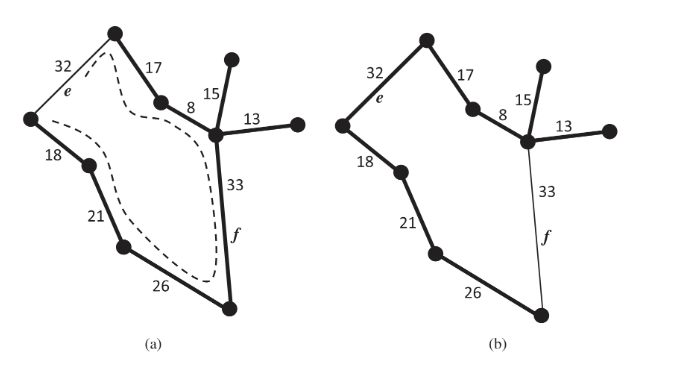
- Minimum Spanning Tree (MST)

-- define MST

Given a weighted undirected connected Graph G = (V, E), computing a spanning tree with a smallest total weight is the problem of Minimum spanning tree (MST).

-- give example of

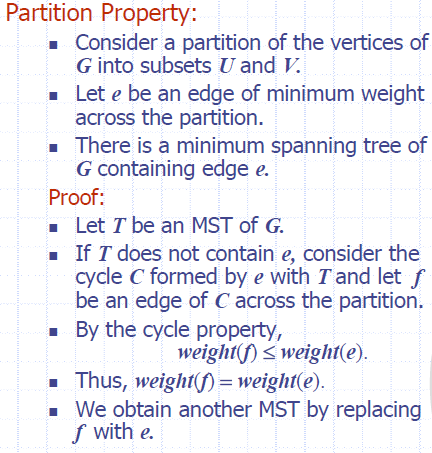
-- properties of MST



**G is a weighted undirected connected Graph, and T is its MST. If e is an edge of G that is not in T, then the weight of e is at least great as any edge in the circle created by adding e to T.**

For example, f=33 is larger than 8, 17, 32, 18, 21, 26 in this circle.

**If the weight in G is distinct, then the MST is unique.**

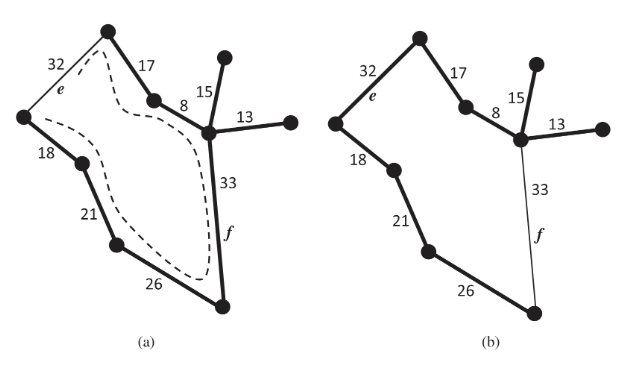


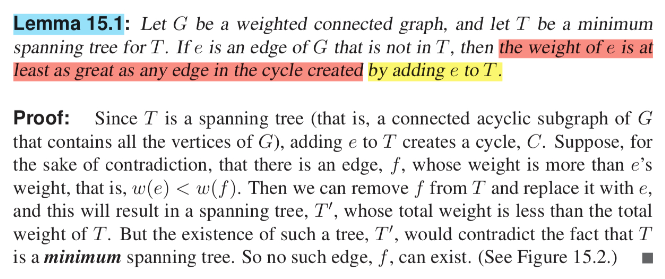
**Kruskal’s MST algorithm still work correctly even if the graph has negative-weight edges, and even negative-weight cycles**

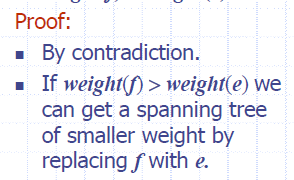
-- review proofs from assignment relating to trees, cycles, MST

-- what is meant by an optimality condition - logical facts or post condition that must hold for any algo that solves MST

-- given the Lemma Cycle Optimality Condition, prove it by contradiction and exchange argument







-- Kruskel's algorithm

-- give code

-- prove correctness and runtime

-- given instance of the problem give the order EDGES are added (when asked for an edge you must give 2 endpoints e.g. (u,v) )

Algorithm KruskalMST(G):

Input: A simple weighted connected undirected Graph G, with n vertices and m edges.

Output: T, it is G's MST.

for each vertex v in G do --------------------------------------O(n)

Define vertex cluster C(v), C(v) <- {v}-----------------

Create a PQ named Q to store all edges in G, using edge's weight as key. -----------O(log m)

T <- 空集 // use T to store edges of MST then

while T has fewer than n - 1 edges do ---------worst case m times

edge (u,v) <- Q.removeMin() -------------------------O(log m)

Let C(u) be the cluster containing vertex u

Let C(v) be the cluster containing vertex v

if C(u) != C(v) then --------------执行n-1次 ~ <= n次

Add edge (u,v) into T

Union C(u) and C(v)--------------------------所以合并总过程union一次为平均log n, total merging time: n log n

return T

runtime: n + log m + m log m + n log n <= (m + n) log n, because m<=n(n-1)/2, then runtime <= m log n.