



Algorithms and Data Structures

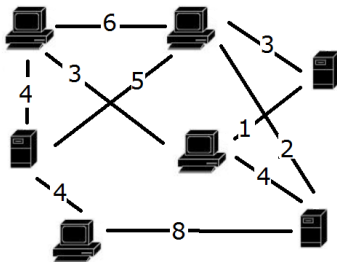
Lecture 17 Minimal Spanning Trees

Jiamou Liu
The University of Auckland



A Typical Network Problem (in 1950s)

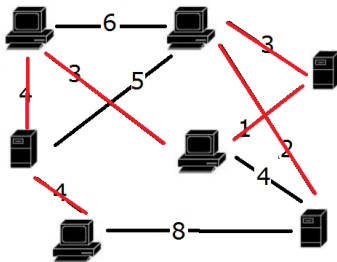
Network a collection of computers while minimizing cost



- All computers must be connected
- We only use possible links
- Avoid cycles

A Typical Network Problem (in 1950s)

Network a collection of computers while minimizing cost

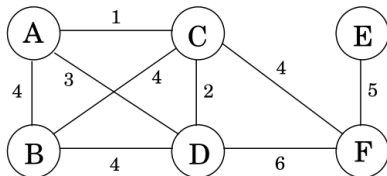


- All computers must be connected
- We only use possible links
- Avoid cycles

Spanning Trees

Spanning Trees

- A graph G is **connected** if it is a connected component itself.

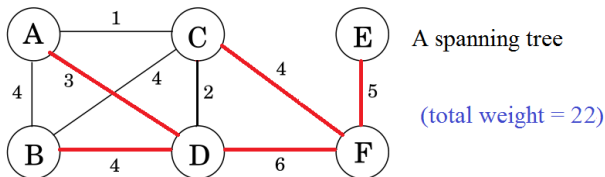


A connected graph

Spanning Trees

Definition

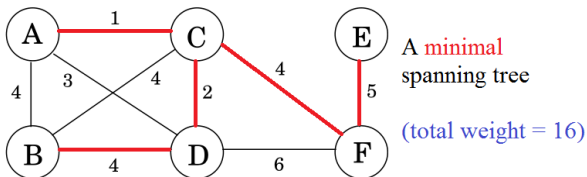
- A graph G is **connected** if it is a connected component itself.
- A **spanning tree** of G is a connected subgraph that contains all nodes in V and no cycles.



Spanning Trees

Definition

- A graph G is **connected** if it is a connected component itself.
- A **spanning tree** of G is a connected subgraph that contains all nodes in V and no cycles.
- A **minimal spanning tree** of a weighted graph is a spanning tree whose total weight is minimal.

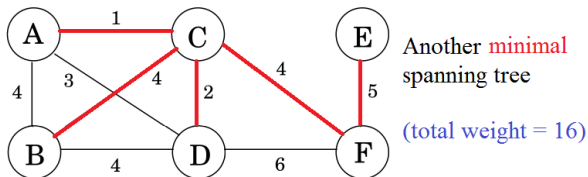


Spanning Trees

Definition

- A graph G is **connected** if it is a connected component itself.
- A **spanning tree** of G is a connected subgraph that contains all nodes in V and no cycles.
- A **minimal spanning tree** of a weighted graph is a spanning tree whose total weight is minimal.

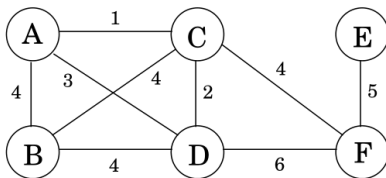
Note: Minimal spanning trees may not be unique.



Minimal Spanning Tree

Minimal Spanning Tree (MST) Problem

- INPUT: A weighted connected (undirected) graph $G = (V, E, w)$
- OUTPUT: A MST of G .



Note: This is once again a **tree construction** problem, just like graph traversal and shortest path problem.

Optimisations in a Weighted Graph

Optimisation Problem

An **optimization problem** contains a **solution set** where each solution has a value.

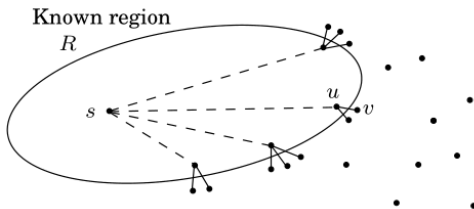
The problem asks to find the solution with the maximal/minimal value (**The optimal solution**).

Optimised Tree Construction in Weighted Graphs

- **Goal:** Construct a tree in the graph that is the optimal solution
- **Optimal Substructure:** If S is an optimal solution, then any subpart of S is also an optimal solution.

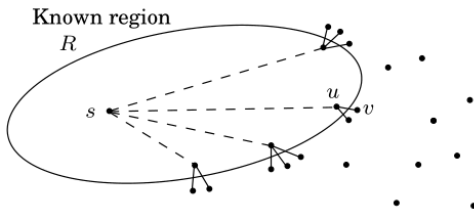
Shortest Path Problem

- **Goal:** Construct a tree in the graph that **minimizes the distance from s to other nodes**.
- **Optimal Substructure:** If $s \rightsquigarrow u \rightsquigarrow v$ is a shortest path, then $s \rightsquigarrow u$ is a shortest path.



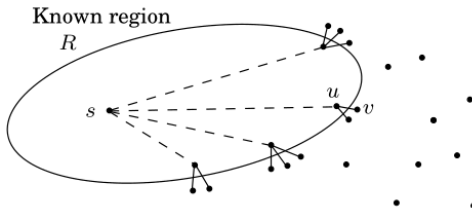
Shortest Path Problem

- **Goal:** Construct a tree in the graph that **minimizes the distance from s to other nodes**.
- **Optimal Substructure:** If $s \rightsquigarrow u \rightsquigarrow v$ is a shortest path, then $s \rightsquigarrow u$ is a shortest path.
- **Greedy Choice:** Suppose R is the “known area”, and v is the node outside of R that minimize the current distance. Then we are safe to add v into R in the next step.



Minimal Spanning Tree Problem

- **Goal:** Construct a tree in the graph that **minimizes the overall weight of the tree**.
- **Optimal Substructure:** If T is an MST of G , and v is a leaf in T , then after removing v from T , we obtain an MST in the subgraph of G with v removed.
- **Greedy Choice:** Can we obtain a similar property as for shortest path problem?

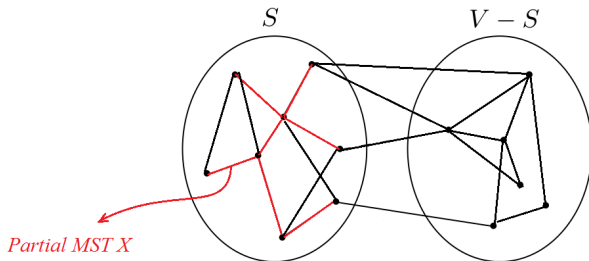


Cut Property

Cut Property (Version 1.0)

Let $G = (V, E, w)$ be a weighted graph.

- We say a **partial MST** of G is a subtree that could lead to an MST.
- Suppose we have constructed a partial MST X on a subset $S \subseteq V$.
- Let e be the lightest edge across the partition between S and $V - S$.
- Then $X \cup \{e\}$ is also a partial MST.

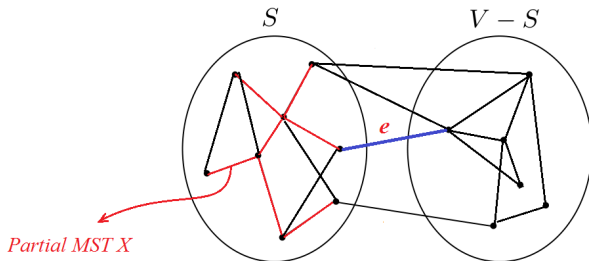


Cut Property

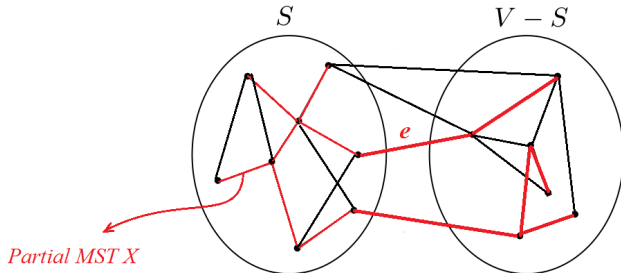
Cut Property (Version 1.0)

Let $G = (V, E, w)$ be a weighted graph.

- We say a **partial MST** of G is a subtree that could lead to an MST.
- Suppose we have constructed a partial MST X on a subset $S \subseteq V$.
- Let e be the lightest edge across the partition between S and $V - S$.
- Then $X \cup \{e\}$ is also a partial MST.

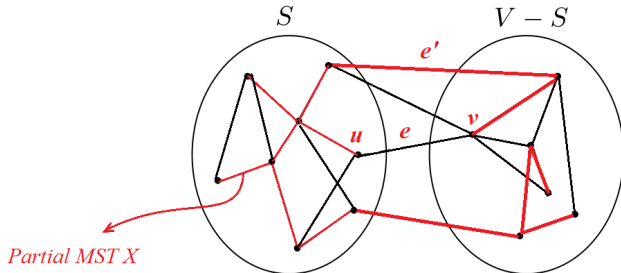


Question. Why does cut property hold?



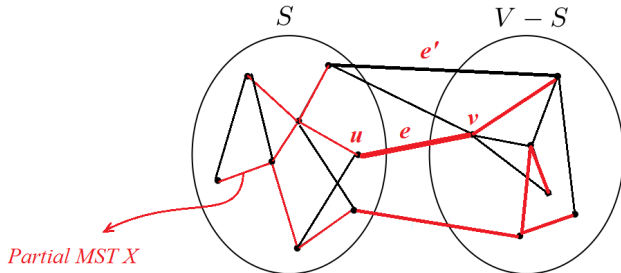
- Since X is a partial MST, there is an MST T that contains X .
- Suppose T contains e . Then we are done.

Question. Why does cut property hold?



- Since X is a partial MST, there is an MST T that contains X .
- Suppose T contains e . Then we are done.
- Suppose T uses e' in the $(S, V - S)$ -partition to connect to v .

Question. Why does cut property hold?

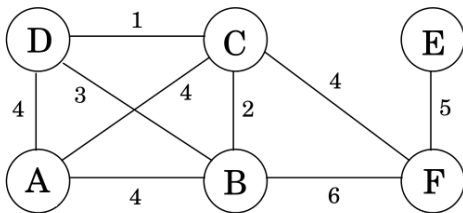


- Since X is a partial MST, there is an MST T that contains X .
- Suppose T contains e . Then we are done.
- Suppose T uses e' in the $(S, V - S)$ -partition to connect to v .
- Then $(T \setminus \{e'\}) \cup \{e\}$ is an MST.
- So $X \cup \{e\}$ is a partial MST.

Prim's Algorithm

Idea:

- Find MST in a similar way as Dijkstra's algorithm.
- Maintain a **set** for the known region.
- Maintain $prev(u)$ for every node u to store the tree.
- Maintain a **priority queue** storing the candidate edges weights.

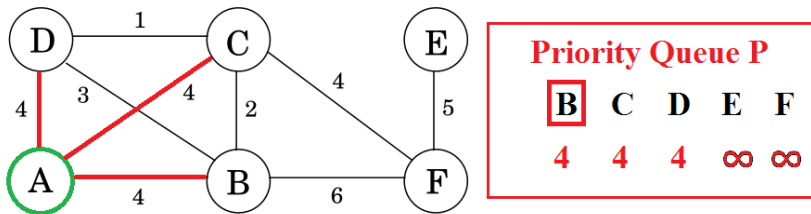


Priority Queue P						
A	B	C	D	E	F	
0	∞	∞	∞	∞	∞	

Prim's Algorithm

Idea:

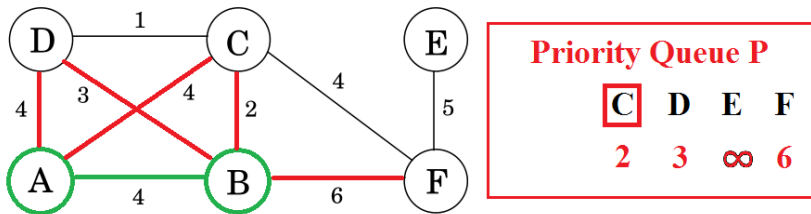
- Find MST in a similar way as Dijkstra's algorithm.
- Maintain a **set** for the known region.
- Maintain $prev(u)$ for every node u to store the tree.
- Maintain a **priority queue** storing the candidate edges weights.



Prim's Algorithm

Idea:

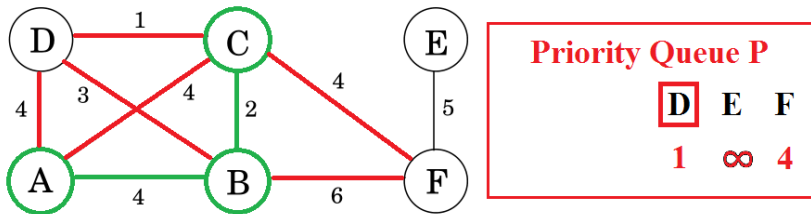
- Find MST in a similar way as Dijkstra's algorithm.
- Maintain a **set** for the known region.
- Maintain $prev(u)$ for every node u to store the tree.
- Maintain a **priority queue** storing the candidate edges weights.



Prim's Algorithm

Idea:

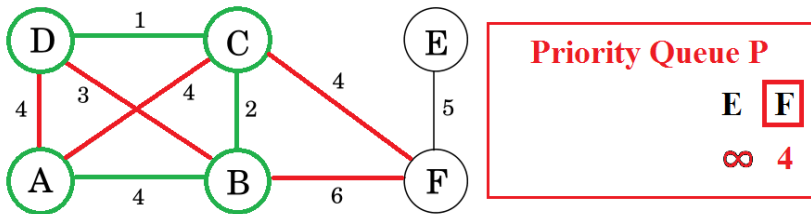
- Find MST in a similar way as Dijkstra's algorithm.
- Maintain a **set** for the known region.
- Maintain $prev(u)$ for every node u to store the tree.
- Maintain a **priority queue** storing the candidate edges weights.



Prim's Algorithm

Idea:

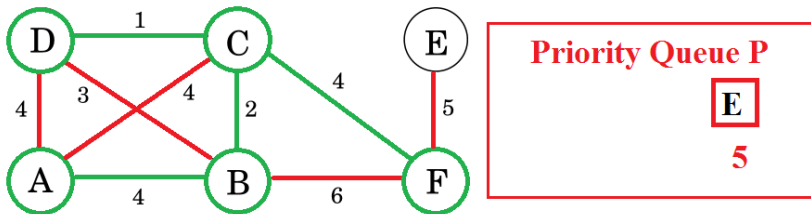
- Find MST in a similar way as Dijkstra's algorithm.
- Maintain a **set** for the known region.
- Maintain $prev(u)$ for every node u to store the tree.
- Maintain a **priority queue** storing the candidate edges weights.



Prim's Algorithm

Idea:

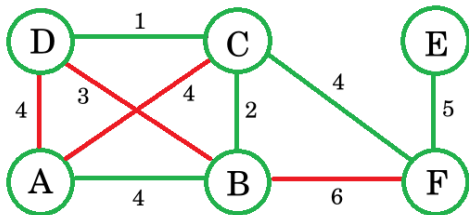
- Find MST in a similar way as Dijkstra's algorithm.
- Maintain a **set** for the known region.
- Maintain $prev(u)$ for every node u to store the tree.
- Maintain a **priority queue** storing the candidate edges weights.



Prim's Algorithm

Idea:

- Find MST in a similar way as Dijkstra's algorithm.
- Maintain a **set** for the known region.
- Maintain $prev(u)$ for every node u to store the tree.
- Maintain a **priority queue** storing the candidate edges weights.



Priority Queue P

Algorithm **MST_Prim**(V, E, w)

1. Let s be the first node in V .
2. Initialize a **set** $R \leftarrow \{s\}$
3. Initialize a **priority queue** P containing $(s, 0)$
4. **for** $u \in V, u \neq s$ **do**
 $prev(u) \leftarrow null$
 $value(u) \leftarrow \infty$
 $P.Insert(u, \infty)$
5. **while** P is not empty **do**
 $u \leftarrow P.DeleteMin()$
 Add u to R
 for $(u, v) \in E$ where $v \notin R$ **do**
 if $weight(u, v) < value(v)$ **then**
 $value(v) \leftarrow weight(u, v)$
 $P.DecreaseKey(v, value(v))$
 $prev(v) \leftarrow u$

Prim's Algorithm: Correctness

Theorem

Let $G = (V, E, w)$ be a weighted graph. After running `MST_Prim(V, E, w)` the tree constructed (as represented by the *prev* pointers) is an MST.

Prim's Algorithm: Correctness

Theorem

Let $G = (V, E, w)$ be a weighted graph. After running `MST_Prim`(V, E, w) the tree constructed (as represented by the *prev* pointers) is an MST.

Proof. We prove by induction the following **loop invariant**: *After each iteration of `while-loop`, we maintain a partial MST.*

□

Prim's Algorithm: Correctness

Theorem

Let $G = (V, E, w)$ be a weighted graph. After running `MST_Prim(V, E, w)` the tree constructed (as represented by the *prev* pointers) is an MST.

Proof. We prove by induction the following **loop invariant**: *After each iteration of **while-loop**, we maintain a partial MST.*

Base Case: Before entering the while-loop, we have a single-node tree, which is a partial MST.

Inductive Step: Suppose after iteration n , we get a partial MST. The cut property guarantees us a partial MST after iteration $n + 1$. \square

Prim's Algorithm: Complexity

Running time

Prim's algorithm runs in exactly the same asymptotic time as Dijkstra's algorithm:

Depending on the implementations of priority queues:

- Lists: $O(n^2)$
- Binary heap/Binomial heap: $O((m + n) \log n)$
- Fibonacci heap: $O(n \log n + m)$

- Spanning tree
- Minimal spanning tree problem as an optimisation problem on weighted graphs
- Cut property
- Prim's algorithm
 - Correctness
 - Complexity

Exercise

Question 1. On the graph below, run Prim's algorithm starting from node 0. Draw the corresponding MST.

