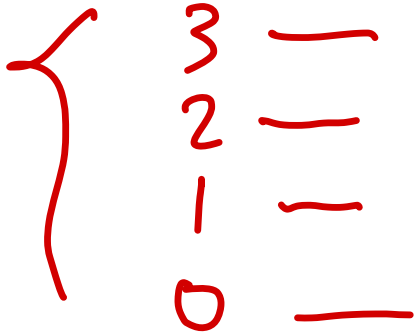
Lecture 19: Minimum Spanning Trees, Part 2

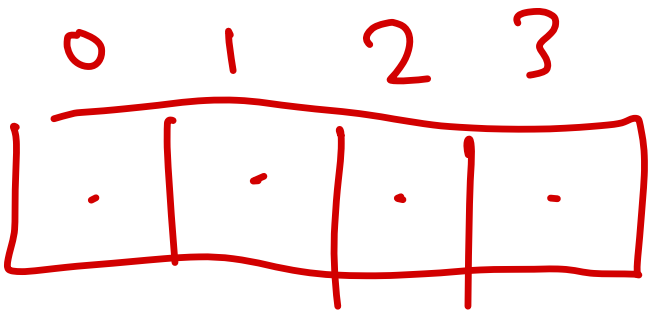
COSC 311 *Algorithms*, Fall 2022

Overview

1. Midterm Comments
2. Prim's Algorithm, Again
3. Kruskal's Algorithm

MT: 7 Q's



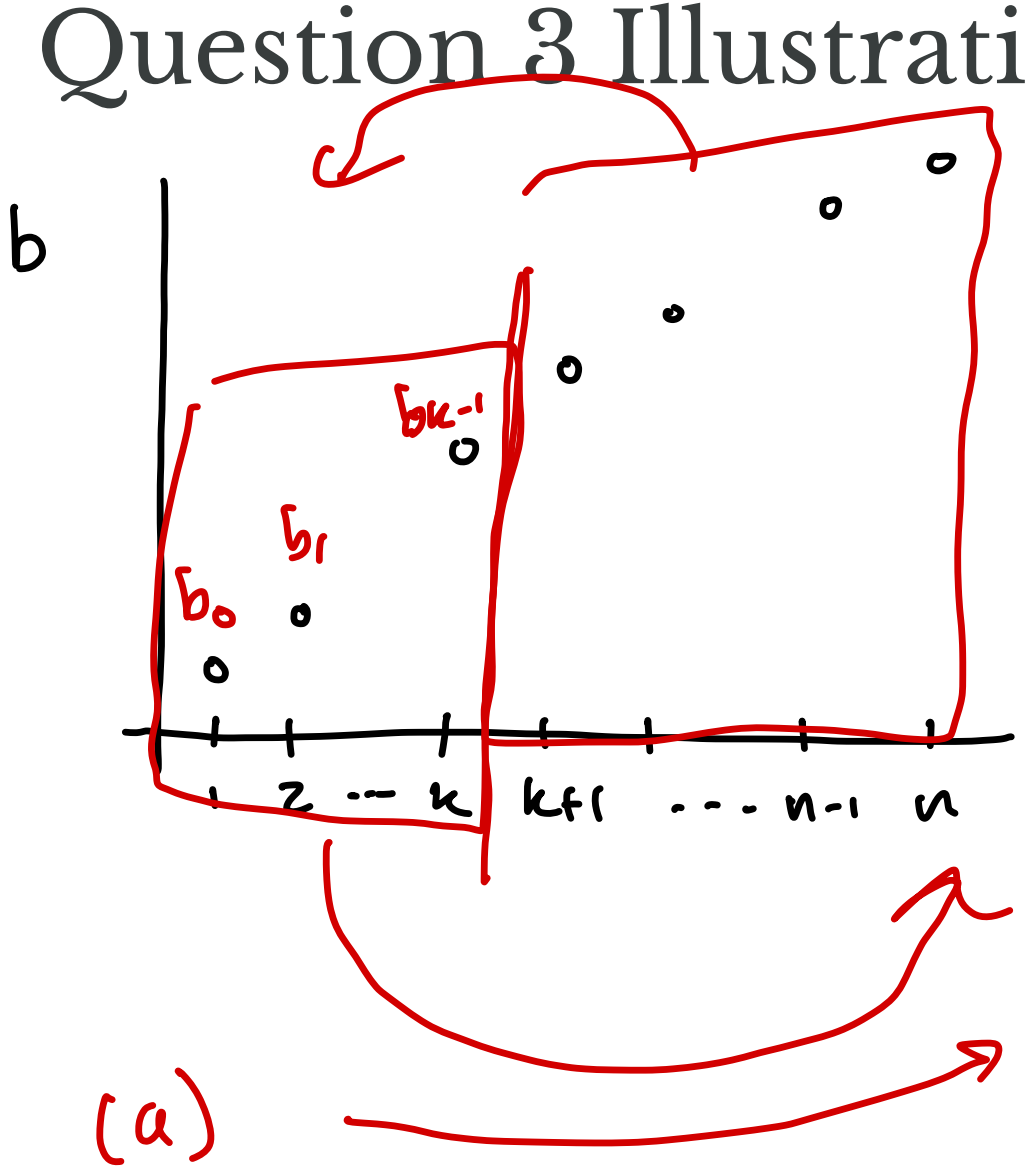


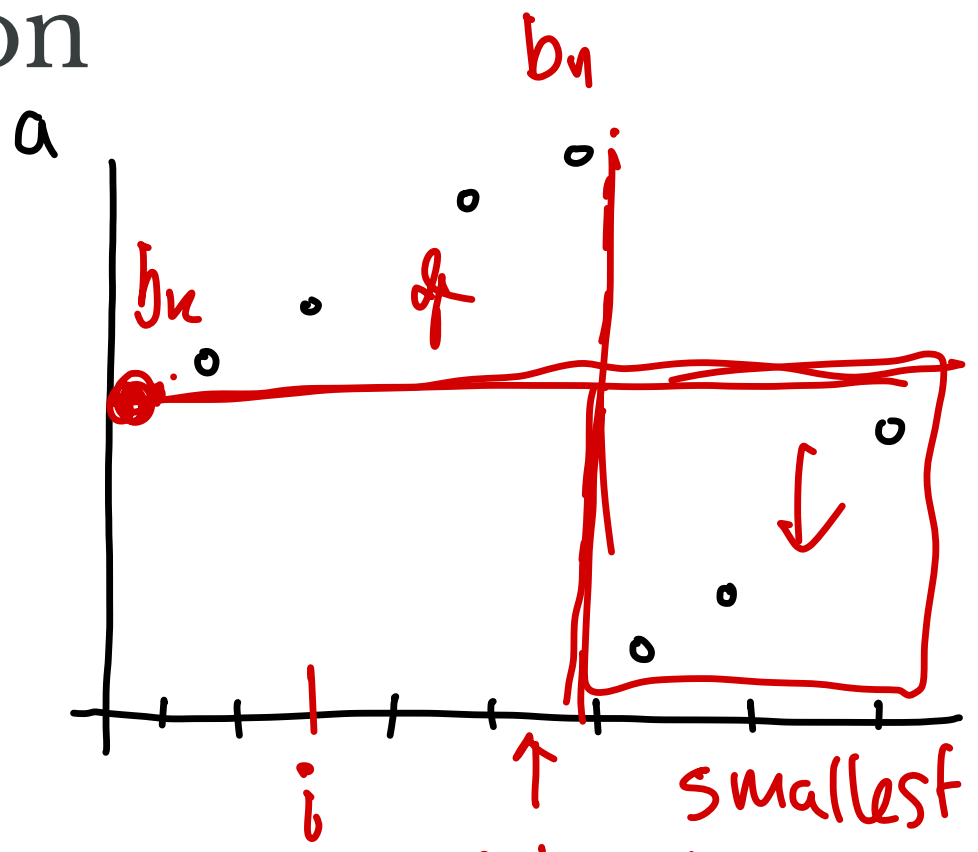
Overall Midterm Comments

1. Interpretation of grades
2. Very happy with grade distribution
3. Two more consistent issues
   - Question 2 (0, Θ, Ω) revisit
   - Question 3 (D&C Algorithm) ←

Find k

Question 3 Illustration





a[i] ? a[c]

$\leq$

$\Rightarrow i \leq n-k$

(a)

(b) Binary Search

MSTs

Input:

- a weighted graph $G = (V, E)$ with edge weights w

Output:

- a set F of edges in E such that
  1. (V, F) is connected
  2. sum of weights of edges in F is minimal among all connected sub-graphs of G

The graph $T = (V, F)$ is called a **minimum spanning tree** of G

Prim's Algorithm

```
PrimMST(V, E):
  initialize set S = {v} with v arbitrary
  initialize set F = {} of MST edges, priority queue Q
  for each neighbor x of v
    add (v, x) to Q with priority w(v, x)
  while Q is not empty
    (u, v) <- removeMin(Q)
    if S doesn't contain v
      add (u, v) to F
      for each neighbor x of v
        add (v, x) to Q with priority w(v, x)
  return (S, F)
```

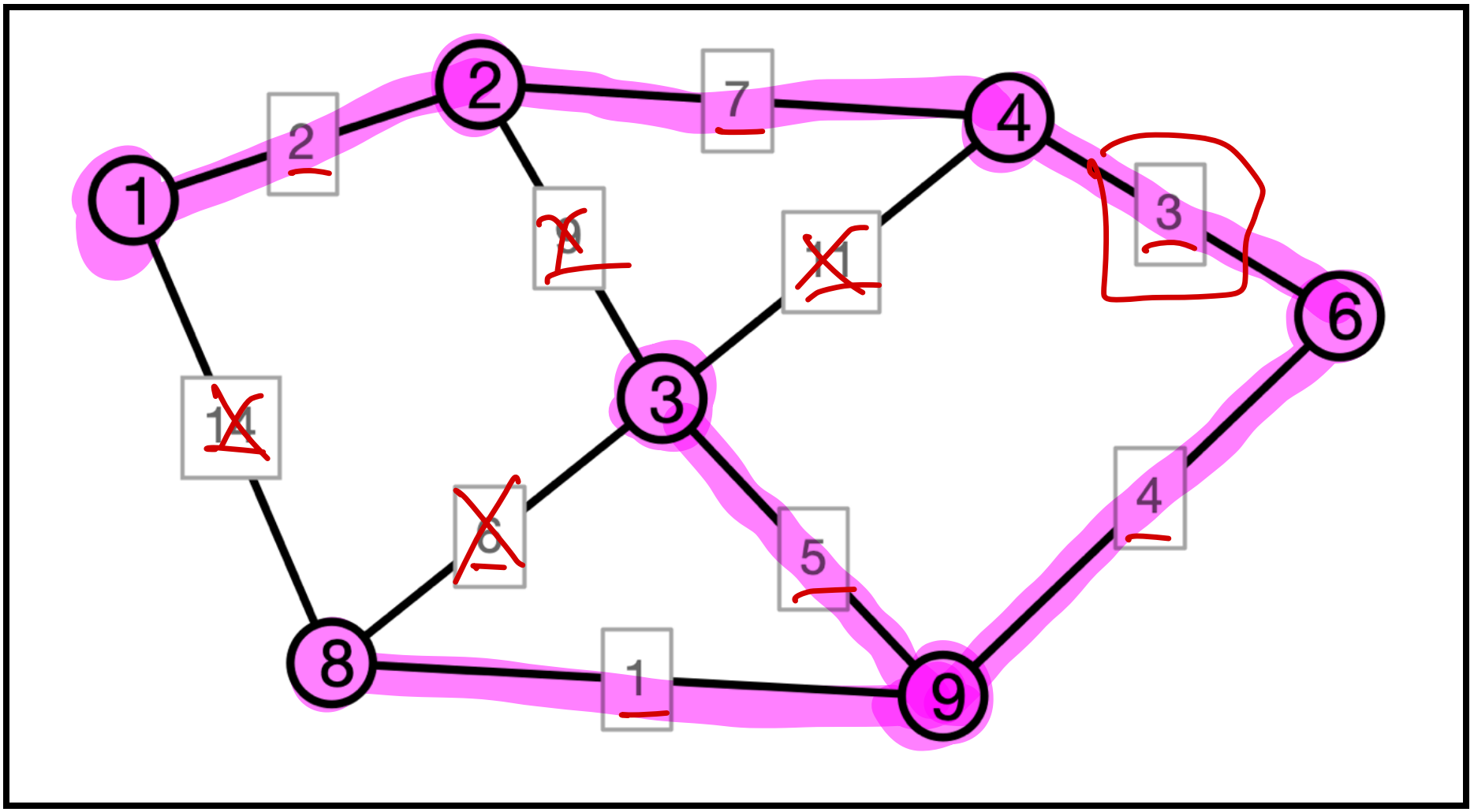
vertices in tree so far

edges so far

v has not yet been added to tree, add v to S

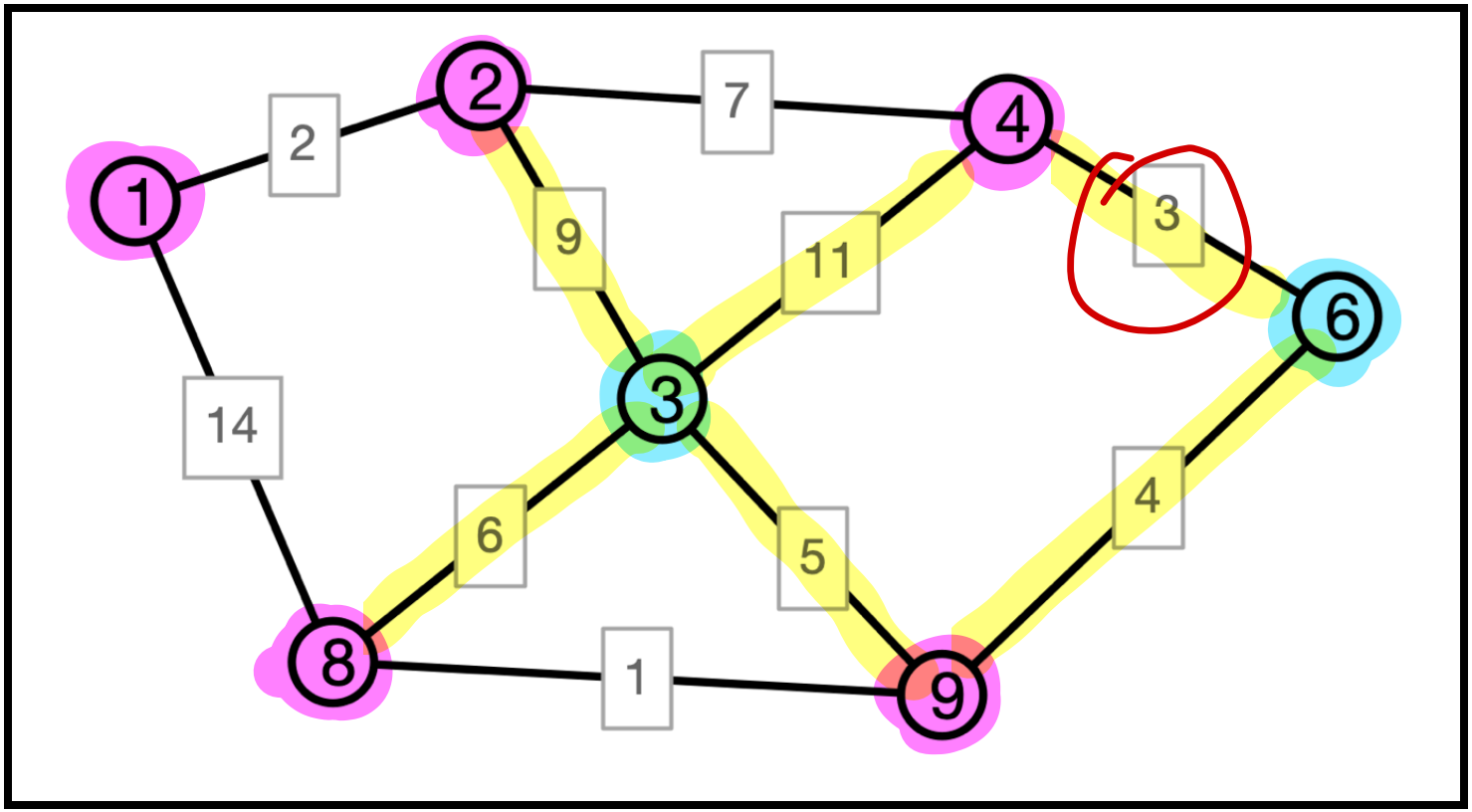
Prim Illustration

Greedy: always choose lightest edge to a new vertex



Cuts in Graphs

Definition. Let $G = (V, E)$ be a graph. A **cut** in G is a partition of V into two (non-empty) subsets U and $V - U$.



Cuts and MSTs

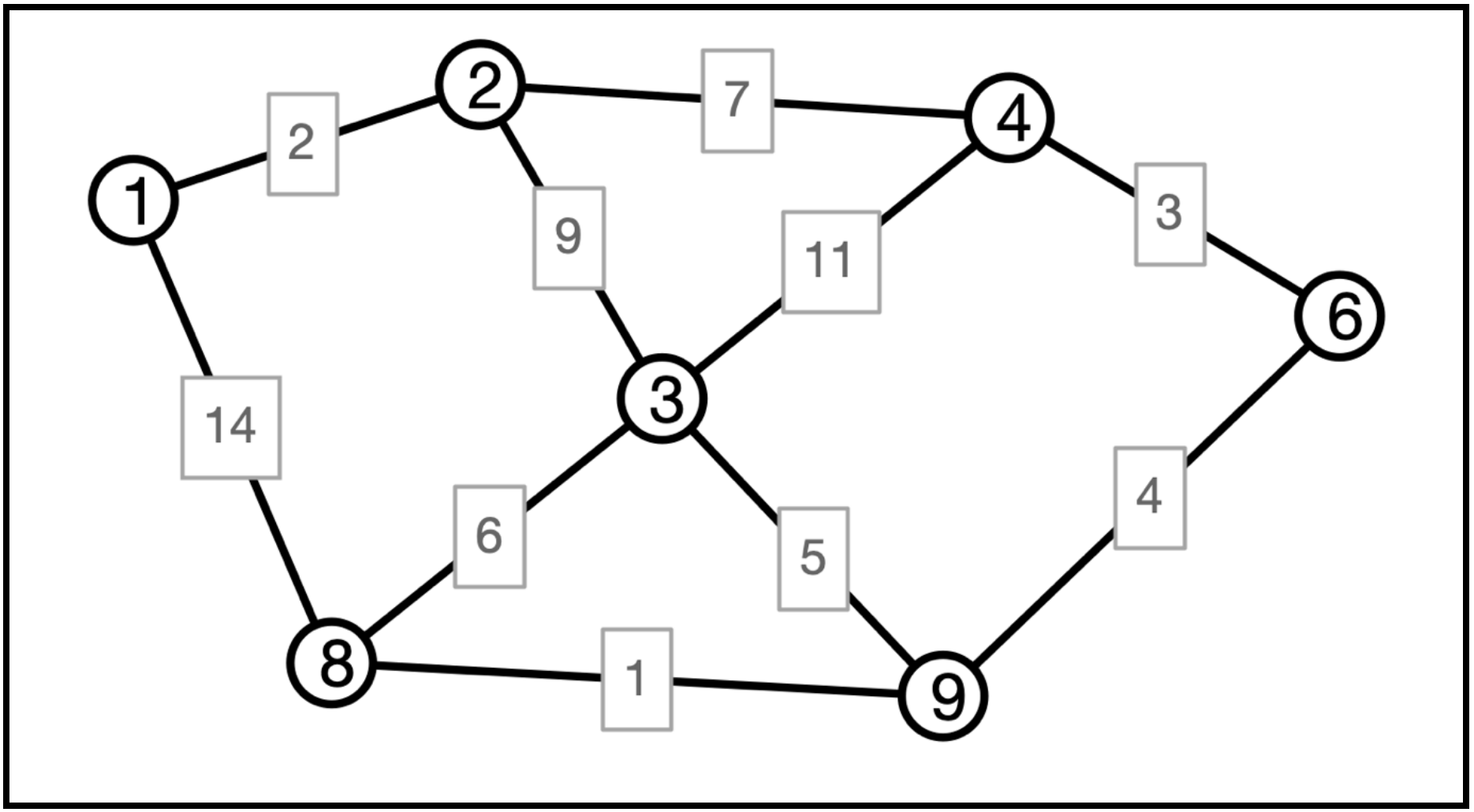
Cut Claim. Suppose:

- $G = (V, E)$ is a weighted graph (with distinct edge weights)
- $U, V - U$ a cut in G
- $T = (V, F)$ an MST
- $e = (u, v)$ is the minimum weight edge that crosses the cut
  - $u \in U$ and $v \in V - U$

Then:

- T contains the edge e

Cut Claim Illustration



Prim, Again

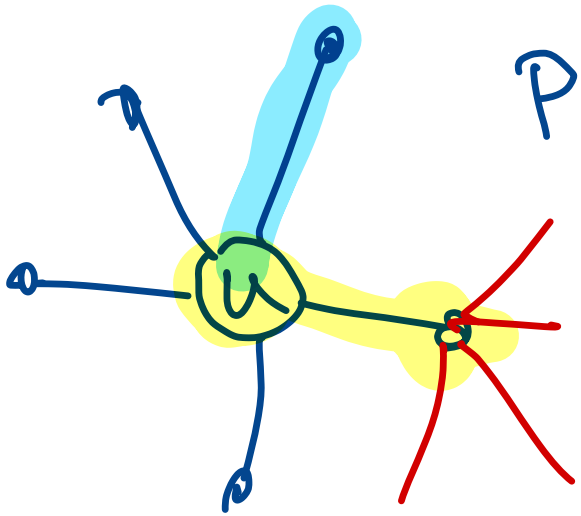
```
PrimMST(V, E):
  initialize set S = {v} with v arbitrary
  initialize set F = {} of MST edges, priority queue Q
  for each neighbor x of v
    add (v, x) to Q with priority w(v, x)
  while Q is not empty
    (u, v) <- removeMin(Q)
    if S doesn't contain v
      add (u, v) to F
      for each neighbor x of v
        add (v, x) to Q with priority w(v, x)
  return (S, F)
```

Prim Correctness I

Claim 1. Every edge added is in MST

Cut claim $\Longrightarrow$ Prim produces an MST.

Why?



Prim chooses cheapest, Cut Claim ⇒ edge must be in MST

Prim Correctness I

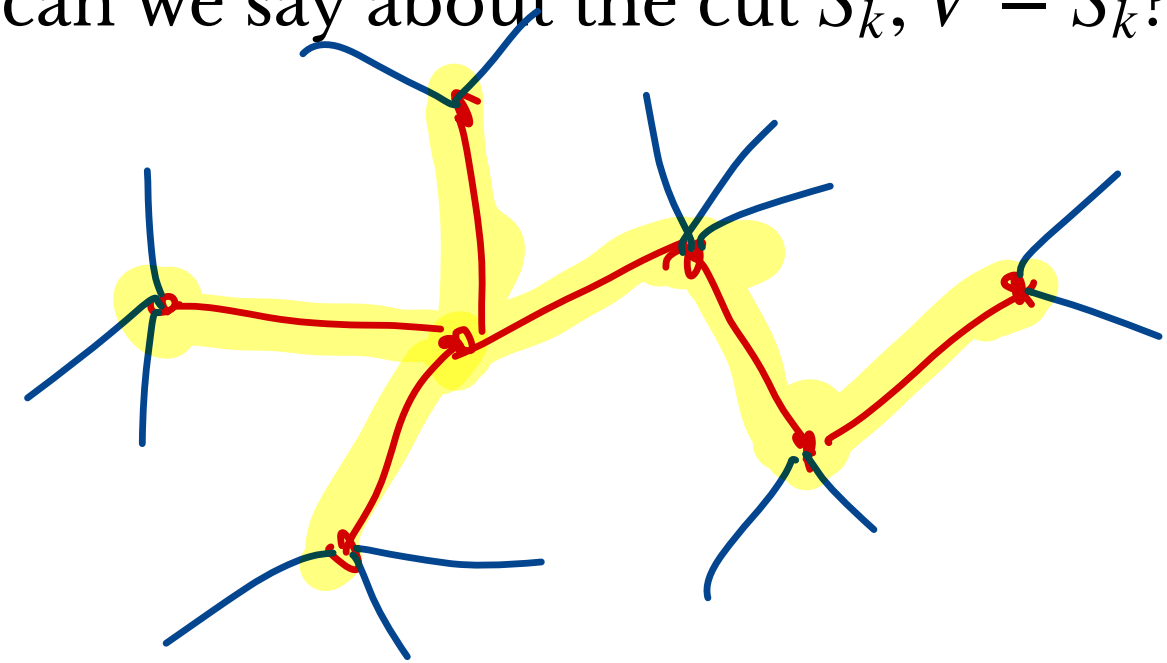
Cut claim $\implies$ Prim produces an MST.

Why?

Consider kth edge e_k added by Prim

- S_k = contents of S before edge added

What can we say about the cut S_k, $V - S_k$?

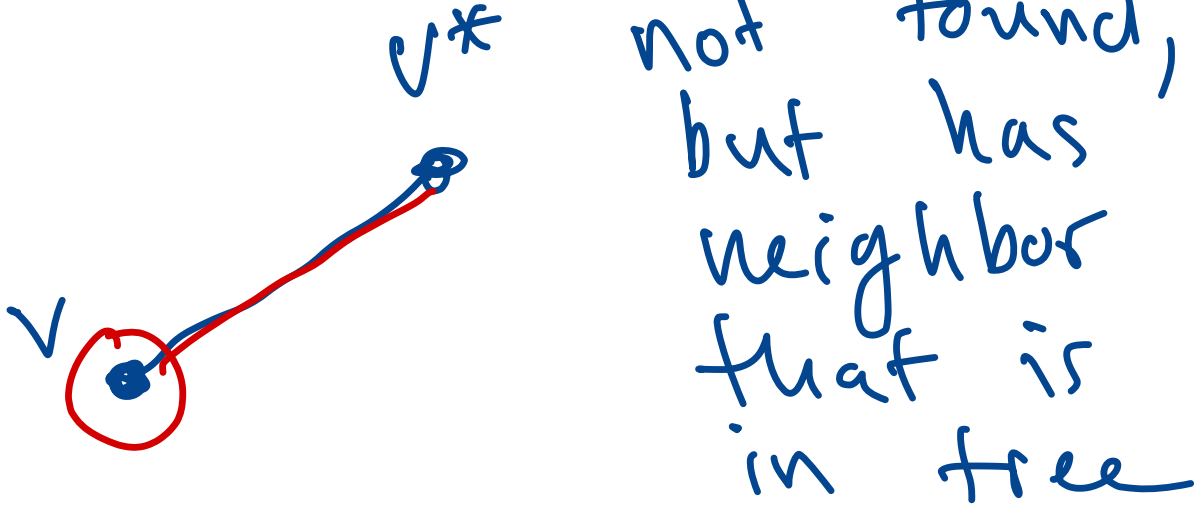


Prim Correctness II

First Conclusion. Every edge e added by Prim's algorithm is in the MST.

Still to show. All MST edges are added.

Why is this so?



Prim Correctness II

First Conclusion. Every edge e added by Prim's algorithm is in the MST.

Still to show. All MST edges are added.

Why is this so?

all vtx connected

- Suffices to argue that Prim produces a spanning tree
- Set of edges found by Prim form a tree
- All vertices of V are in tree (if G is connected)

Conclusion. Prim's algorithm produces an MST.

Prim Running Time?

boolean array, ops in $O(1)$ time

Min heap $O(\log(\text{size}))$ ops

```
PrimMST(V, E):
  initialize set S = {v} with v arbitrary
  initialize set F = {} of MST edges, priority queue Q
  for each neighbor x of v
    add (v, x) to Q with priority w(v, x)
  while Q is not empty
    (u, v) <- removeMin(Q)
    if S doesn't contain v
      add (u, v) to F, add v to S
      for each neighbor x of v
        add (v, x) to Q with priority w(v, x)
  return (S, F)
```

$O(n)$ iterations $\Rightarrow O(n \log n)$

$O(m \log m)$

only n times, once per vertex

$\leq 2m$ iterations

Total iterations of this

$\deg(v)$ iterations

$\deg(v_1) + \deg(v_2) + \deg(v_3) + \dots + \deg(v_n) = 2m$

$\Rightarrow$ i.e. $O(m \log m)$

Total $O(m \log m)$

n vertices, m edges

Conclusion

Prim's algorithm:

- computes an MST in G
- if efficient priority queue is used, running time is $O(m \log n)$

Prim's algorithm is **greedy**

- to grow a tree, always add the lightest outgoing edge

MSTs, Another Way

Prim:

- Grow tree greedily from a single seed vertex
- Maintain a (connected) tree

Edge Centric View:

- Maintain a collection of edges (not necessarily a tree)
- Add edges to collection to eventually build an MST

Questions:

- How to *prioritize* edges?
- How to determine whether or not to include an edge?