CS3000
6/6 - Tues.

Admin
- Long hw4 due today 9pm
- Short hw3 out now, due thurs 9pm
- Rec 4 today!
- Fun optional recitation thurs

Agenda
1. Minimum Spanning Trees
2. Kruskal
3. Prim

Recap

DFS ⇝ topological ?
track the order in which we hit dead ends (v.f)

DFS ⇝ determine if there's a cycle?
if there's a back edge, there's a cycle

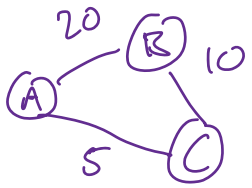
DFS run-time
$\Theta(V+E)$ (pretty good!)

1. Minimum Spanning Trees

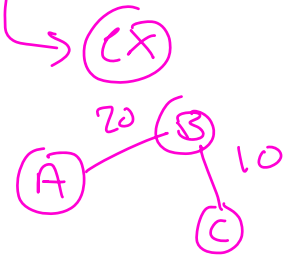
- apply techniques we know to graph problems

Today:
- undirected
- weighted

} graph G

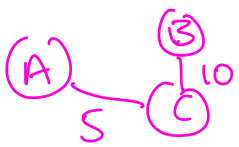


Want:
- Spanning tree
  - ↳ tree subset of edges include all vertices



- minimum spanning tree
  - ↳ least total weight

tree ✓
all vertices ✓
total weight = 30



tree ✓
all vertices ✓
total weight = 15

MST

MST is an optimization problem

DP
Greedy ←

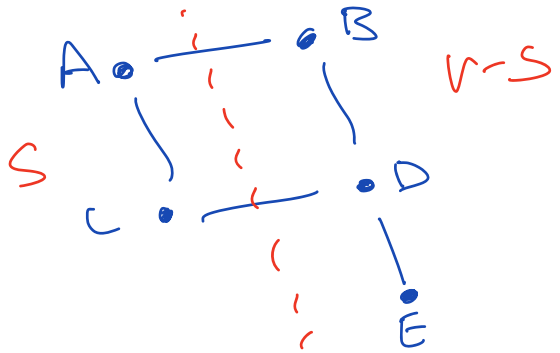
Implementation Detail

- weight function, parameter to our functions

Func (G, w) ⟶ w(u, v) weight of edge (u, v)

Definitions

- MST is a set of edges
- Set $A \subseteq MST$ (while we're building)
- **safe edge** can be added to A, and A is still $\subseteq MST$
- A cut $(S, V-S)$ is a partition of vertices V

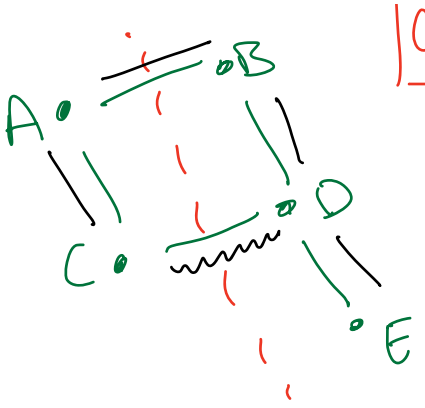


- An edge (u,v) with $u \in S$ and $v \in V-S$
  "crosses the cut" above: (A,B) (C,D)

- A **light edge** is edge with min weight crossing the cut

Both algos...

- cut graph (keep some vertices in S)
- add light edge to MST (A)
  ↳ greedy choice!



Greedy works for MST

(C, D) is light edge

Suppose we chose (A, B) instead

↳ we can swap in (C, D) and get a better tree

[adding in (C, D) will create a cycle in MST, we can remove same edge to get back to a tree.]

2. Kruskal + Prim

- graph G
- undirected
- weighted
- Adj list

Sets

- MAKE-SET(v) → create one-element set with v
- FIND-SET(v) → returns rep from v's set
- $A \cup \{v\}$ → add element to set A
- UNION(u, v) → draw edge between u and v (puts u, v in same set)

Heaps

- INSERT(H, v) ⟶ insert v into H and heapify
- EXTRACT-MIN(H) ⟶ removes + return smallest element, and heapify
- DECREASEKEY(H, v, w) ⟶ replace v's key with w, and heapify

Kruskal: make set of edges that form MST

Prim: track v.key and v.π on every vertex

What we want to know...

- Kruskal: at each step, what's in A, what are sets
- Prim: at each step, what are v.key and v.π
- What is total weight of MST?
- Be ready to answer Qs from other team

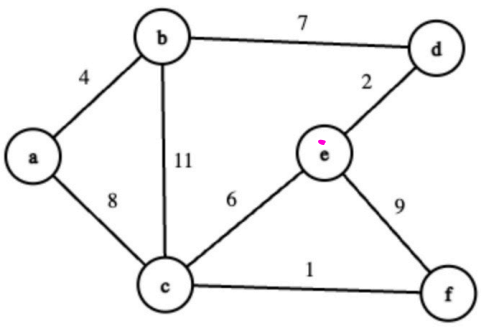
10:46

KRUSKAL(G, w)

```
1  A = {}
2  for each vertex v ∈ G.V
3        MAKE-SET(v)
4  create a single list of the edges in G.E
5  sort the list of edges into monotonically increasing order by weight w
6  for each edge (u, v) taken from the list in sorted order
7        if FIND-SET(u) ≠ FIND-SET(v)
8              A = A ∪ {(u, v)}
9              UNION(u, v)
10 return A
```

PRIM(G, w, r)

```
1  for each vertex u ∈ G.V
2        u.key = ∞
3        u.π = NIL
4  r.key = 0
5  H = {}
6  for each vertex u ∈ G.V
7        INSERT(H, u)
8  while H ≠ {}
9        u = EXTRACT-MIN(H)
10       for each vertex v ∈ G.adj[u]
11             if v ∈ H and w(u, v) < v.key
12                   v.π = u
13                   v.key = w(u, v)
14                   DECREASE-KEY(H, v, w(u, v))
```



Kruskal

A {}

sets a, b, c, d, e, f

Step #1 (c,f) weight 1

A = {(c,f)} sets: a, b, d, e, cf

Step#2 (e,d) w 2 sets: a, b, de, cf

A = {(c,f), (e,d)}

Step#3 (a,b) w 4

A = {(c,f), (e,d), (a,b)} sets: ab, de, cf

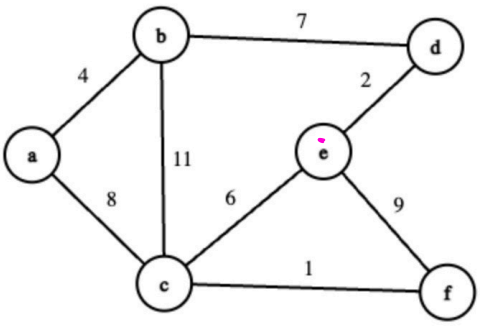
Step#4 (c,e) w 6 sets: ab, decf

A = {(c,f), (e,d), (a,b), (c,e)}

Step#5 (b,d) w 7 sets: abdecf

A = {(c,f), (e,d), (a,b), (c,e), (b,d)}

1 2 4 6 7 = 20



Prim

	A	B	C	D	E	F
key	0	∞	∞	∞	∞	∞
π	nil	nil	nil	nil	nil	nil

H = A (smallest)

Step #1

	A	B	C	D	E	F
key	0	4	8	∞	∞	∞
π	nil	A	A	nil	nil	nil

H = BCDEF

Step #2 extract B

	A	B	C	D	E	F
key	0	4	8	7	∞	∞
π	nil	A	A	B	nil	nil

H = CDEF

Step #3 extract D

	A	B	C	D	E	F
key	0	4	8	7	2	∞
π	nil	A	A	B	D	nil

Step #4 Extract E H = CEF

	A	B	C	D	E	F
key	0	4	6	7	2	9
π	nil	A	E	B	D	E

Step #5 Extract C H = CF

	A	B	C	D	E	F
key	0	4	6	7	2	1
π	nil	A	E	B	D	C

H = F

Step #6 Extract F

	A	B	C	D	E	F
key	0	4	6	7	2	1
π	nil	A	E	B	D	C

Final

	A	B	C	D	E	F
key	0	4	6	7	2	1
π	nil	A	E	B	D	C

total weight = 4 + 6 + 7 + 2 + 1 = 20 omg ;;

Run Time

```
PRIM(G, w, r)
 1  for each vertex u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  H = {}
 6  for each vertex u ∈ G.V
 7      INSERT(H, u)
 8  while H ≠ {}
 9      u = EXTRACT-MIN(H)
10      for each vertex v ∈ G.adj[u]
11          if v ∈ H and w(u, v) < v.key
12              v.π = u
13              v.key = w(u, v)
14              DECREASE-KEY(H, v, w(u, v))
```

V lg V

V + E

lgV each time

Θ((V+E) lg V)

loop + heap

```
KRUSKAL(G, w)
 1  A = {}
 2  for each vertex v ∈ G.V
 3      MAKE-SET(v)
 4  create a single list of the edges in G.E
 5  sort the list of edges into monotonically increasing order by weight w
 6  for each edge (u, v) taken from the list in sorted order
 7      if FIND-SET(u) ≠ FIND-SET(v)
 8          A = A ∪ {(u, v)}
 9          UNION(u, v)
10  return A
```

E

Sorting: Θ(E lg E)

Band by sort!

(Roughly the same)