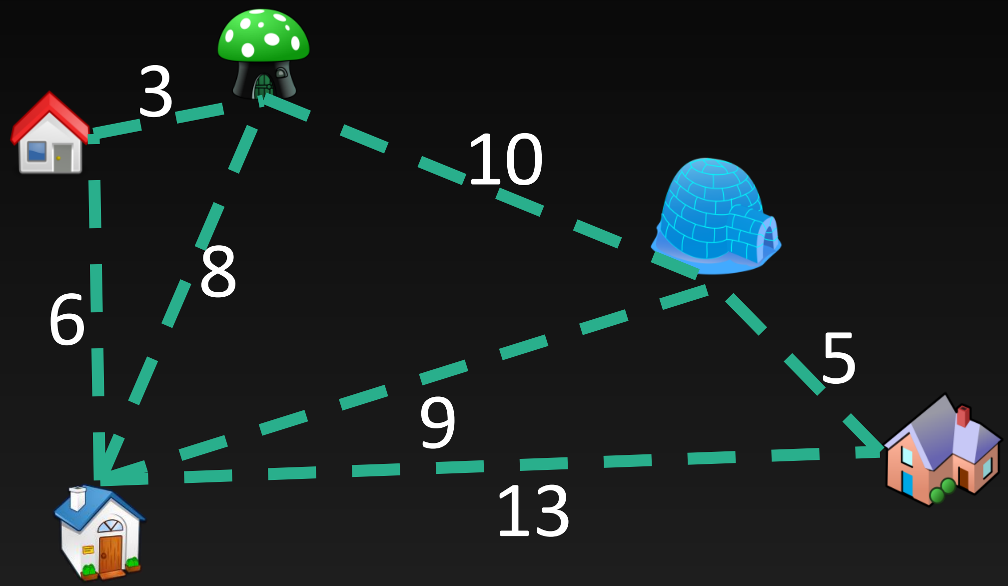
CS 4800: Algorithms & Data

Lecture 16

March 14, 2017

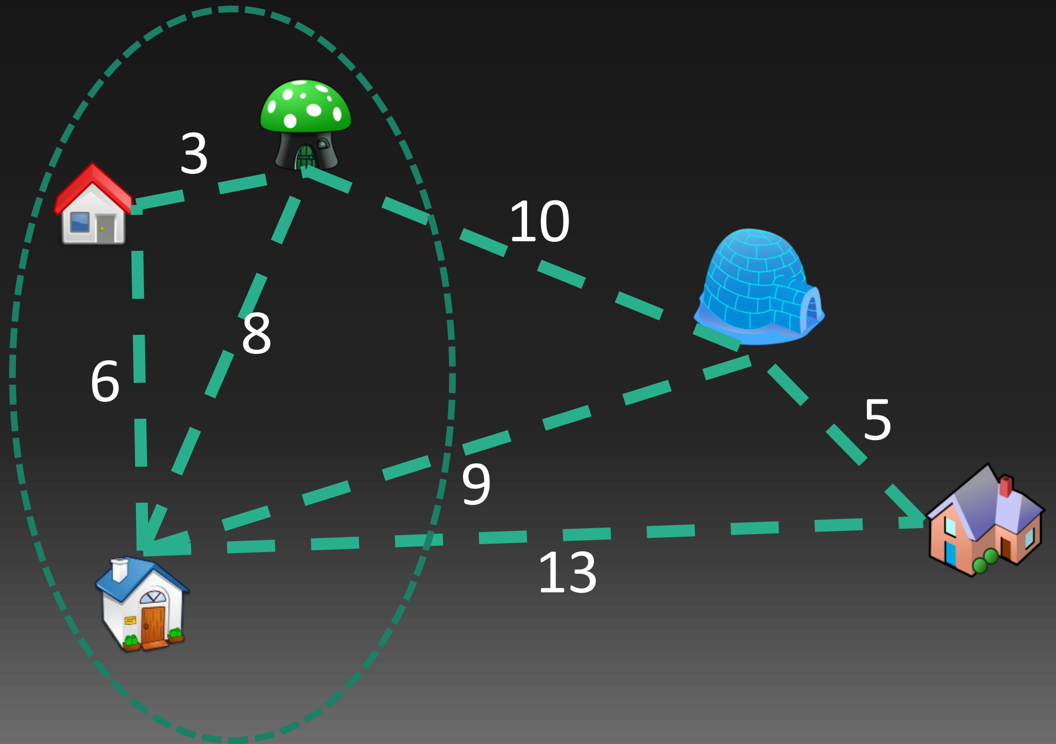
Minimum spanning tree (MST)



- G = (V, E, w), w positive
- Want a set of edges that connects all V and has minimum cost
- For simplicity, assume all weights are distinct

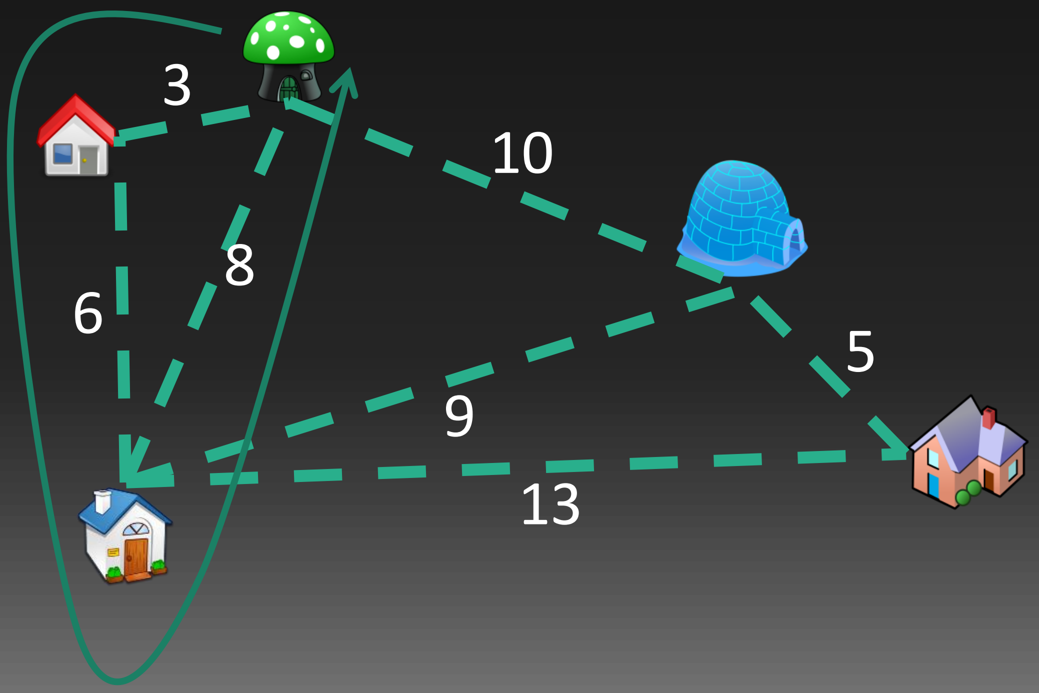
Blue rule

- Pick a set of nodes S
- Color minimum weight edge in cut induced by S **blue**



Red rule

- Pick a cycle C
- Color the maximum weight edge in C **red**



What we proved

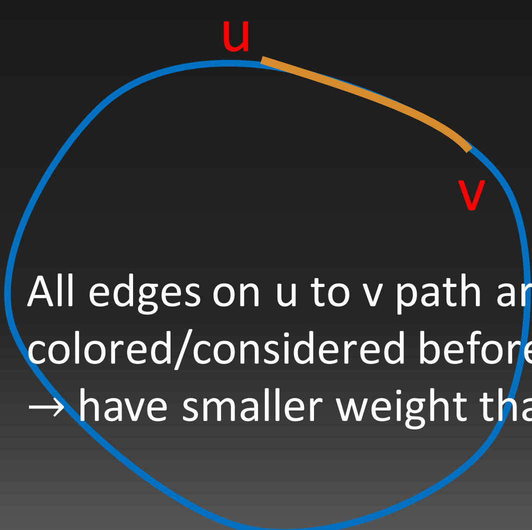
- All blue edges belong to the minimum spanning tree
- All red edges do not belong to the minimum spanning tree

Generic algorithm

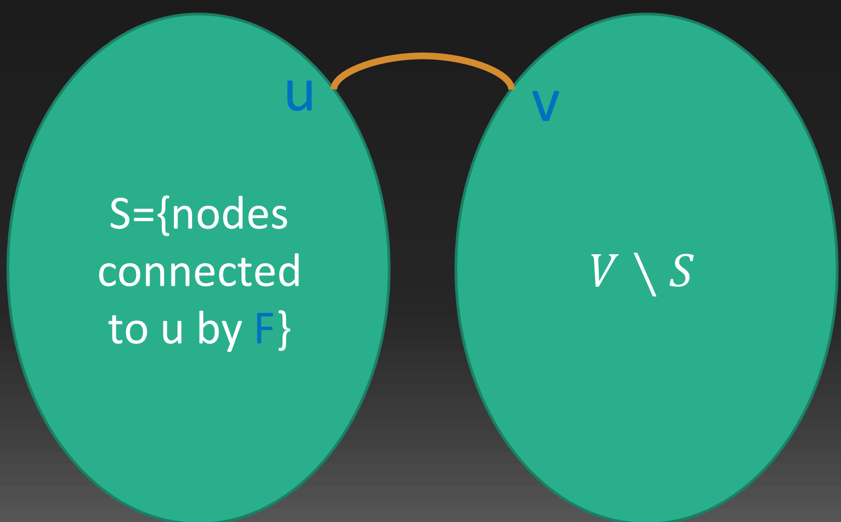
- Maintain an acyclic set of blue edges F
- Initially no edge is colored, $F = \emptyset$
- Repeat the following in arbitrary order
  - Consider a cut with no blue edge. Color the minimum weight edge in the cut blue.
  - Consider a cycle with no red edge. Color the maximum weight edge in the cycle red.
  - Terminate when V-1 edges colored blue.

Kruskal's algorithm

- Consider edges in order of increasing weights
- When considering e=(u,v)
  - If u and v are connected by F, color e red
  - If u and v are not connected by F, color e blue

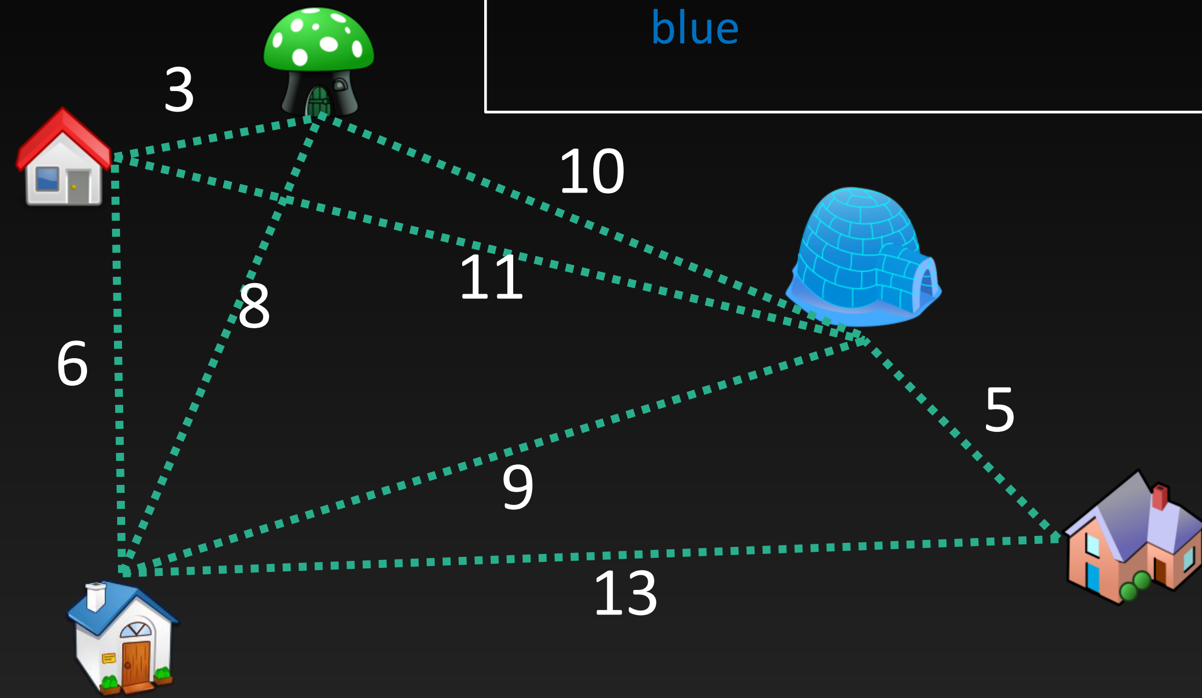


All edges on u to v path are colored/considered before (u,v)
→ have smaller weight than e



Example

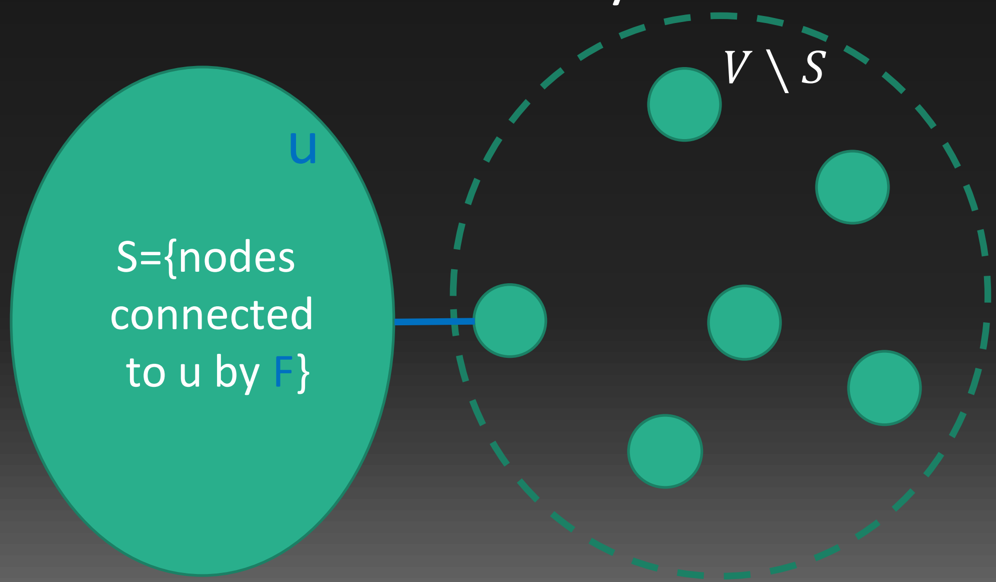
- Consider edges in order of increasing weights
- When considering e=(u,v)
  - If u and v are connected by F, color e red
  - If u and v are not connected by F, color e blue



3	5	6	8	9	10	11	13
✓	✓	✓	✗	✓	✗	✗	✗

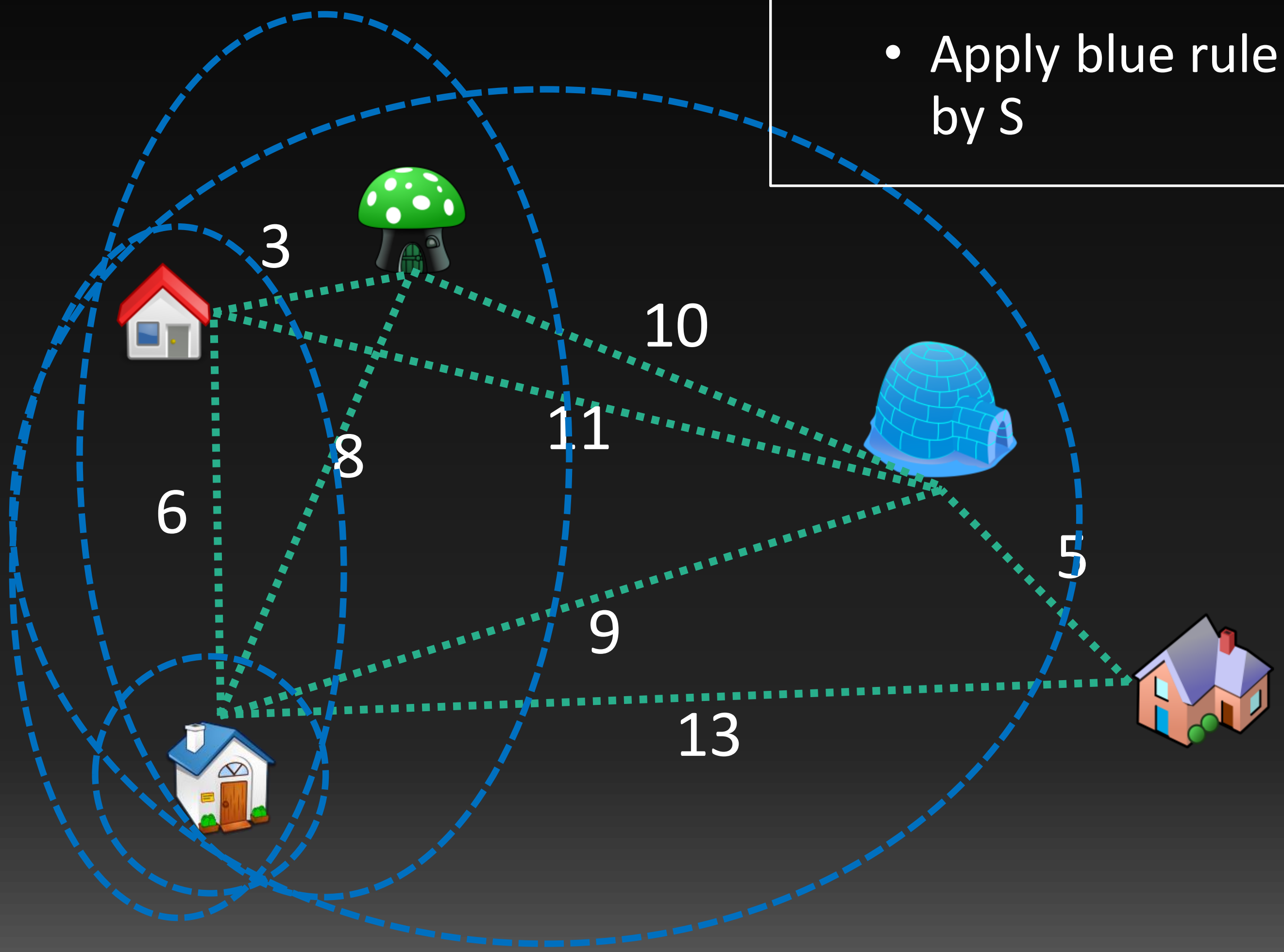
Prim’s algorithm

- Pick an arbitrary root node u
- S = {nodes connected to u by blue edges}
- While $S \neq V$
  - Apply blue rule to cut induced by S



Example

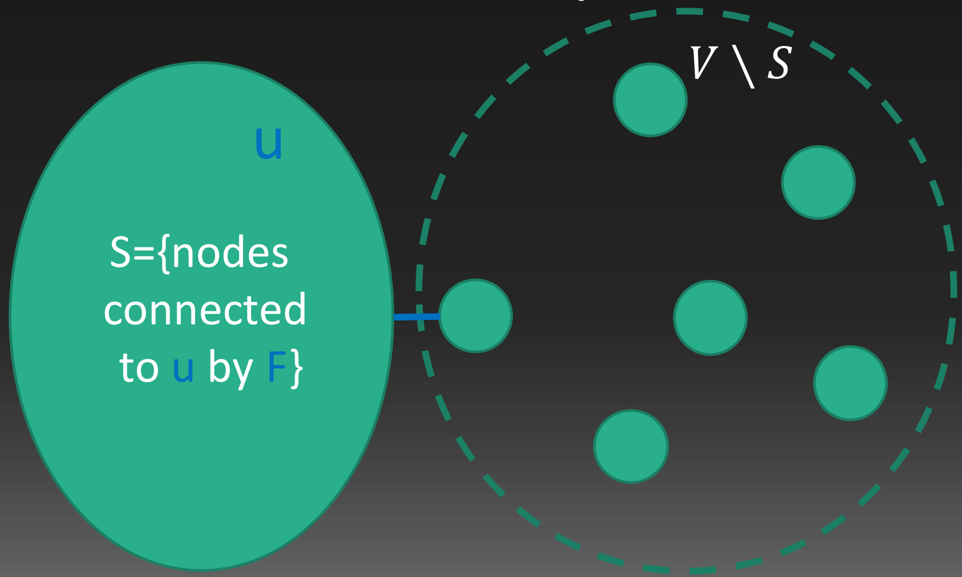
- Pick an arbitrary root node u=
- S = {nodes connected to u by blue edges}
- While $S \neq V$
  - Apply blue rule to cut induced by S



Prim's algorithm

- Pick an arbitrary root node u
- S = {nodes connected to u by blue edges}
- While $S \neq V$
  - Apply blue rule to cut induced by S

Need to maintain collection of edges and find minimum

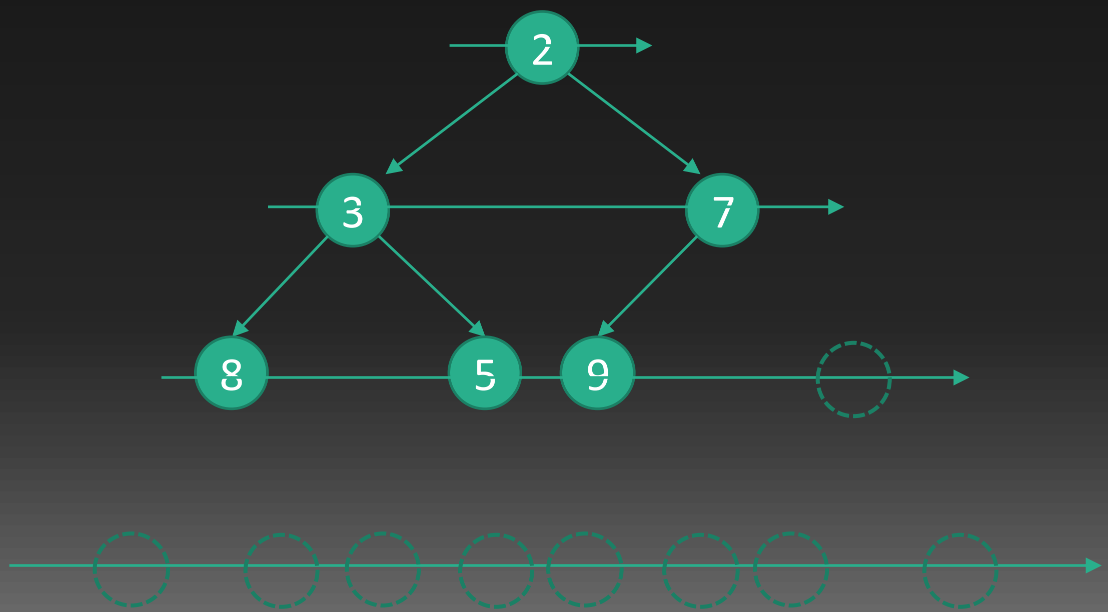


Priority queue

- Data structure maintaining collection of pairs (id, key)
- **Insert**: Insert a new pair (id, key) into the queue
- **Find-min**: Find the pair with minimum key
- **Extract-min**: Find the pair with minimum key and remove it from the queue
- **Decrease-key(id, D)**: Decrease the key of element id to D

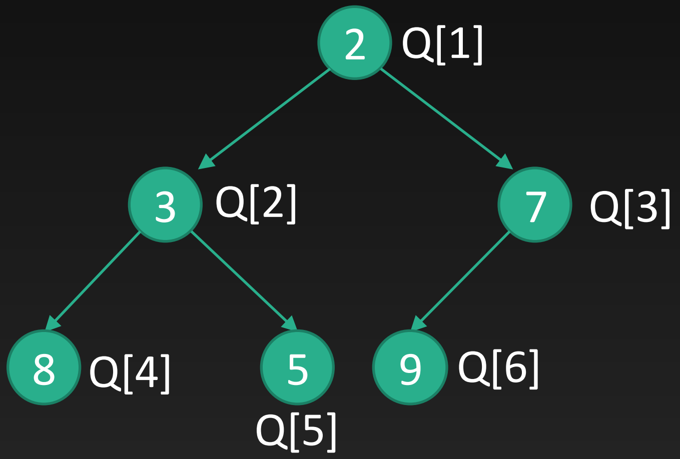
Binary heap

- Full binary tree
- Each node stores an (id, key) pair
- Key of parent is no larger than keys of children



Implicit binary heap

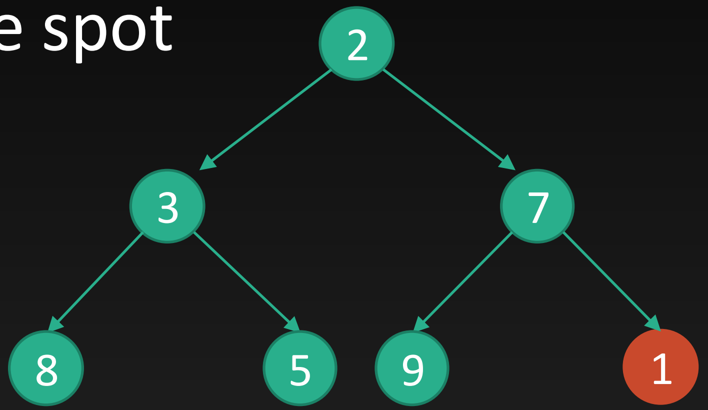
- Store as array Q[1…n]
- The children of node i are nodes 2i and 2i+1



Index	1	2	3	4	5	6
Key	2	3	7	8	5	9

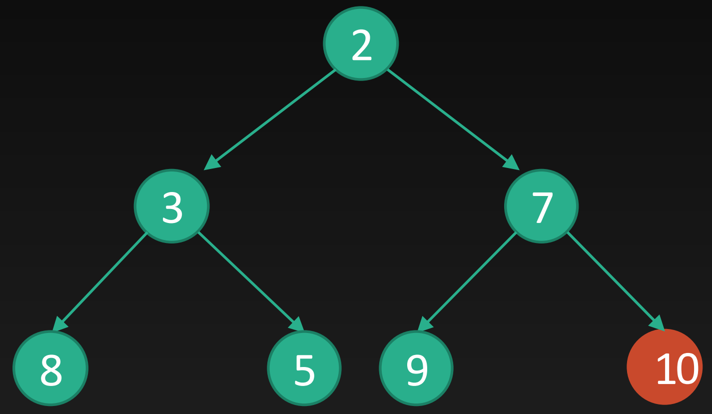
Insert

- Put new key at next available spot
- Bubble up to maintain heap property
- Insert takes O(log n) time



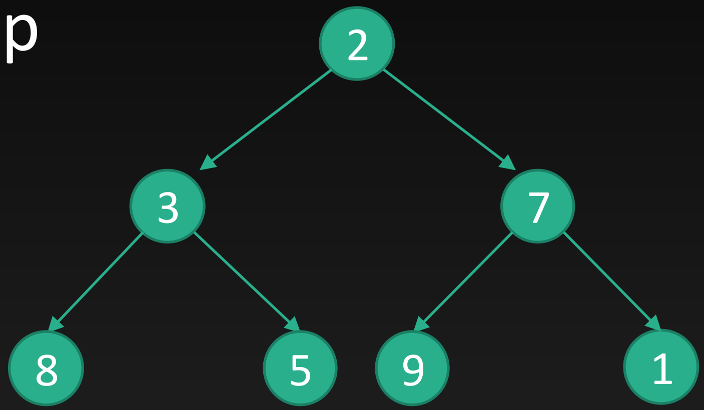
Decrease-key

- Bubble up to maintain heap property
- Decrease-key takes O(log n) time



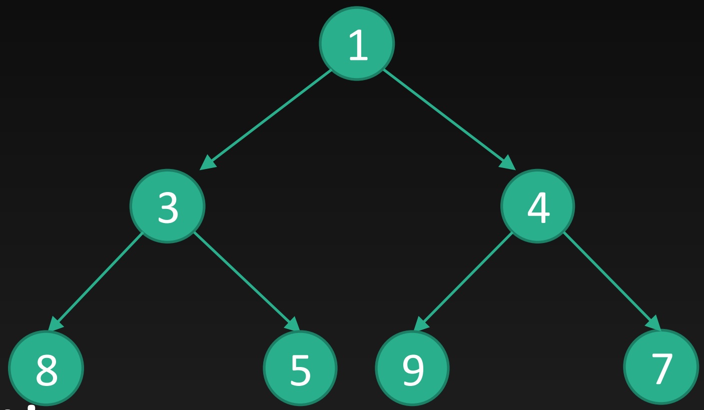
Find-min

- Minimum is always at the top of the heap
- Find-min runs in O(1)



Extract-min

- Remove top node
- Put bottom node at the top
- Bubble down to maintain heap property
- Extract-min runs in O(log n) time

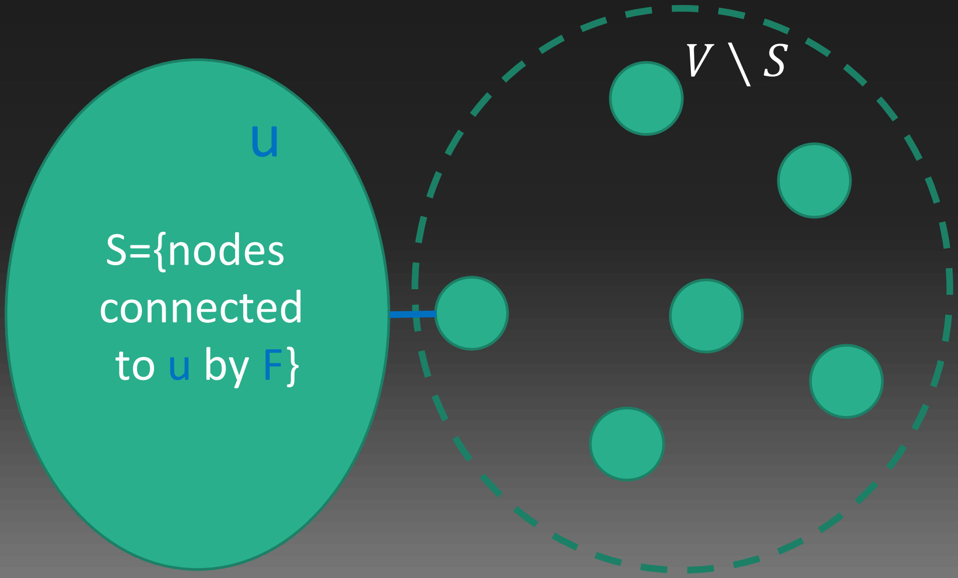


Running time of heap

Operation	Binary heap	Fibonacci heap
Insert	O(log n)	O(1)
Find-min	O(1)	O(1)
Extract-min	O(log n)	O(log n)
Decrease-key	O(log n)	O(1) (amortized)

Prim's algorithm

- Pick root node u
- S = {nodes connected to u by blue edges}
- While $S \neq V$
  - Find min weight edge between S and $V \setminus S$ and color it blue
  - Update S (new edges between S and $V \setminus S$)



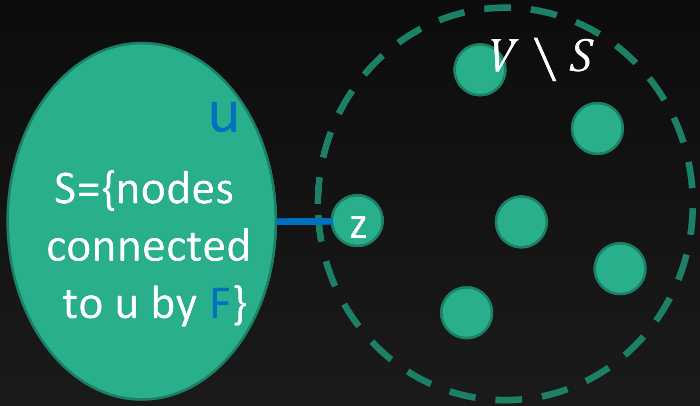
Implementing Prim's algorithm

key(v) = min weight edge between v and S

- $Q = \emptyset$, $F = \emptyset$
- Pick start node u, insert (u, 0) into Q
- Insert (v, ∞) into Q for all vertices $v \neq u$
- Set $pred(v) = u$ for all vertices v
- While $Q \neq \emptyset$

V times
  - $z \leftarrow ExtractMin(Q)$ ← O(log V)
  - $F \leftarrow F \cup \{(z, pred(z))\}$
  - For $v \in adjacent(z)$
    - If $v \in Q$ and $key(v) > w(z, v)$

E times
      - DecreaseKey$(v, w(z, v))$ ← O(log V)
      - $pred(v) \leftarrow z$



Find min weight edge between S and $V \setminus S$ and color it blue

Update S (new edges between S and $V \setminus S$)

O((V+E)log V) time