Graphs: Minimum Spanning Tree

Greedy Approach

1. Greedy: Take locally optimal move
    - When does that lead to global optimum?
        - Knapsack: Doesn't work
    - Minimum Spanning Tree (MST)
        - Greedy approach does work
        - Prove Kruskal's and Prim's algorithm
        - Cut property

MST Problem

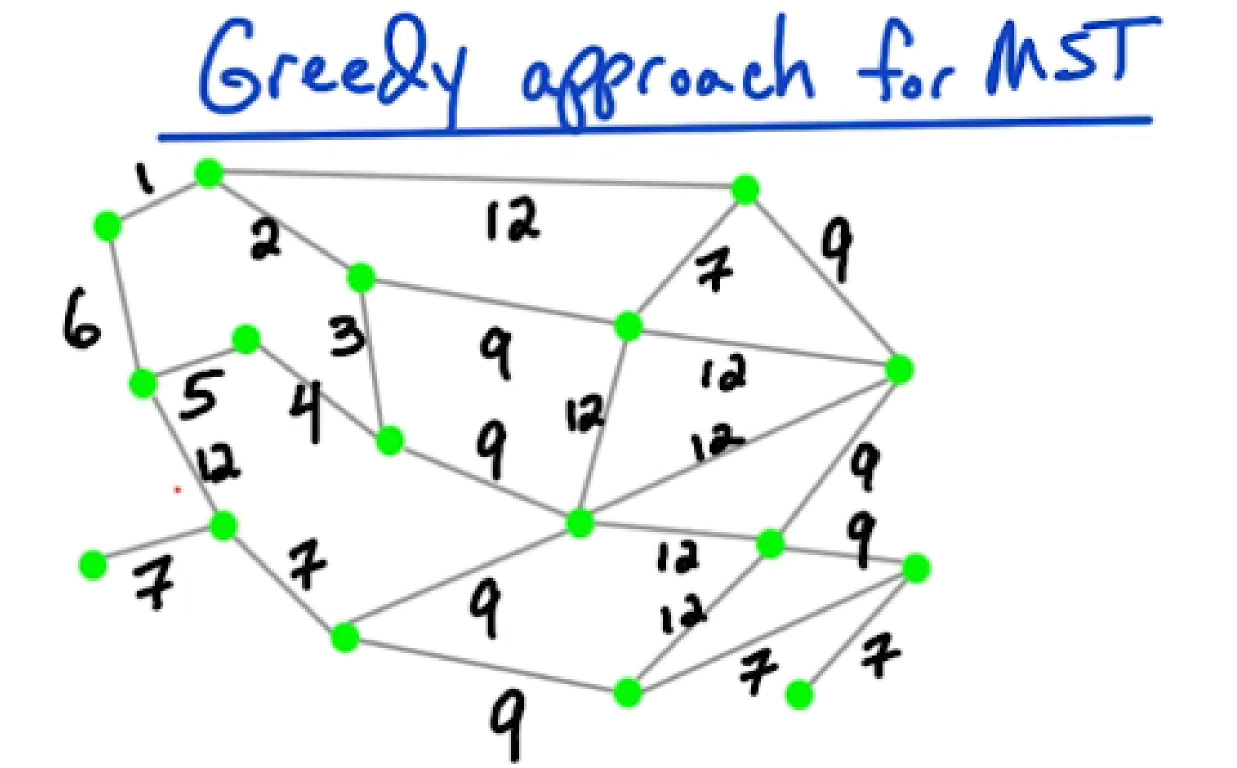
1. Given: Undirected graph G = (V,E) with weights w(e) for e in E
2. Goal: Find minimal size, connected subgraph of minimum weight
    - Find minimum weight spanning tree of G
        - Spanning tree: Minimal size, connected subgraph
    - Minimum weight: For T in E, w(T) = sum(w(e))

Tree Properties

1. Tree = connected, acyclic graph
2. Basic properties:
    - Tree on n vertices has n-1 edges
    - In a tree, exactly one path between every pair of vertices
    - Any connected G = (V,E) with |E| = |V| - 1 is a tree

Greedy Approach for MST

1. Sort edges by increasing weight
    - If an edge doesn't create a cycle, add it to the MST



Tree

Kruskal's Algorithm

```
Kruskals(G):
    input: undirected G=(V,E) with weights w(e)
    Sort E by incrasing weight
    Set X = {}
    For e = (v,w) in E (go through in order):
        if x U e doesn't have a cycle:
            x = x U e
```

Kruskal's Analysis

1. Runtime Analysis
    - Sort: O(mlogn) time
        - m = |E|, n = |V|
    - Let c(v) = component containing v in (v,x)
        - c(w) = component containing w in (w,x)
        - If c(v) != c(w) then add e to x
    - Use Union-find data structure
        - O(logn) time per operation
    - Total runtime: O(mlogn)
        - Steps 1 and 3 (sort and for loop) both take O(mlogn)

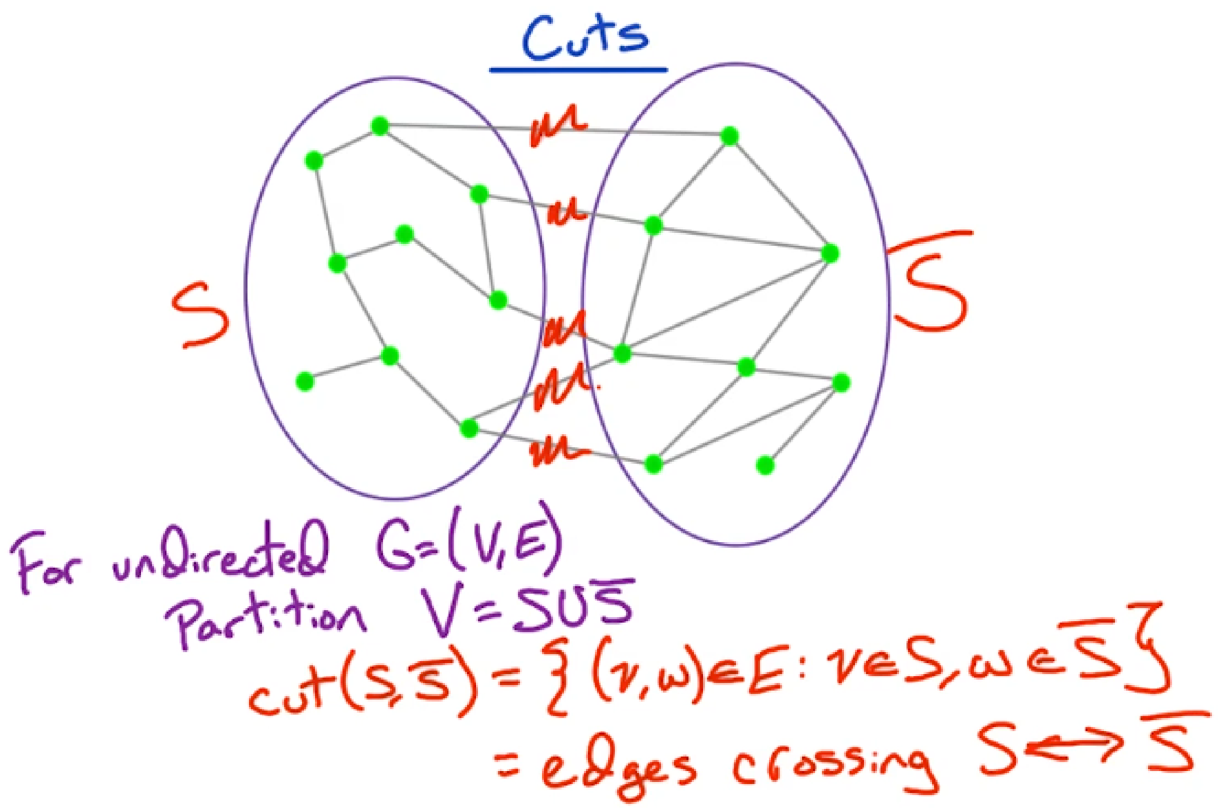
Kruskal's Correctness

1. Proof by induction:
    - Assume X is correct so far

   - X is a subgraph of T where T is a MST
   - Add e=(v,w) to X
   - Claim: X U e is a subgraph of T' where T' is a MST
2. Key Property
   - If we consider one component S=c(v), the set of vertices containing v
   - S' = rest of the vertices
   - Both S and S' are components
     - X doesn't cross S <-> S'
   - e is a minimum weight edge crossing from S to S'

Cuts

1. For undirected G=(V,E)
   - Partition V = S U S'
   - cut(S,S') = {(v,w) in E: v in S, w in S'}
     - = edges crossing S <-> S'



Cut

Cut Property

1. Lemma: For undirected G=(V,E)
   - Take X as a subset of E where X is a subset of T for a MST T
   - Take S as a subset of V where no edge of X is in the cut (S,S')
   - Look at all edges of G in cut(S,S')
   - Let e* be any minimum weight edge in cut(S,S')
   - Then: X U e* is a subset of T' where T' is a MST

Cut Property: Kruskal's

1. The above cut property is used to prove Kruskal's algorithm

Proof Outline

1. Fix G, X, T, S
   - X is a subset of T where T is a MST
   - No edge of X crosses S <-> S'
   - w(e*) <= $w(e_1), \ldots, w(e_4)$
2. Goal: Construct MST T' where:
   - X U e* is a subset of T'

Constructing T

1. Goal: Construct MST T' where:
   - X U e* is a subset of T'
   - If e* is in T: T' = T
   - If e* is not in T
     - Look at T U e*: Has a cycle C
     - Take e' in T crossing S <-> S'
       - This is guaranteed to be greater in weight than e*
     - Set T' = T U e* - e'

T' is a Tree

1. T' = T U e* - e'
   - Show: T' is connected and has n-1 edges
     - n-1 edges: T was a tree, we added one edge and subtracted another
     - Connected: For y,z in V
       - Let P be path y -> z in T
       - Let C be cycle in T U e*
       - Let P' = C - e' is a path c -> d in T'
       - To go y -> z in T': Use P, replace e' by P'

T is a MST

1. T' = T U e* - e'
   - T' is a tree
   - Know w(T) is minimum
     - w(e*) <= w(e')
     - w(T') = w(T) + w(e*) - w(e') <= w(T)
       - w(e*) - w(e') <= 0
2. Key ideas:
   - Statement of cut property: Minimum weight edge across the cut is part of a MST
   - Proof of cut property: I can take a tree T and add an edge into that tree that creates a cycle. I can remove an edge from that cycle and get a new tree

Prim's Algorithm

1. MST algorithm akin to Dijkstra's algorithm
   - Use cut property to prove correctness of Prim's algorithm