COMP108
Data Structures and Algorithms

Greedy Algorithm (Part II Minimum Spanning Tree)

Professor Prudence Wong

pwong@liverpool.ac.uk

2022-23

Minimum Spanning Tree (MST)

Given an undirected connected graph G

- The edges are labelled by weight

Spanning tree of G

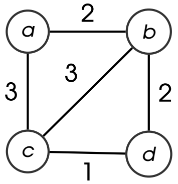
- a tree containing all vertices in G

Minimum spanning tree of G

- a spanning tree of G with minimum weight

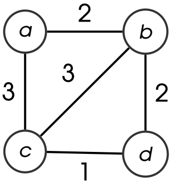
Examples

Graph G (edge label is weight)

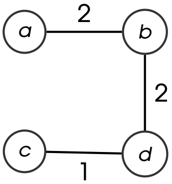


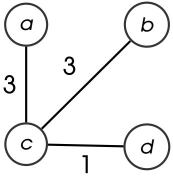
Examples

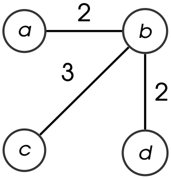
Graph G (edge label is weight)



Spanning trees of G

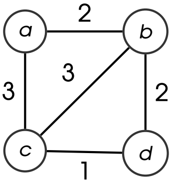




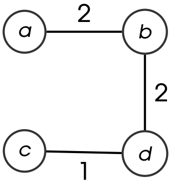


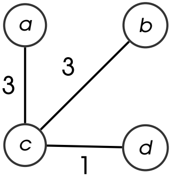
Examples

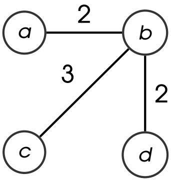
Graph G (edge label is weight)



Spanning trees of G



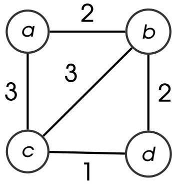




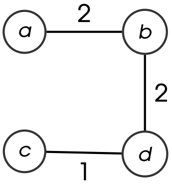
Which is MST?

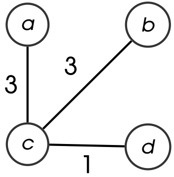
Examples

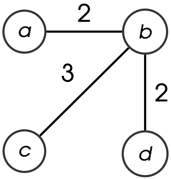
Graph G (edge label is weight)



Spanning trees of G



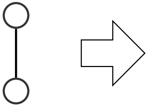




Which is MST?

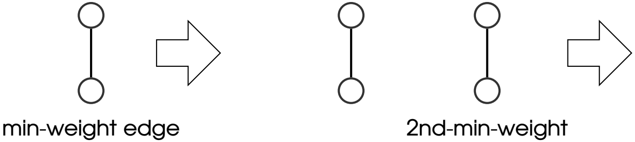
How many possible spanning trees?

Idea of Kruskal's algorithm - MST

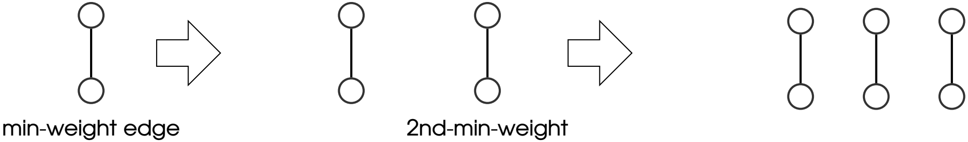


min-weight edge

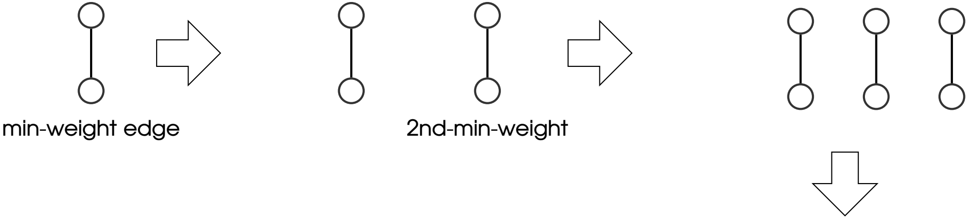
Idea of Kruskal's algorithm - MST



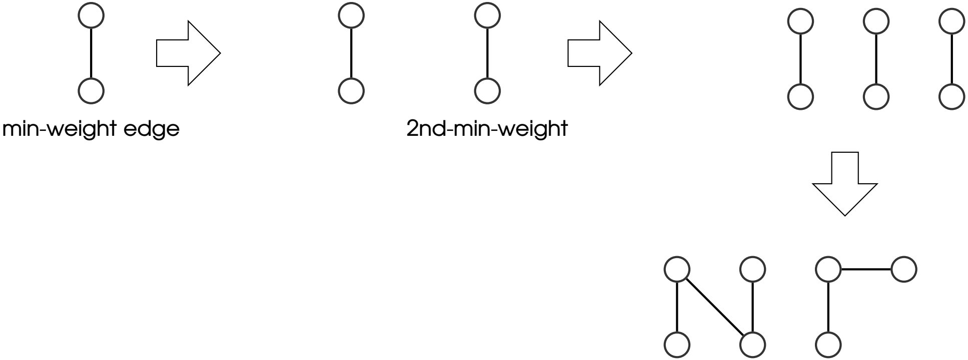
Idea of Kruskal's algorithm - MST



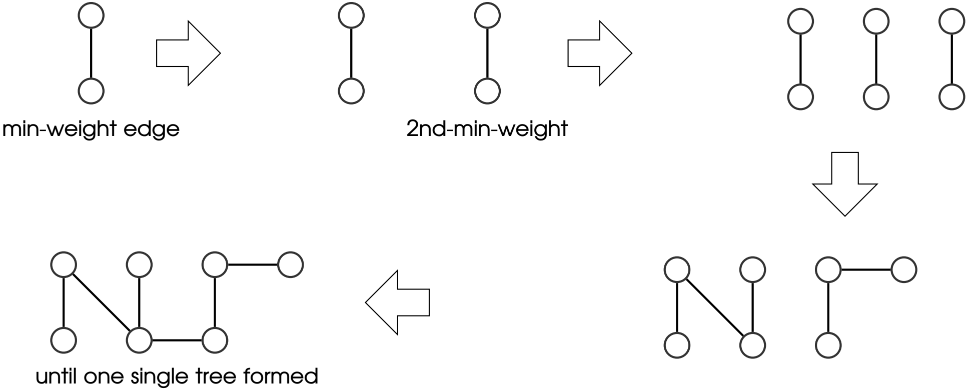
Idea of Kruskal's algorithm - MST



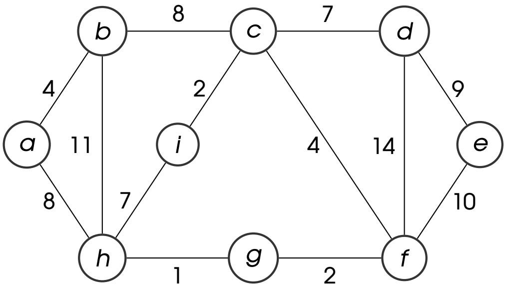
Idea of Kruskal's algorithm - MST



Idea of Kruskal's algorithm - MST



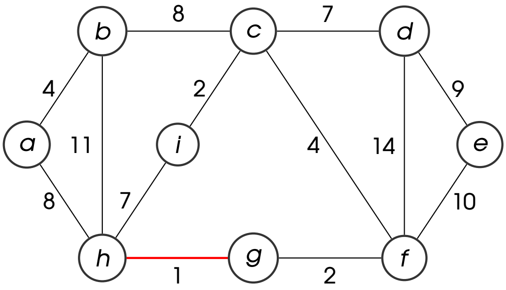
Kruskal's algorithm - MST



Arrange edges from smallest to largest weight

	(g, h)	1
	(c, i)	2
	(f, g)	2
	(a, b)	4
	(c, f)	4
	(c, d)	7
	(h, i)	7
	(a, h)	8
	(b, c)	8
	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

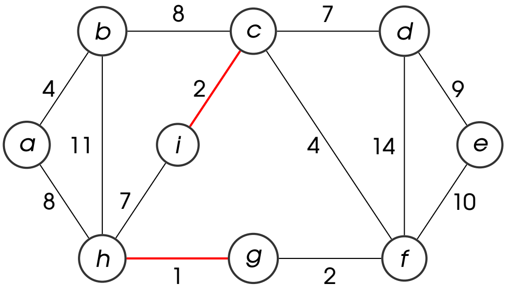
Kruskal's algorithm - MST



Choose the minimum weight edge

✓	(g, h)	1
	(c, i)	2
	(f, g)	2
	(a, b)	4
	(c, f)	4
	(c, d)	7
	(h, i)	7
	(a, h)	8
	(b, c)	8
	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

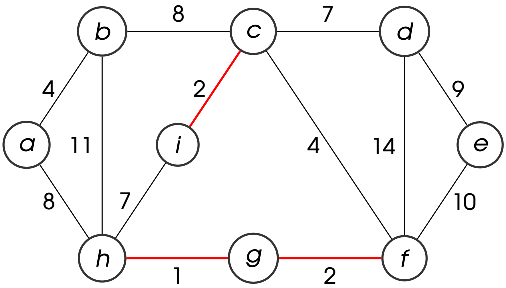
Kruskal's algorithm - MST



✓	(g, h)	1
✓	(c, i)	2
	(f, g)	2
	(a, b)	4
	(c, f)	4
	(c, d)	7
	(h, i)	7
	(a, h)	8
	(b, c)	8
	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

Choose the next minimum weight edge

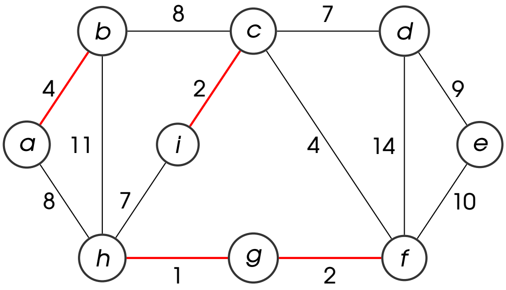
Kruskal's algorithm - MST



✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
	(a, b)	4
	(c, f)	4
	(c, d)	7
	(h, i)	7
	(a, h)	8
	(b, c)	8
	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

Continue as long as no cycle forms

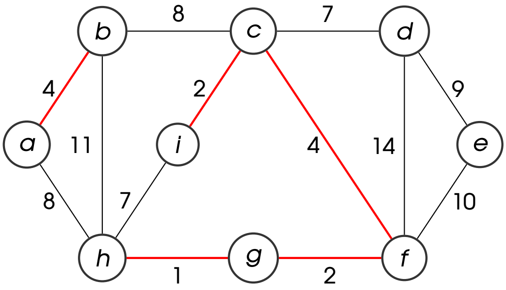
Kruskal's algorithm - MST



Continue as long as no cycle forms

✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
	(c, f)	4
	(c, d)	7
	(h, i)	7
	(a, h)	8
	(b, c)	8
	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

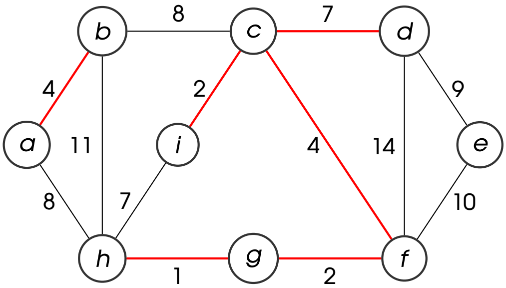
Kruskal's algorithm - MST



Continue as long as no cycle forms

✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
✓	(c, f)	4
	(c, d)	7
	(h, i)	7
	(a, h)	8
	(b, c)	8
	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

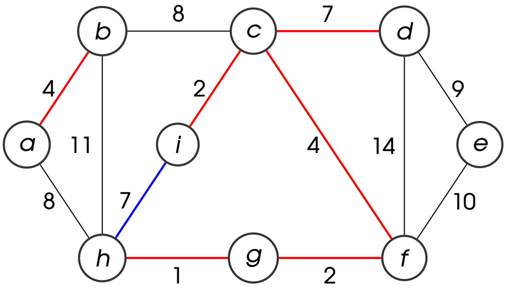
Kruskal's algorithm - MST



Continue as long as no cycle forms

✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
✓	(c, f)	4
✓	(c, d)	7
	(h, i)	7
	(a, h)	8
	(b, c)	8
	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

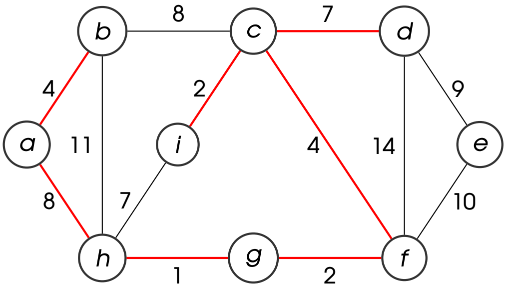
Kruskal's algorithm - MST



✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
✓	(c, f)	4
✓	(c, d)	7
	(h, i)	7
	(a, h)	8
	(b, c)	8
	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

next edge cannot be included, otherwise, cycle is formed

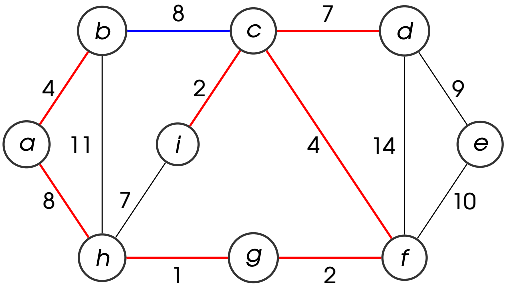
Kruskal's algorithm - MST



Choose the next minimum weight edge

✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
✓	(c, f)	4
✓	(c, d)	7
	(h, i)	7
✓	(a, h)	8
	(b, c)	8
	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

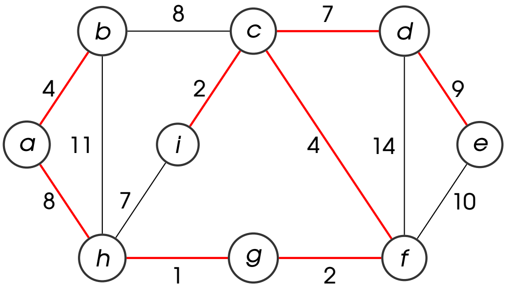
Kruskal's algorithm - MST



✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
✓	(c, f)	4
✓	(c, d)	7
	(h, i)	7
✓	(a, h)	8
	(b, c)	8
	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

next edge cannot be included, otherwise, cycle is formed

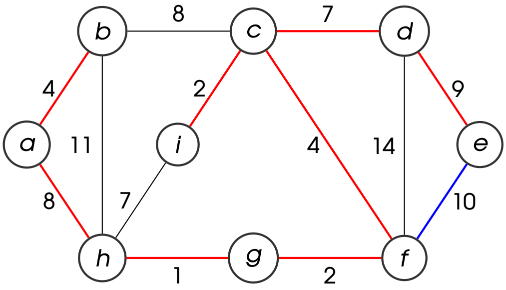
Kruskal's algorithm - MST



Choose the next minimum weight edge

✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
✓	(c, f)	4
✓	(c, d)	7
	(h, i)	7
✓	(a, h)	8
	(b, c)	8
✓	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

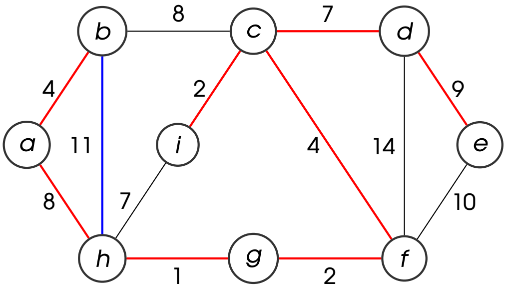
Kruskal's algorithm - MST



✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
✓	(c, f)	4
✓	(c, d)	7
	(h, i)	7
✓	(a, h)	8
	(b, c)	8
✓	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

next edge cannot be included, otherwise, cycle is formed

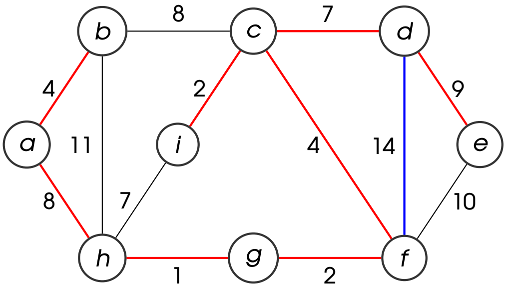
Kruskal's algorithm - MST



✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
✓	(c, f)	4
✓	(c, d)	7
	(h, i)	7
✓	(a, h)	8
	(b, c)	8
✓	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

next edge cannot be included, otherwise, cycle is formed

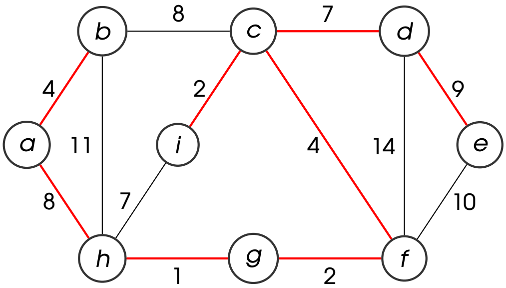
Kruskal's algorithm - MST



✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
✓	(c, f)	4
✓	(c, d)	7
	(h, i)	7
✓	(a, h)	8
	(b, c)	8
✓	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

next edge cannot be included, otherwise, cycle is formed

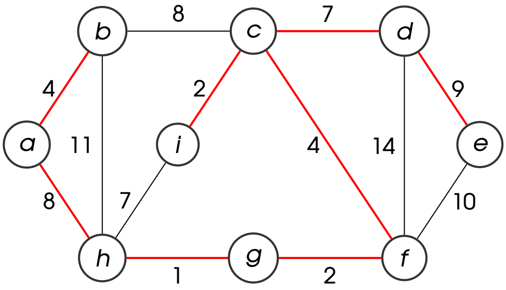
Kruskal's algorithm - MST



✓	(g,h)	1
✓	(c,i)	2
✓	(f,g)	2
✓	(a,b)	4
✓	(c,f)	4
✓	(c,d)	7
	(h,i)	7
✓	(a,h)	8
	(b,c)	8
✓	(d,e)	9
	(e,f)	10
	(b,h)	11
	(d,f)	14

MST is found when all edges are examined

Kruskal's algorithm - MST



✓	(g, h)	1
✓	(c, i)	2
✓	(f, g)	2
✓	(a, b)	4
✓	(c, f)	4
✓	(c, d)	7
	(h, i)	7
✓	(a, h)	8
	(b, c)	8
✓	(d, e)	9
	(e, f)	10
	(b, h)	11
	(d, f)	14

Order of edges selected $(g, h), (c, i), (f, g), (a, b), (c, f), (c, d), (a, h), (d, e)$

Kruskal's algorithm - MST

Kruskal's algorithm is **greedy** in the sense that

- it always attempt to select the **smallest** weight edge to be included in the MST

Pseudo code

// Given an undirected connected graph $G = (V, E)$

$T \leftarrow \emptyset$

$E' \leftarrow E$

while $E' \neq \emptyset$ do

begin

end

Pseudo code

// Given an undirected connected graph $G = (V, E)$

$T \leftarrow \emptyset$

$E' \leftarrow E$

while $E' \neq \emptyset$ do

begin

  pick an edge e in E' with minimum weight

end

Pseudo code

// Given an undirected connected graph $G = (V, E)$

$T \leftarrow \emptyset$

$E' \leftarrow E$

while $E' \neq \emptyset$ do

begin

pick an edge $\boldsymbol{e}$ in E' with minimum weight

if adding e to T does **not** form cycle then

add e to T, i.e., $\boldsymbol{T \leftarrow T \cup \{e\}}$

end

Pseudo code

// Given an undirected connected graph $G = (V, E)$

$T \leftarrow \emptyset$

$E' \leftarrow E$

while $E' \neq \emptyset$ do

begin

  pick an edge $\boldsymbol{e}$ in E' with minimum weight

  if adding e to T does **not** form cycle then

    add e to T, i.e., $\boldsymbol{T \leftarrow T \cup \{e\}}$

  remove e from E', i.e., $\boldsymbol{E' \leftarrow E' \setminus \{e\}}$

end

Pseudo code

// Given an undirected connected graph $G = (V, E)$

```
T ← ∅
E' ← E
while E' ≠ ∅ do
begin
    pick an edge e in E' with minimum weight
    if adding e to T does not form cycle then
        add e to T, i.e., T ← T ∪ {e}
    remove e from E', i.e., E' ← E' \ {e}
end
```

How to determine if cycle is formed?

How to determine if cycle is formed?

n is # vertices; m is # edges

When we consider a new edge $e = (u, v)$, there may be three cases:

if u & v are both non-colored, then edge can be selected and given a new color;

O(1) **1.** at most one of u and v is end point of an already chosen edge

if u or v are not colored (one of them is colored, the other isn't), can select, give uncolor vertex same color as other vertex

O(1) **2.** u and v both belong to the same partial tree that has been already chosen

if u & v are the same color, then can't select

O(n) **3.** u and v belong to two separate partial trees that have been already chosen

if u & v have different colors C1 and C2, then can select, but we need to recolor so that every vertices of C2 to C1

Use coloring

n is # vertices; m is # edges

Pseudo code

// Given an undirected connected graph $G = (V, E)$ sorting before: O(m log m)

$T \leftarrow \emptyset$ O(1)

$E' \leftarrow E$ O(1)

while $E' \neq \emptyset$ do O(m)

begin

pick an edge e in E' with minimum weight pre-sorting: O(1)

Time complexity?

if adding e to T does **not** form cycle then O(n)

add e to T, i.e., $T \leftarrow T \cup \{e\}$ O(1)

remove e from E', i.e., $E' \leftarrow E' \setminus \{e\}$ O(1)

end

O(nm) => polynomial

Summary

Summary: Kruskal's algorithm for Minimum Spanning Tree

Next: Dijkstra's algorithm for Single-Source Shortest-Paths

For note taking