**Minimum Spanning Tree**

*Lecture 16:*

Greedy algorithms:

Greedy algorithms make choices based on immediate rewards rather than looking ahead to see the optimum.

Minimum Spanning Trees:

Given a connected, undirected, weighted graph G = (V, E), we want to find a subgraph T = (V, E’) of G, such that:

* T spans all vertices of G.
* The total weight of the edges in T in minimized.

Such a tree is called a minimum spanning tree of G.

An example of a graph and its MST:

*a*

*b*

*h*

*c*

*d*

*e*

*f*

*g*

*i*

4

8

7

9

10

15

4

2

2

6

1

7

11

8

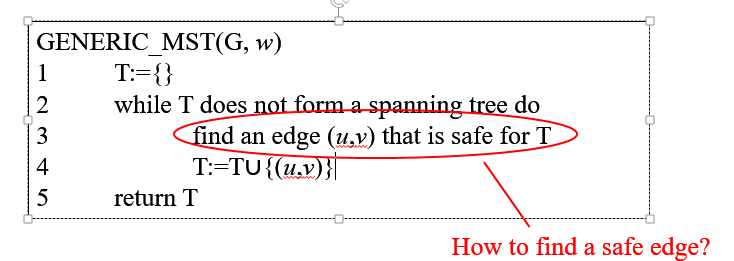
Note:

* The MST may not be unique: replacing (b, c) with (a, h) yields another spanning tree with the same minimum weight.
* However the total weight of these MSTs is the same.

Safe edge:

A safe edge is when it has the **minimum weight** and adding it to the tree does not destroy the tree-structure of the set. (i.e. doesn’t create a loop)

How to grow a Spanning Tree so it becomes a MST:



There are **two** ways:

* **Kruskal’s algorithm:**
  + The set T is a *forest.*
  + The next safe edge is the one with the *least-weight* and *does not form a cycle with edges in T.*
* **Prim’s algorithm:**
  + The set T is a *single tree.*
  + The next safe edge is the one with the *least-weight* and *is connected to edges in T* but *does not form a cycle with them*.

TLDR Kruskal’s doesn’t care if we have a tree or a forest, while Prim’s needs the MST to be a single tree.

Kruskal’s: (look at presentation for slides for a step-by-step.

a

b

h

c

d

e

f

g

i

4

8

7

9

10

14

4

2

2

6

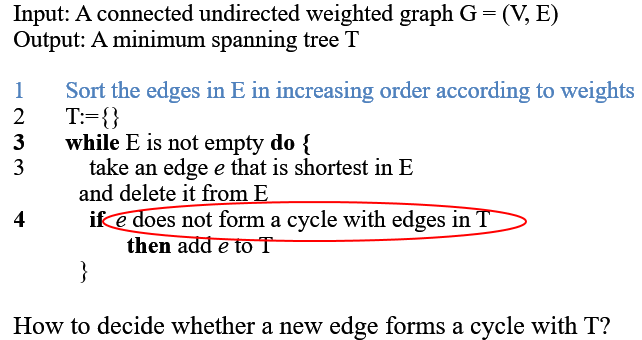
1

7

11

8

The algorithm:



**How to decide if it forms a cycle?**

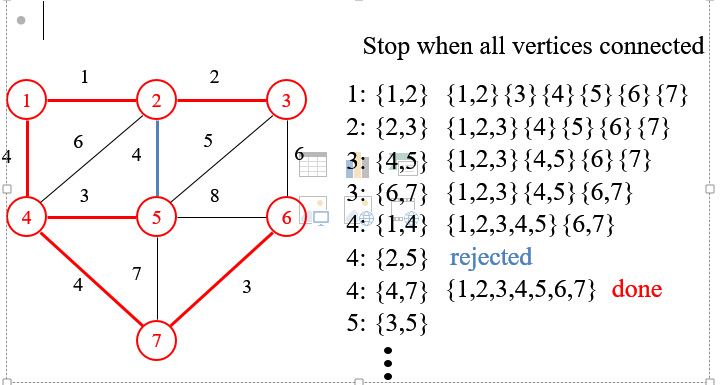
Idea: the new edge forms a cycle with T: the edge has to connect two nodes representing the same subtree.

Thus we need to quickly find which subtree the two vertices of an edge belong to.

We use union-find data structure:

* Initially all nodes are assigned to a separate subtree.
* When two subtrees are combined by an edge we have to perform the union of the two subtrees.

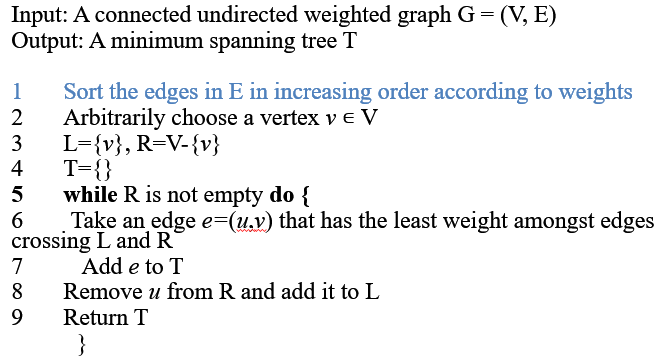
Example: (look at slides for a step-by-step)



**Overview of the algorithm:**

* Correctness:
  + Outputs a spanning tree.
    - T is always a forest.
    - T is spanning: every vertex v, the edge that has the least weight and connects v to T must be considered in the iterations.
  + Minimum weight:
    - Proof by induction.
* **Run-time:** O(|E| log|E|)
  + Sorting the edges takes more time than the run-time.

Prim’s algorithm:



**Execution of Prim’s Algorithm:** (check slides for step-by-step)

a

b

h

c

d

e

f

g

i

4

8

7

9

10

14

4

2

2

6

1

7

11

8

Implementation of the algorithm: instead of initializing with many subtrees, there are only two. One is connected to the root vertex, the other contains the rest of the vertices. (Unlike Kruskal’s algorithm, where you initialize with trees that are as many as the vertices?)

Dijkstra’s Algorithm: Finds the shortest path

Given a connected, undirected, non-negative weighted graph, finds the shortest path from a source *s* to all other nodes.

How it works:

* For each node *u* maintains the distance d[u] from *s* (initialized to be infinity).
* When we add a new node v to T, update the distance of v’s unchosen neighbors if going through v is shorter than their current distance.

Example: (step-by-step in slides)

