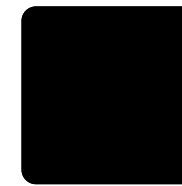
CS3383 Unit 2: Greedy. Lecture 1. Minimum Spanning Trees

David Bremner

February 15, 2024

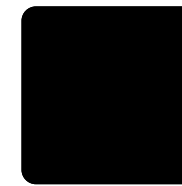
Greedy

- Minimum Spanning Tree
- MST Algorithms

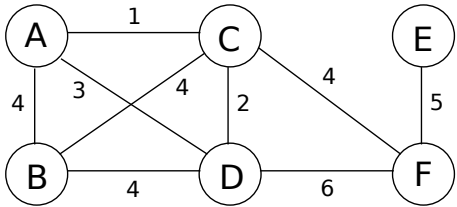


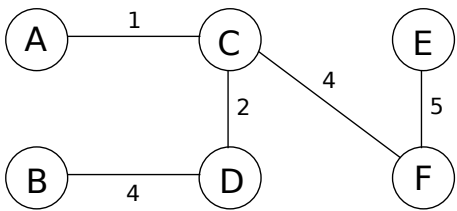
Greedy

- Minimum Spanning Tree
- MST Algorithms



Minimum spanning tree





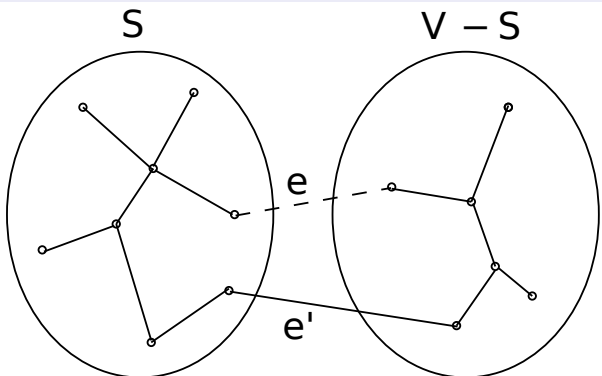
Minimum Spanning Tree

Given $G = (V, E)$, $w : E \to \mathbb{R}$, a *minimum spanning tree* T is a spanning tree (i.e. connecting all vertices) that minimizes $\text{cost}(T) = \sum_{e \in T} w(e)$

Cut Property

Lemma

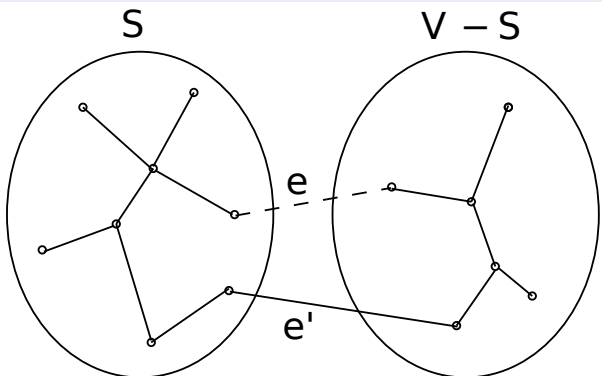
Let T be a minimum spanning tree, $X \subset T$ s.t. X does not connect $(S, V - S)$. Let e be the lightest edge from S to $V - S$. $X \cup e$ is part of some MST.



Cut Property

Lemma

Let T be a minimum spanning tree, $X \subset T$ s.t. X does not connect $(S, V - S)$. Let e be the lightest edge from S to $V - S$. $X \cup e$ is part of some MST.



Greedy Algorithms in General

Discrete Optimization Problems

- solution defined by a sequence of choices
- solutions are ranked from best to worst

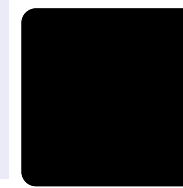
Greedy Algorithms in General

Discrete Optimization Problems

- solution defined by a sequence of choices
- solutions are ranked from best to worst

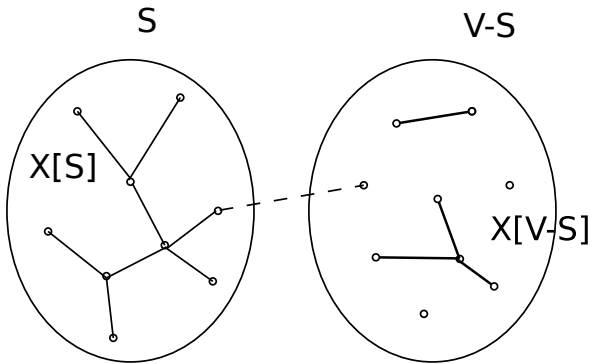
Greedy Design Strategy

- Each choice leaves one smaller subproblem
- Prove that $\exists$ an optimal solution that makes the greedy choice
- Show that the greedy choice, combined with an opt. sol. to subproblem, yields opt. sol. to the original problem.



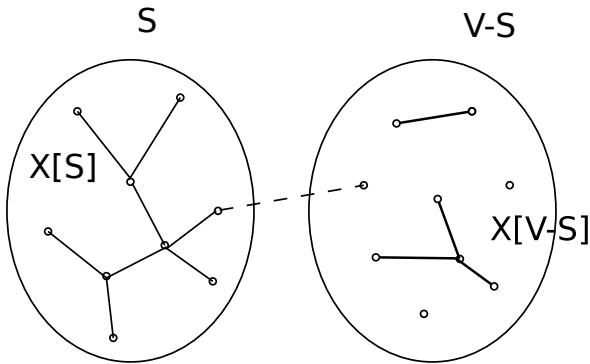
Generic MST

$X \leftarrow \{\}$
while $|X| < |V| - 1$ **do**
    Choose S s.t. X does not connect $(S, V - S)$
    Add the lightest crossing edge to X
end while

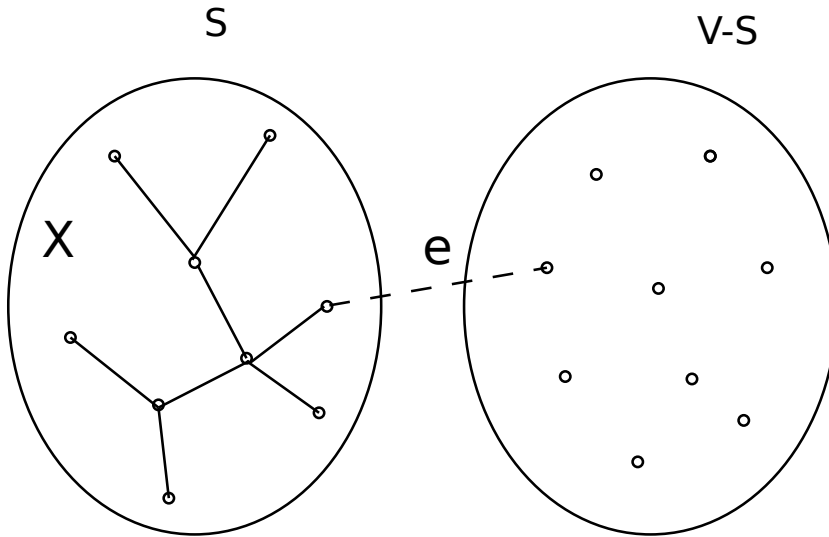


Generic MST

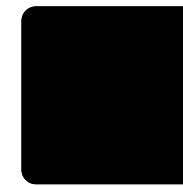
$X \leftarrow \{\}$
while $|X| < |V| - 1$ **do**
    Choose S s.t. X does not connect $(S, V - S)$
    Add the lightest crossing edge to X
end while



Prim's Algorithm

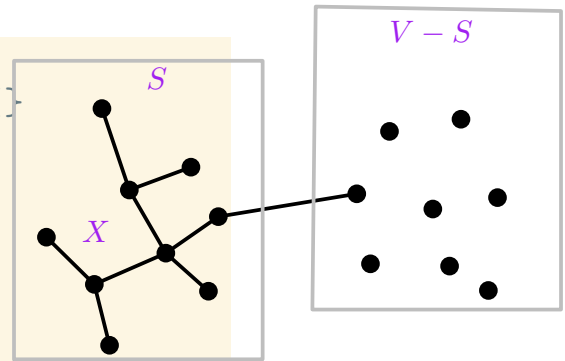


$S =$ nodes reached so far



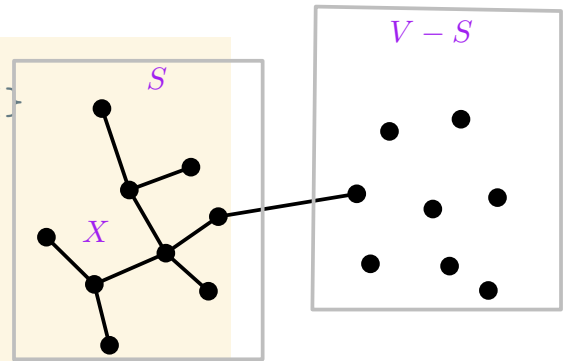
Prim's Algorithm

```
def prim(G,root):
    pq = pqdict(); prev = {}
    for v in G.keys():
        pq.additem(v,inf)
    pq.updateitem(root,0)
    while len(pq)>0:
        v = pq.pop()
        for (z,weight) in G[v]:
            if z in pq and weight < pq[z]:
                prev[z]=v
                pq.updateitem(z,weight)
    return prev
```

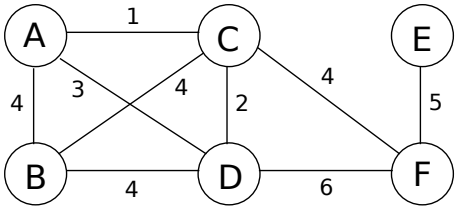


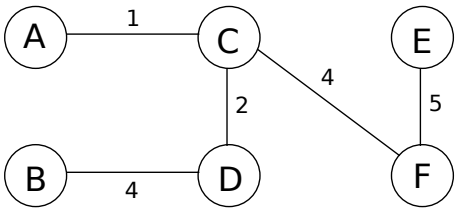
Prim's Algorithm

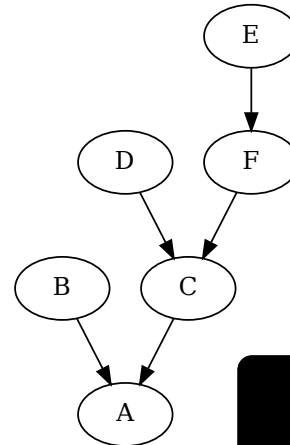
```
def prim(G,root):
    pq = pqdict(); prev = {}
    for v in G.keys():
        pq.additem(v,inf)
    pq.updateitem(root,0)
    while len(pq)>0:
        v = pq.pop()
        for (z,weight) in G[v]:
            if z in pq and weight < pq[z]:
                prev[z]=v
                pq.updateitem(z,weight)
    return prev
```



Prim example







```
# source for data https://raw.githubusercontent.co

from math import inf
from prim import prim,as_dot
import sys

def as_dot(T):
    print("digraph␣G␣{")
    for v in T.keys():
        print ("\t{:d}␣->␣{:d}".format(v,T[v]))
    print("}")

def total_dist(T,D):
    sum = 0
    for v in T.keys():
```

```
        sum += D[v][T[v]]
    return sum

edges = [
    [0,1,81.5865],
    [1,2,82.5435],
    [2,3,67.9046],
    [3,4,64.1961],
    [4,5,253.6179],
    [5,6,113.4832],
    [1,6,89.886],
    [0,9,252.6349],
    [0,36,338.7641],
    [1,7,73.8749],
    [5,8,208.4023],
```

```
    [5,25,123.7738],
    [7,8,169.7241],
    [8,20,196.889],
    [8,23,173.2129],
    [8,35,160.5986],
    [9,35,118.3903],
    [10,11,261.9837],
    [10,33,185.8489],
    [10,36,219.6409],
    [36,9,349.2051],
    [11,13,209.3103],
    [12,13,95.6952],
    [12,17,232.2036],
    [13,14,285.0276],
    [14,15,129.7066],
```

```
    [14,16,187.3142],
    [15,21,298.9509],
    [16,17,130.5206],
    [17,18,137.4058],
    [18,19,165.1677],
    [19,35,130.2419],
    [20,19,84.6042],
    [20,23,181.5531],
    [20,29,332.0392],
    [21,22,125.4547],
    [22,20,337.2769],
    [23,24,165.3347],
    [24,25,180.6815],
    [26,23,66.4409],
    [26,30,112.0243],
```

```
    [27,29,54.7012],
    [27,31,174.6213],
    [28,31,92.6074],
    [28,25,154.9616],
    [29,30,177.6956],
    [30,27,73.2377],
    [31,32,189.1641],
    [32,25,166.7202],
    [33,11,91.1888],
    [33,34,199.7968],
    [34,12,202.7576],
    [34,18,171.3123],
    [34,35,161.2244],
    [34,9,138.7795]
]
```

```
Graph = { j : [] for j in range(0,37) }
Dist = [ [ inf for j in range(37) ] for i in range
for edge in edges:
    (a,b,dist) = edge
    Dist[a][b] = Dist[b][a] = dist
    Graph[a].append((b,dist))
    Graph[b].append((a,dist))

T = prim (Graph,36)
as_dot(T)
print(total_dist(T,Dist),file=sys.stderr)
```