solution: Use Kruskal's / Prim's algorithm, instead of increasing order use decreasing order

Example: Using Kruskal's Algorithm

for all u ∈ V :
    makeset(u)

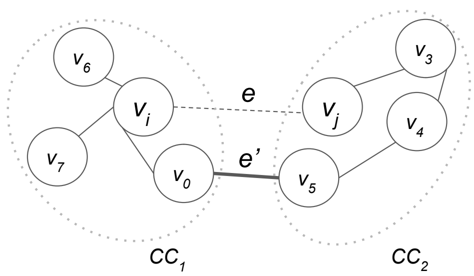
X = {}
Sort the edges E by weight for all edges $\{u,v\}$ ∈ E, in **decreasing order** of weight:

if find(u) $\neq$ find(v):
    add edge $\{u, v\}$ to X
    union(u; v)

Option 2: Multiple each edge by (-1) and use Kruskal's/ Prim's algorithm. (*think about how to prove this is correct.*).

Problem 4 (20 points). Give a counter-example or prove the following statement: Let $G = (V, E)$ be an undirected graph. Let C be one cycle in G and let e be an edge in C. If the weight of e is strictly larger than any other edge in C, then e is not in any minimum spanning tree of G.

Solution. The statement is true. We will prove this by contradiction. Suppose, we have a MST T of graph G that contains the edge $e = (v_i, v_j)$. If we remove the edge e from the tree T, we get two separate non-empty connected components CC_1 and CC_2. Let $v_i \in CC_1$ and $v_j \in CC_2$. A cycle $C = (v_0, v_1, \ldots, v_{k-1}, v_k, v_0)$ containing edge $e = (v_i, v_j)$, can be written as the concatenation of two paths: $P_1 = (v_i, v_j)$, and $P_2 = (v_j, v_{j+1}, \ldots, v_i)$. After removing edge e, it is possible to connect CC_1 and CC_2 by adding some other edge $e' = (u, v)$ from P_2 such that $u \in CC_1$, and $v \in CC_2$. Note that, $e' \notin T$. *(If both e and e' were in T, then that would have created a cycle, but a MST cannot have cycle.)* Let, T' be the tree obtained from T by removing e and adding e'. According to the question, $l(e') < l(e)$ for all edge $e' \in C$ where $e' \neq e$. Therefore T' is a spanning tree with smaller weight than T. This proves that T cannot be a MST.



Problem 5 (20 points). Give a counter-example or prove the following statement: Let $G = (V, E)$ be an undirected graph with edge weight $w(e)$ for any $e \in E$. Let T be an MST of G. Let X be a connected subgraph of G. Then $T \cap X$ is contained in some MST of X.

Solution. The statement is true. Let $T \cap X = \{e_1, e_2, \cdots, e_k\}$. Suppose for $1 \leq i \leq k$, $P = \{e_1, e_2, \cdots, e_i\}$ is contained in some MST of X. Now we prove that $P \cup \{e_{i+1}\}$ is also in some MST of X. Removing edge e_{i+1} from T divides T in two parts giving a cut $(S, G \backslash S)$ and a corresponding cut $(S_X, X \backslash S_X)$ of X with $S_X = S \cap X$. Now, e_{i+1} is the lightest edge in G (and hence also in X) crossing the cut, otherwise we can include the lightest edge and remove e_{i+1} to get a better spanning tree for G. No other edges in T, and hence in P, crosses the cut. We can then apply the cut property to get that $P \cup e_{i+1}$ must be contained in some MST of X. Continuing in this manner we will conclude that $T \cap X = \{e_1, e_2, \cdots, e_k\}$ must be contained in some MST of X.