CS7800: Advanced Algorithms

Greedy Algs II

- Minimum spanning trees
- Classroom scheduling

Jonathan Ullman

09-16-2022

Network Design Problems

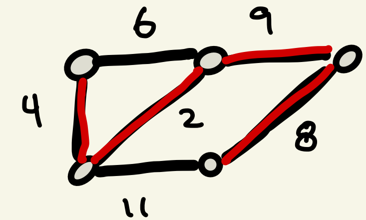
Given:

- A set of nodes V
- A set of potential edges E
- Costs $\{w_e\}$ for building each edge

$$G = (V, E, \{w_e\})$$

build a network that is well connected and cheap

Minimum Spanning Trees



Given:

- A set of nodes V
- A set of potential edges E
- Costs $\{w_e\}$ for building each edge

≥ 0

$G = (V, E, \{w_e\})$

≥ 0

build a network that is well connected and cheap

A subgraph $T = (V, E')$

T is connected

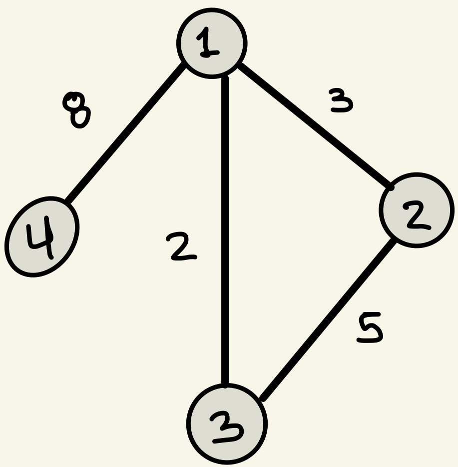
minimize $\sum_{e \in T} w_e$

T is a tree

Brief Aside: Adjacency List Representation

$|V| = n$ $\quad |E| = m$

Graph $G = (V, E, \{w_e\})$



Adjacency List of G

Nodes → **Neighbors**

- 1 → [2, w=3] → [3, w=2] → [4, w=8]
- 2 → [1, w=3] → [3, w=5]
- 3 → [2, w=5] → [1, w=2]
- 4 → [1, w=8]

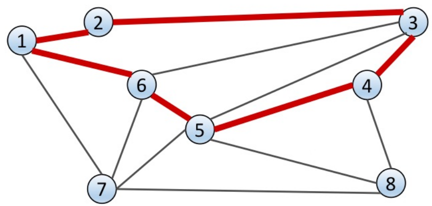
Size: $O(n+m)$

Time to List Neighbors of i : $O(\deg(i)+1)$

Time to List All Edges : $O(n+m)$

Cycles and Cuts

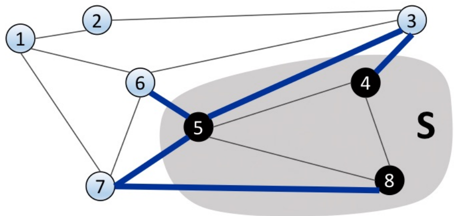
- **Cycle:** a set of edges $(v_1, v_2), (v_2, v_3), \ldots, (v_k, v_1)$



1, 2, 3, 4, 5, 6, 1

Cycle C = (1,2),(2,3),(3,4),(4,5),(5,6),(6,1)

- **Cut:** a partition of the nodes into $S, \bar{S}$



Cut S	= {4, 5, 8}
Cutset	= (5,6), (5,7), (3,4), (3,5), (7,8)

$$\text{Cutset}(S) = \{ e = (i,j) : i \in S, j \in \bar{S} \}$$

<u>Fact:</u> Any cycle intersects any cut in an even number of edges.

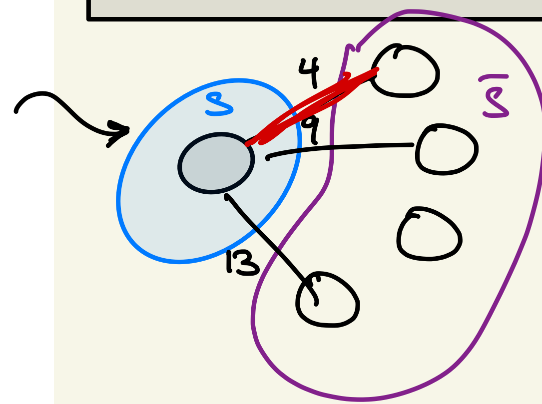
Cycle and Cut Property of MST

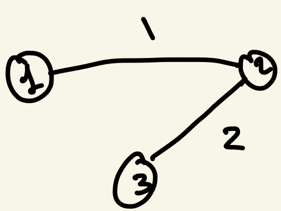
Assumption:

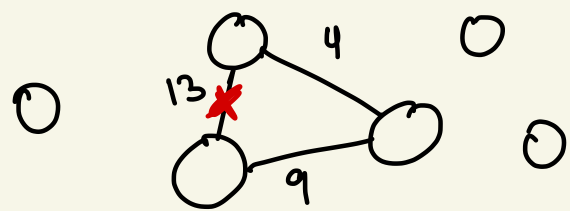
All w_e are distinct

- **Cut Property:** Let S be a cut. Let e be the minimum weight edge cut by S. Then the MST T^* contains e
  - We call such an e a safe edge

- **Cycle Property:** Let C be a cycle. Let f be the maximum weight edge in C. Then the MST T^* does not contain f.
  - We call such an f a useless edge

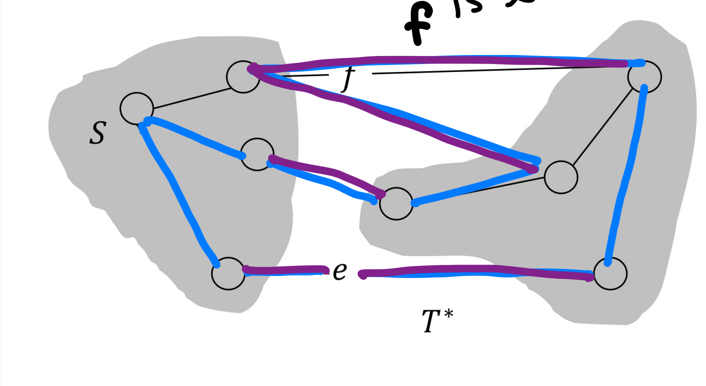






Proof of Cut Property

f is some edge on the cycle



- **Cut Property:** Let S be a cut. Let e be the minimum weight edge cut by S. Then the MST T^* contains e
  - We call such an e a safe edge

- let T^* be the MST
- let $S, \bar{S}$ be a cut
- let e be the min wt edge in the cutset of S
- assume $e \notin T^*$ (for contradiction)
- adding e to T^* would create a cycle C containing e
- C intersects the cut in ≥ 2 places, let $f \neq e$ be one of then $w_f > w_e$
- consider $T' = T^* + \{e\} - \{f\}$ (① $wt(T') < wt(T^*)$ ② T' is a spanning tree)

The "Only MST Algorithm"

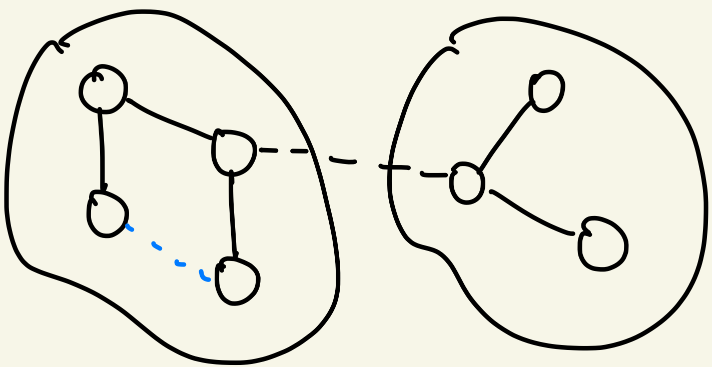
Let $T = (V, \emptyset)$

While T is not connected:

  add one or more new safe edges

Return T

Always terminates and outputs an MST no matter how we choose safe edges



Borůvka's Algorithm

Input: $G = (V, E, \{w_e\})$

Let $T = (V, \emptyset)$

count $\leftarrow$ CountLabel(T)

While count > 1

AddSafeEdges(G, T, count)

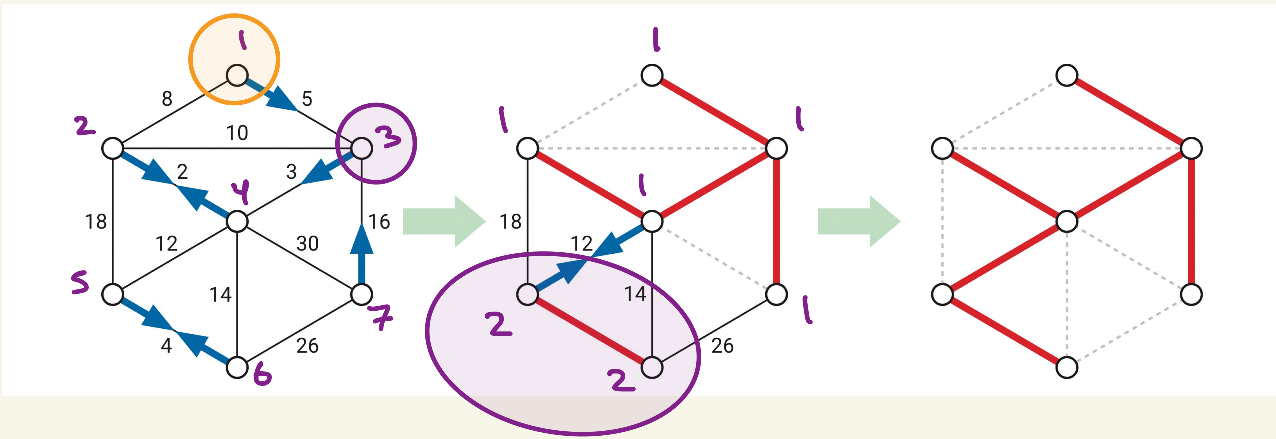
count $\leftarrow$ CountLabel(T)

Return T

Count connected components
Label nodes by component

Find the min-wt edges
leaving each connected component

Borůvka's Algorithm



Borůvka's Algorithm Correctness

Input: $G = (V, E, \{w_e\})$

Let $T = (V, \emptyset)$

count $\leftarrow$ CountLabel(T)

While count > 1

  AddSafeEdges(G, T, count)

  count $\leftarrow$ CountLabel(T)

Return T

Follows from the general template

Borůvka's Algorithm Running Time

Input: $G = (V, E, \{w_e\})$

Let $T = (V, \emptyset)$

count $\leftarrow$ CountLabel(T)

While count > 1

 AddSafeEdges(G, T, count)

 count $\leftarrow$ CountLabel(T)

Return T

How many iterations?

$O(\log n)$

How much time per iteration?

$O(m)$

Total Time: $O(m \log n)$

Borůvka's Algorithm Running Time

Input: $G = (V, E, \{w_e\})$

Let $T = (V, \emptyset)$

count ← CountLabel(T)

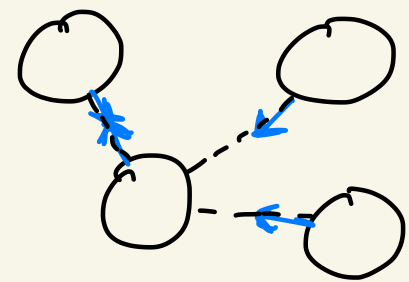
While count > 1

  Add Safe Edges(G, T, count)

  count ← CountLabel(T)

Return T

How many iterations?



- if I have count components I will add $\geq \frac{count}{2}$ edges
- every edge reduces count by 1
- next iteration there are at most $\frac{count}{2}$ components $\Rightarrow O(\log n)$

Borůvka's Algorithm Running Time

Input: $G = (V, E, \{w_e\})$

Let $T = (V, \emptyset)$

count ← CountLabel(T)

While count > 1

  Add Safe Edges(G, T, count)

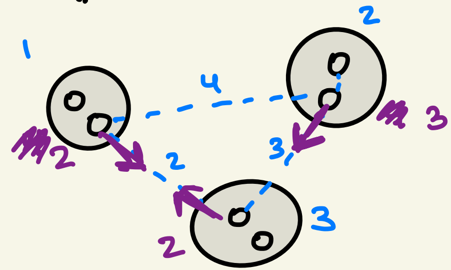
  count ← CountLabel(T)

Return T

How much time per iteration?

array for component labels

1	2	3	4	5	6
1	2	2	3	3	1

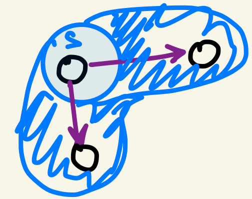


loop over edges

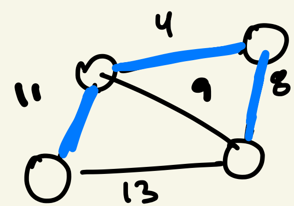
maintain cheapest edge so far

time : $O(n+m)$

Other MST Algorithms



- Prim:
    - Maintain a component S
    - Initially $S = \{v_1\}$
    - Add the safe edge for S until $S = V$



- Kruskal:
    - Sort edges
    - For each edge add it to T if it merges two components into one.

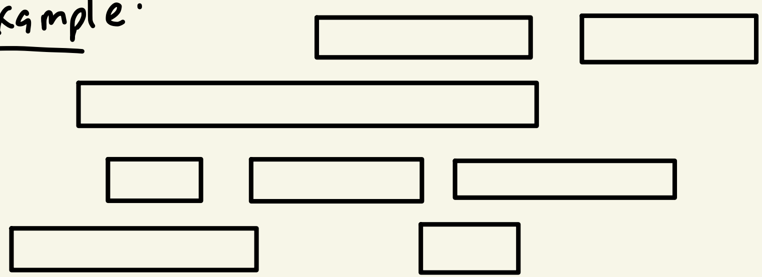
One More Greedy Proof Technique

<u>Input</u>: n "classes" $[s_i, f_i]$

<u>Output</u>: A "room" for each class so that

1. No room has two overlapping classes
2. The total number of rooms is as small as possible

<u>Example</u>:



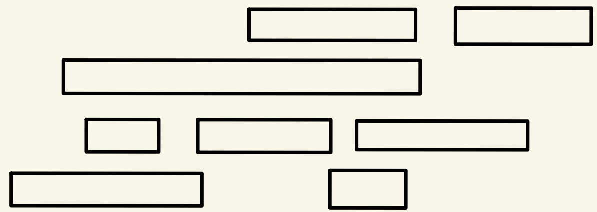
4 classrooms

A Greedy Algorithm

Sort intervals by start
$s_1 \le s_2 \le \dots \le s_n$

For $i = 1, \dots, n$:

assign class i to the
lowest numbered empty room



Proof of Correctness: Duality

$$\overbrace{\underbrace{TTTT\ldots T}_{x}H\;TTH\;TH\;\cdots\;TH}^{y}$$

$x+1$ heads