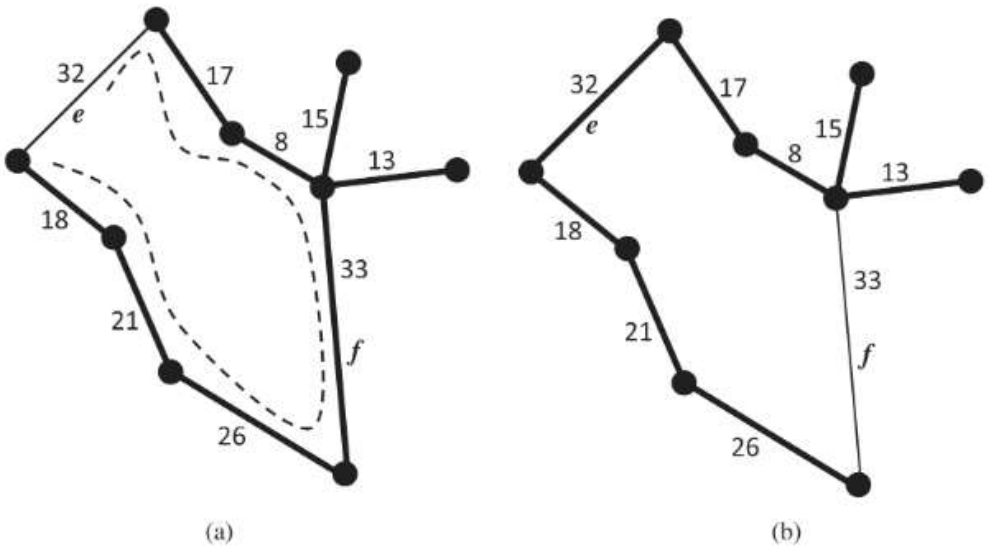
MST: Minimum Spanning Tree

1 Define:

Given a simple undirected connected weighted Graph G, with n vertices and m edges, computing the spanning tree with total smallest edge weight is the MST minimum spanning tree.

2 Properties of MST

2.1 circle property



Given a simple weight connected undirected weight, if an edge e is not in the T (edge cluster of MST), then the weight of e is at least as greater than the weight of any edge in the circle created by adding e into T.

Prove by contradiction: **Exchange Arguement**

Assume the weight of e is smaller than an edge f's weight in the circle, and f is in T. And then we can replace f with e, and create a new Spanning Tree T'. And weight(T') will be smaller than weight(T), so T' will be the MST in G rather than T. It will contradict the fact T is MST. So, no such edge f exists.

2.2 partition property (cut partition)

Given a partition of the vertices of G into subsets U and V, let e be an edge of minimum weight across the partition, and then there is a MST of G containing edge e.

Prove by contradiction:

Assume T (MST of G) contains other edge f rather than e, and weight f > weight e. And

then we can create a new spanning tree T' through replacing f with e in T. Total weight of T' will be smaller than T(MST). It contradicts the fact that T is MST. So, no such edge f exists.

3 Review proofs from assignment relating to trees, cycles, MST

Suppose G is an undirected, connected, weighted graph such that the edges in G have distinct edge weights. Show that the minimum spanning tree for G is unique.

Proof by contradiction:

Given a weighted connected undirected simple Graph G, assume T1 and T2 are 2 MST of G, so Weight(T1) = Weight(T2). There must be at least one edge e contained in T1 rather than T2. If we add e into T2, it will create a circle. f is an edge in this circle and in T2, and e and f have one same node together. If weight e is larger than f, then we can replace e with f, and add f into T1 to create a new spanning tree T1'. Weight T1' will be smaller than T1, thus it contrasts T1 is MST. If weight e < weight f, then we can replace f with e, and add e into T2 to create a new spanning tree T2'. Weight T2' < Weight T2, it also contrasts T2 is MST. So, for this G, MST is unique.

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

circle property

partition property (cut partition)

5 given the Lemma Cycle Optimality Condition, prove it by contradiction and exchange argument circle property

Above

6 ~~Kruskel~~'s algorithm

Algorithm KruskalMST(G):

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

Output: the G's MST named T.

for each vertex in G do

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

Create a PQ named Q to store all edges in G, using the edge weight as key.

T <- 空集 //use T to store the edge of MST

while T has fewer than n-1 edges do

(u, v) <- Q.removeMin()

Let C(u) be the cluster containing v

Let C(v) be the cluster containing u

if C(u) != C(v) then

Add edge (u, v) into T

Union C(v) and C(u)

return T

6.1 prove correctness and runtime

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.

6.2 Given instance of the problem give the order EDGES are added

见 iPad

Shortest Paths

1 Definition

Given a weighted graph and 2 vertices u and v, we want to find a path of minimum total weight between u and v.

2 Compare/contrast with MST

看题意答吧

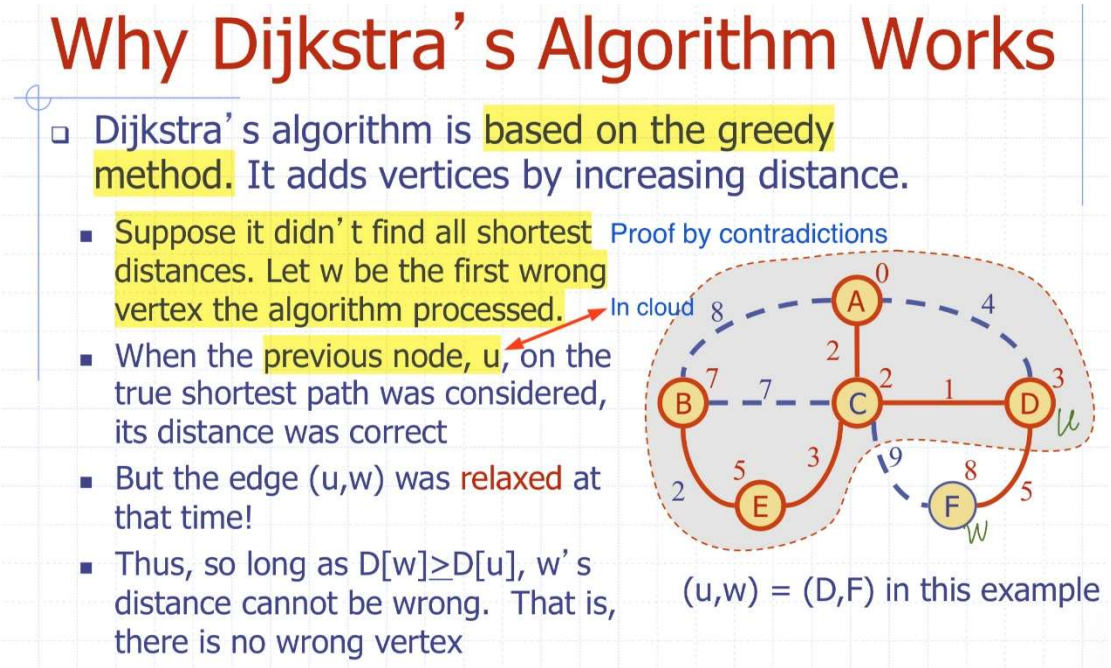
3 Properties of a shortest path, can an edge weight be < 0 for undirected graph? Explain Why It doesn't work for negative-weight edges.

1 Dijkstra's Algorithm is based on the greedy method. It adds vertices by increasing distance. If a node with a negative incident edge were to be added late to the cloud, it could mess up distances for vertices already in the cloud.

2 A sub-path of a shortest path is itself a shortest path.

3 There is a tree of shortest paths from a start vertex to all the other vertex.

4 Proofs relating to shortest path discussed in class (prefix of a shortest path)



5 Dijkstra's algorithm and main idea

Main idea: use 'edge weight' BFS and Greedy Strategy, edge relaxation to update the Distance from start vertex v to other vertex u, and non-negative edge, and set unexplored vertex's D as ∞ and D[v] = 0.

```
Algo DijkstraShortPaths(G, v):
        Input: A weighted undirected Graph G, and a start point v.
        Output: the Distance D from v to other vertex in G.

        D[v] <- 0

        for each vertex u != v in G do
                D[u] <- 无穷

        Create a PQ named Q to store all vertices from v to u, using the D labels as key

        while Q is not empty do
                u <- Q.removeMin()
                for each u's adjacent vertex z do
                        if D[u] + weight(u,z) < D[z] then
                                D[z] <- D[u] + weight(u,z)
                                use D[z] to update vertex z's key in Q
        return D
```

6 Edge relation and importance of

Edge Relation is that:

if D(u) + weight(u,z) < D(z) then

D(z) <- D(u) + weight(u,z)

u is the vertex in the cloud, and z is u's adjacent vertex and not in the cloud.

Edge relation can help update the distance from v to these adjacent vertices, and keep the their Distance D is shortest so far.

Once new vertex is added into the cloud, its adjacent vertex distance will be updated

correctly. Until all vertices are added into the cloud, this updating process will stop. Finally, we will get all shortest paths from start point v to other vertex.

6 Explain why it is safe to remove D[u] from the minimum priority queue before all edges are examined

Solution:

At first, all edges are non-negative, and Dijkstra's Algo is based on Greedy strategy and weight BFS. We will not worry that a new added edge will mess up the cloud and the past shortest paths. And for v is the start vertex D[v] <- 0, it will be removed firstly, and it is correct.

And then for those vertices u are not explored, the Distance from the start point v to u is ∞. When we get ready to use Q.removeMin() to put new vertex into the Cloud, Q.removeMin() will choose vertex with minimal key. rather than choose these vertex u, whose D[u] = ∞.

The core to keep safe to remove D[u], is that edge relaxation can update each adjacent vertex's key.

if D(u) + weight(u,z) < D(z) then

D(z) <- D(u) + weight(u,z)

update z's distance key in Q

Through iterations, this cloud will be more and more big until containing all vertex and returning the shortest paths.

7 Given instance, give the order EDGES are added to the shortest path

一定要看 iPad