ITP20001/ECE20010 Data Structures

Chapter 6

- *Graph, Digraph*
- *Minimum Spanning Tree (MST)*
  - *Introduction*
  - *greedy algorithm*
  - *Kruskals's algorithm*
  - *Prim's algorithm*

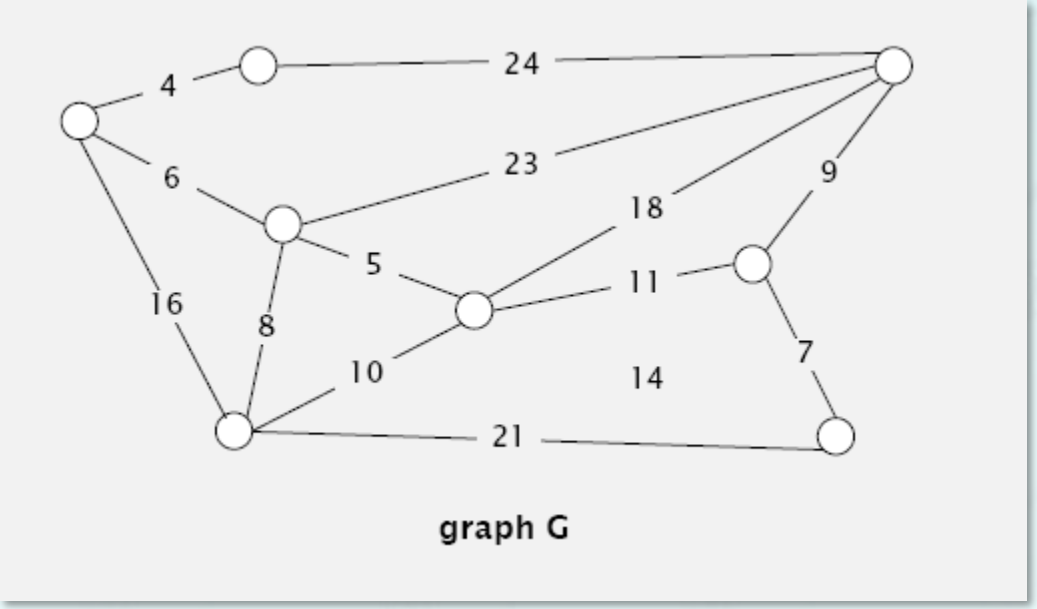
Major references:

1. Fundamentals of Data Structures by Horowitz, Sahni, Anderson-Freed,
2. Algorithms 4th edition - Part 1 & Part 2 by Robert Sedgewick and Kevin Wayne
3. Wikipedia and many resources available from internet

Prof. Youngsup Kim, idebtor@gmail.com, Data Structures, CSEE Dept., Handong Global University

Minimum spanning tree

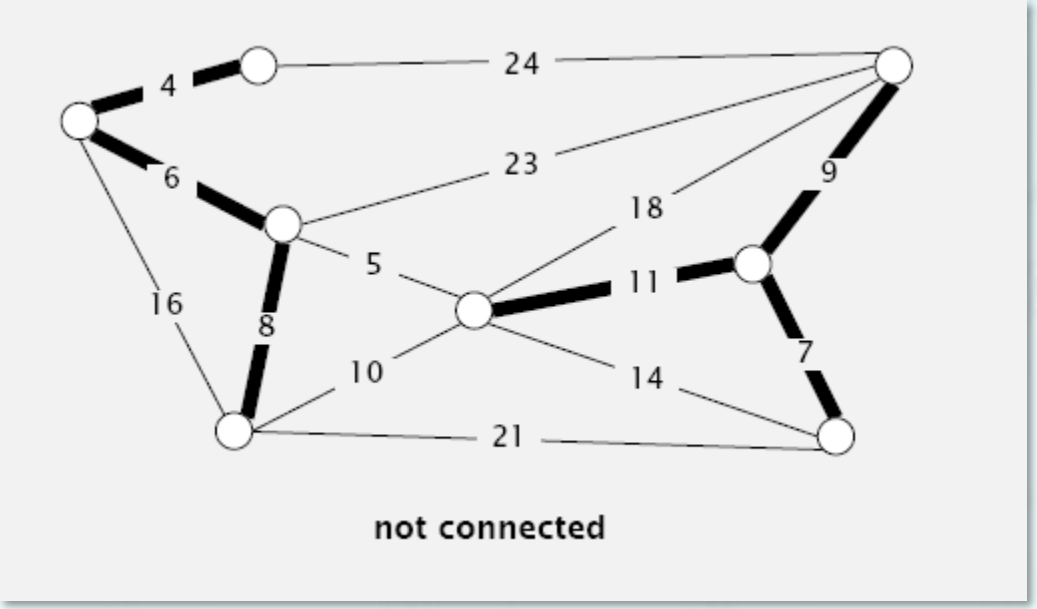
Given: Undirected graph G with positive edge weights (connected)
Definition: A spanning tree of G is a subgraph T that is both a tree (**connected and acyclic**) and spanning (**includes all of the vertices**).
Goal: **Find a min weight spanning tree.**



graph G

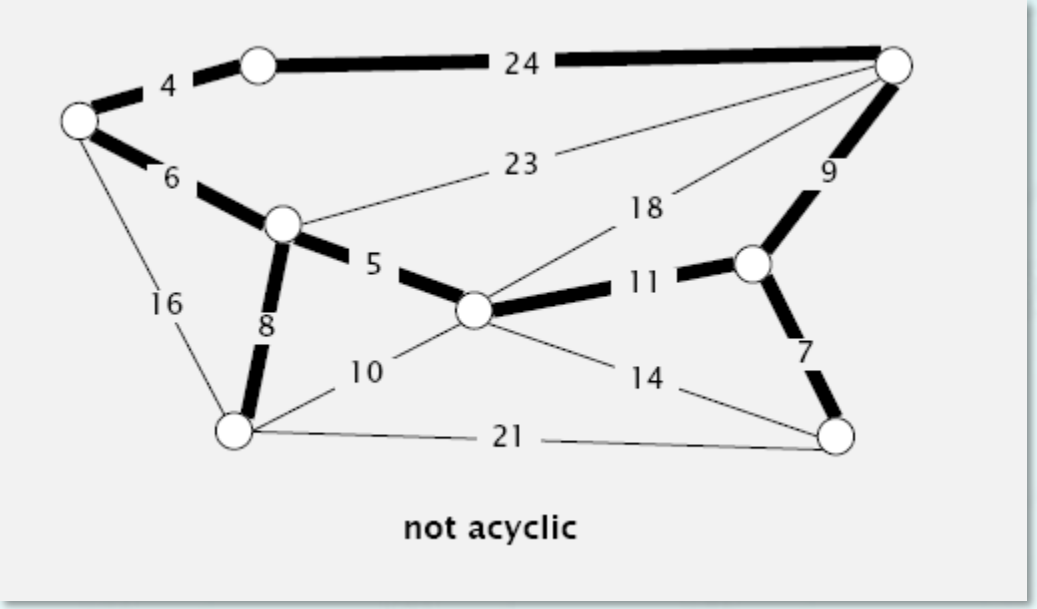
Minimum spanning tree

Given: Undirected graph G with positive edge weights (connected)
Definition: A spanning tree of G is a subgraph T that is both a tree (connected and acyclic) and spanning (includes all of the vertices).
Goal: Find a min weight spanning tree.



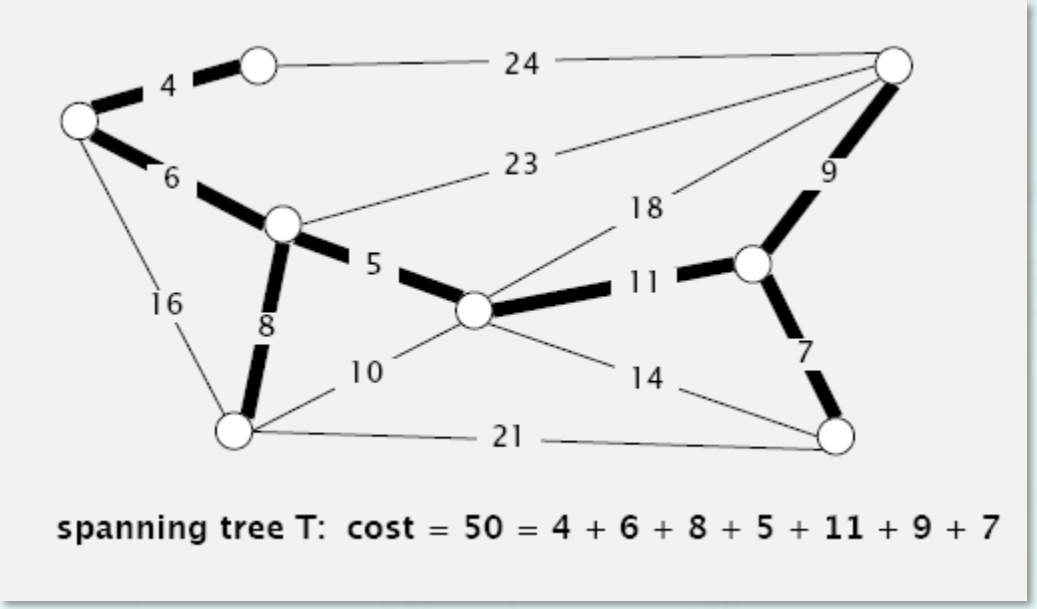
Minimum spanning tree

Given: Undirected graph G with positive edge weights (connected)
Definition: A spanning tree of G is a subgraph T that is both a tree (connected and acyclic) and spanning (includes all of the vertices).
Goal: Find a min weight spanning tree.



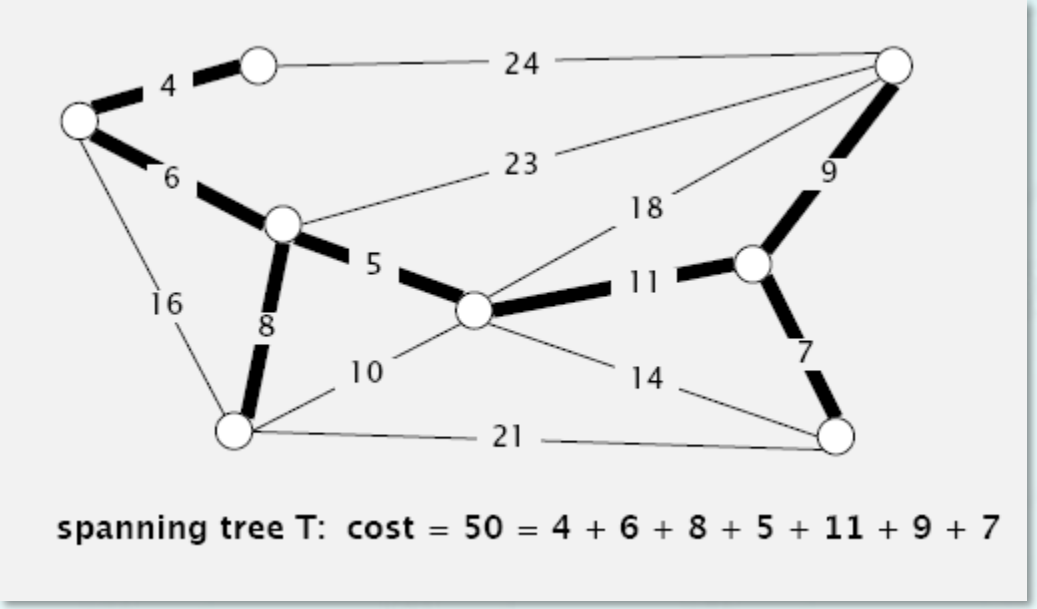
Minimum spanning tree

Given: Undirected graph G with positive edge weights (connected)
Definition: A spanning tree of G is a subgraph T that is both a tree (connected and acyclic) and spanning (includes all of the vertices).
Goal: Find a min weight spanning tree.



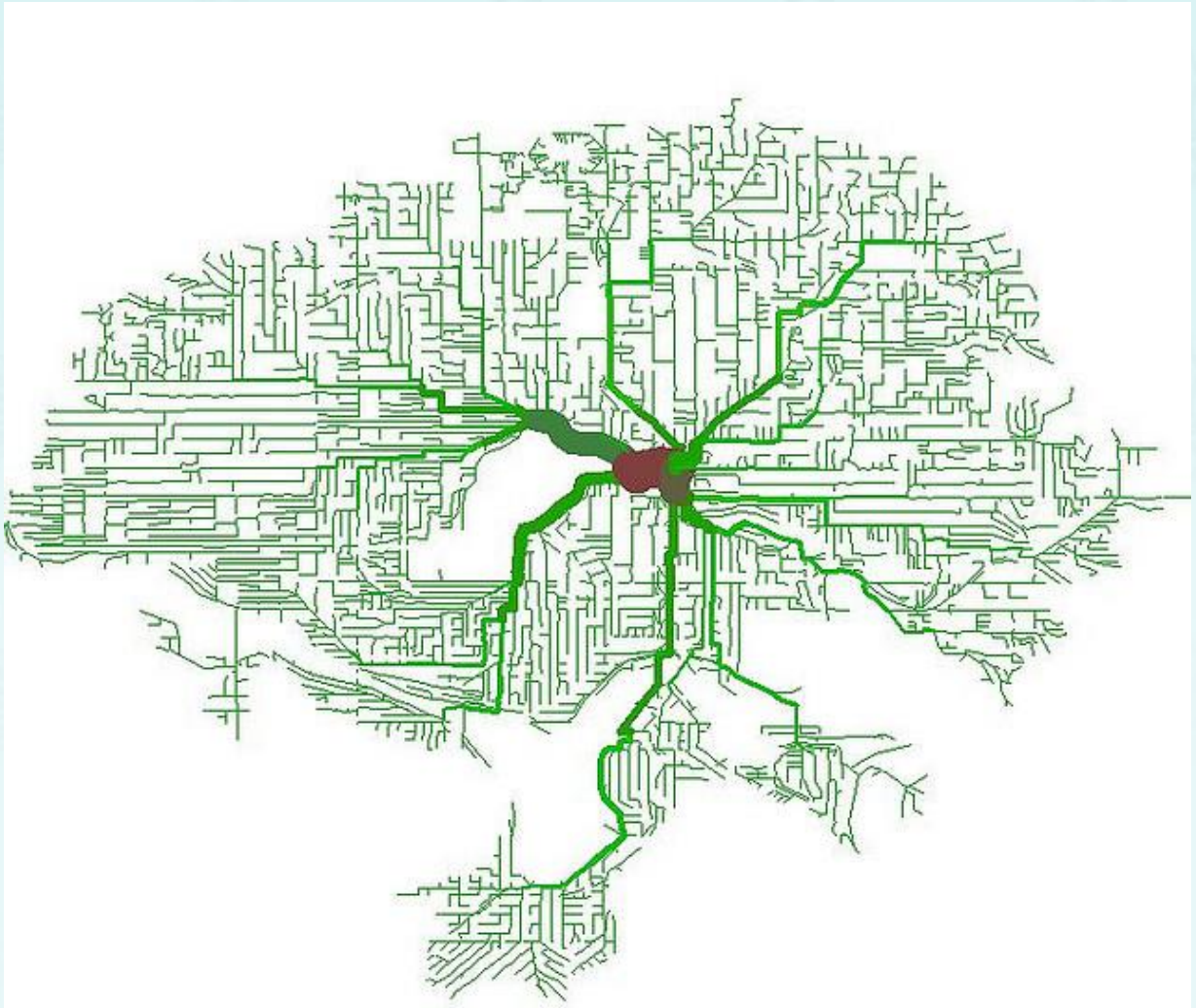
Minimum spanning tree

Given: Undirected graph G with positive edge weights (connected)
Definition: A spanning tree of G is a subgraph T that is both a tree (connected and acyclic) and spanning (includes all of the vertices).
Goal: Find a min weight spanning tree.



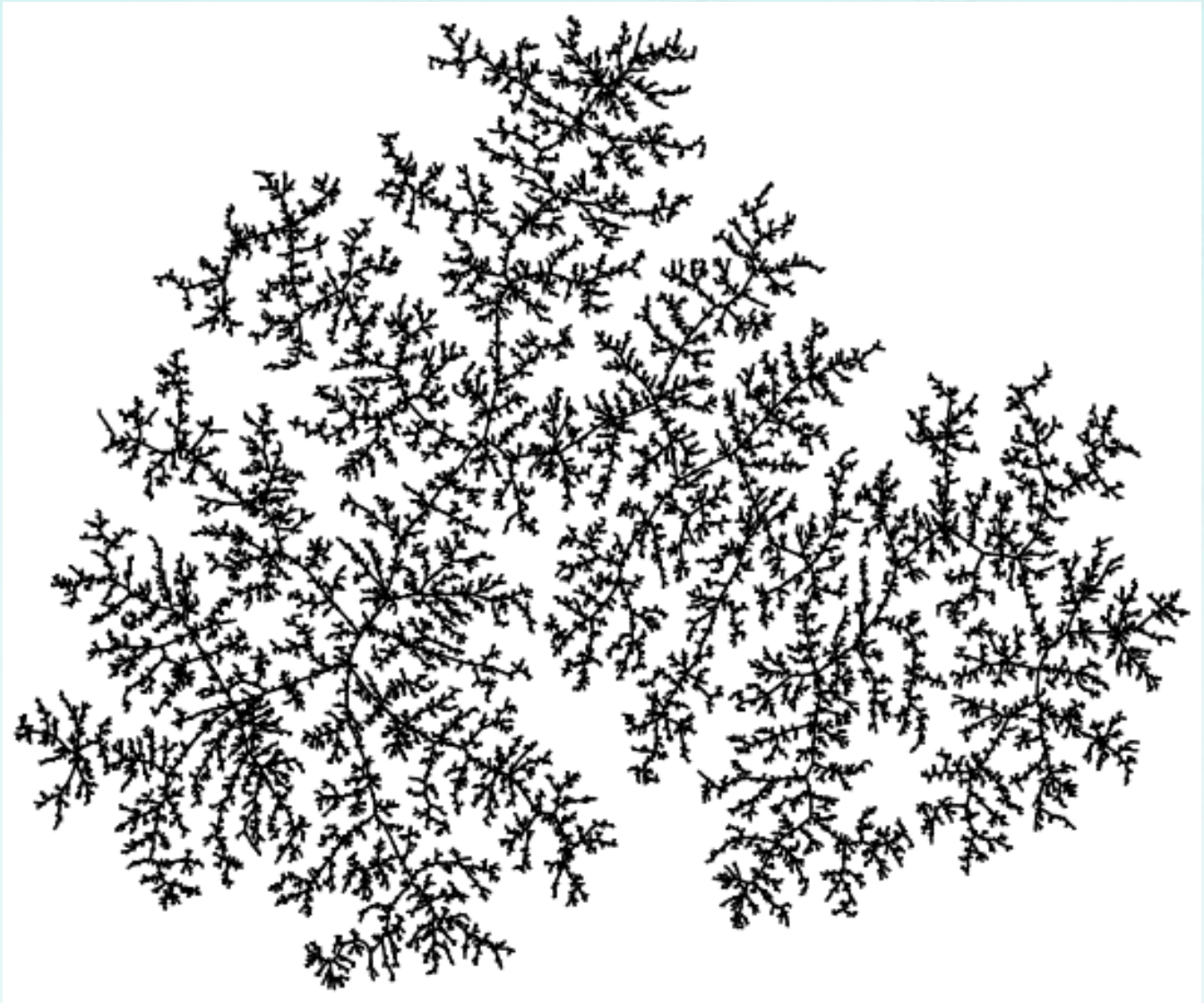
Brute force: Try all spanning trees?

MST of bicycle routes in North Seattle



https://www.flickr.com/photos/ewedistrict/21980840/sizes/z/in/photostream/

Models of nature – MST of random graph



http://algo.inria.fr/broutin/gallery.html/

Applications

MST is fundamental problem with diverse applications.

- Dithering.
- Cluster analysis.
- Max bottleneck paths.
- Real-time face verification.
- LDPC codes for error correction.
- Image registration with Renyi entropy.
- Find road networks in satellite and aerial imagery.
- Reducing data storage in sequencing amino acids in a protein.
- Model locality of particle interactions in turbulent fluid flows.
- Autoconfig protocol for Ethernet bridging to avoid cycles in a network.
- Approximation algorithms for NP-hard problems (e.g., TSP, Steiner tree).
- Network design (communication, electrical, hydraulic, computer, road).

http://www.ics.uci.edu/~eppstein/gina/mst.html

ITP20001/ECE20010 Data Structures

Chapter 6

- *Graph, Digraph*
- *Minimum Spanning Tree (MST)*
  - *Introduction*
  - ***greedy algorithm***
  - *Kruskals's algorithm*
  - *Prim's algorithm*

Major references:

1. Fundamentals of Data Structures by Horowitz, Sahni, Anderson-Freed,
2. Algorithms 4th edition - Part 1 & Part 2 by Robert Sedgewick and Kevin Wayne
3. Wikipedia and many resources available from internet

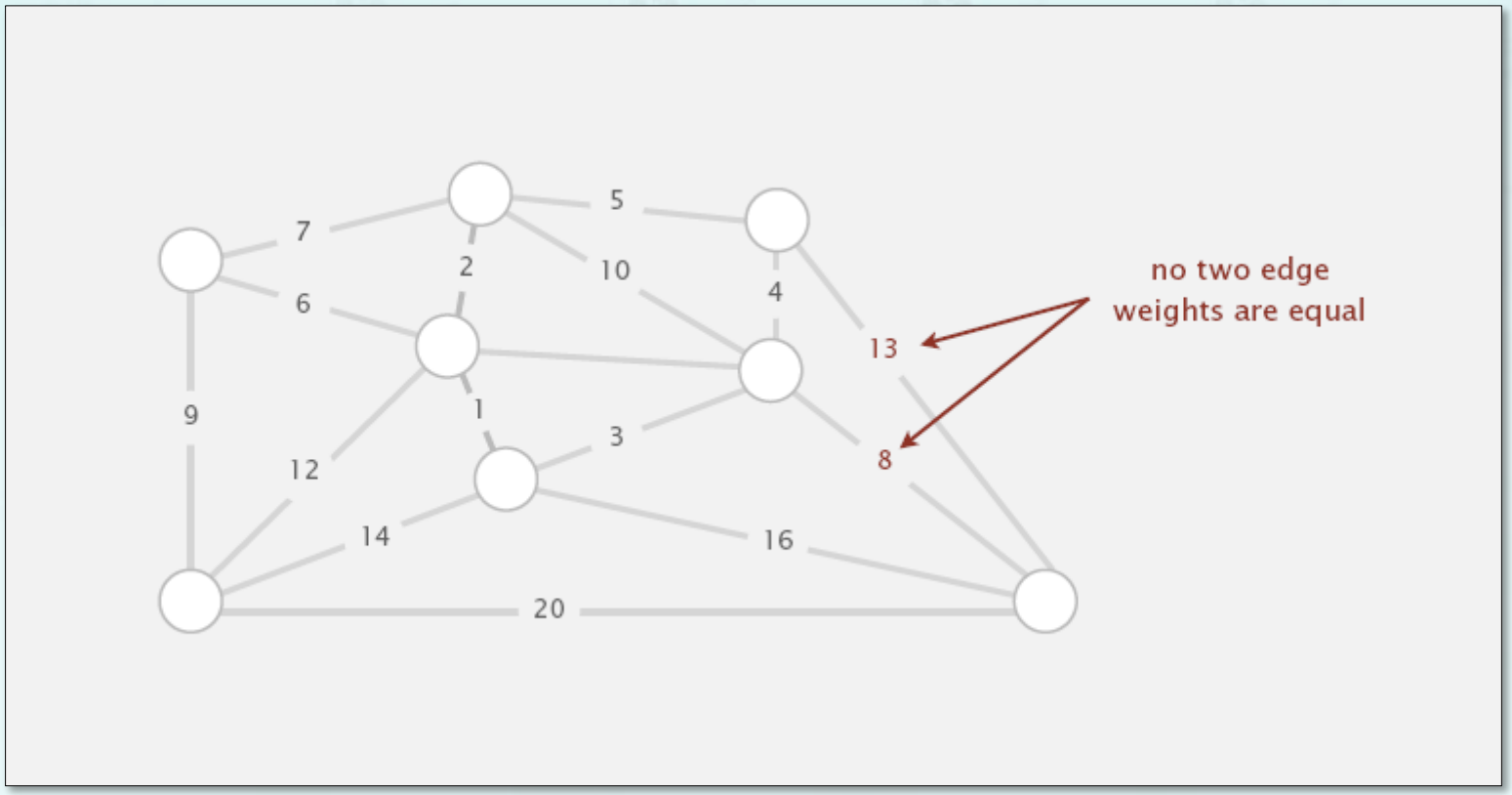
Prof. Youngsup Kim, idebtor@gmail.com, Data Structures, CSEE Dept., Handong Global University

Greedy algorithm – Simplifying assumptions

Simplifying assumptions:

- Edge weights are distinct.
- Graph is connected.

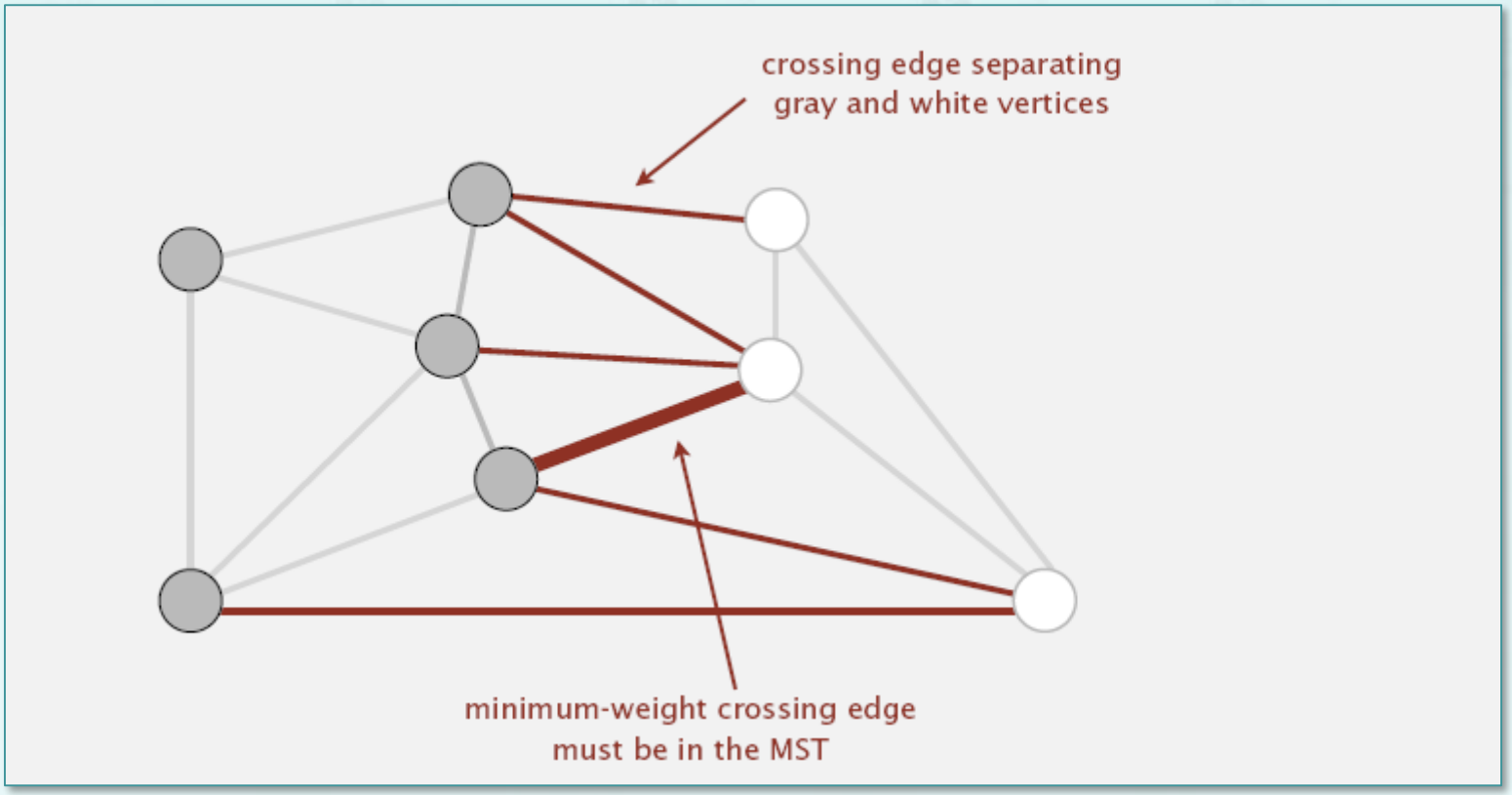
Consequence: MST exists and is unique.



Greedy algorithm – Simplifying assumptions

Def: A cut in a graph is a partition of its vertices into two (nonempty sets).
Def: A crossing edge connects a vertex in one set with a vertex in the other.

Cut property: Given any cut, the crossing edge of min weight is in the MST.



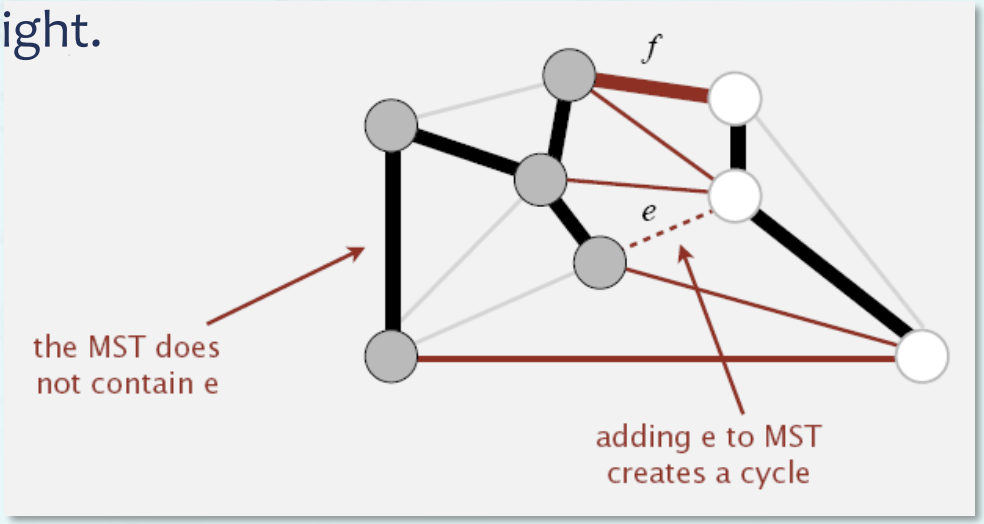
Greedy algorithm – Cut property

Def: A cut in a graph is a partition of its vertices into two (nonempty sets).
Def: A crossing edge connects a vertex in one set with a vertex in the other.

Cut property: Given any cut, the crossing edge of min weight is in the MST.

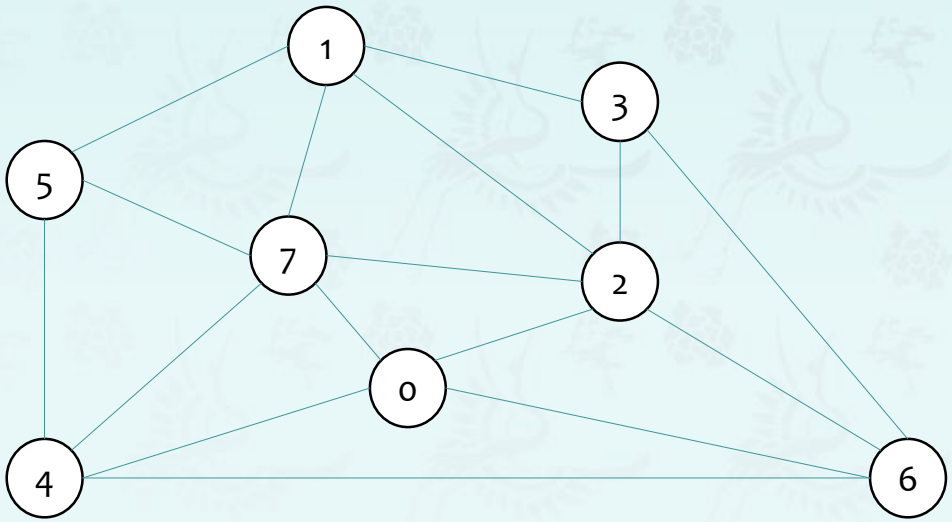
Proof: Suppose min-weight crossing edge e is not in the MST.

- Adding e to the MST create a cycle.
- Some other edge f in cycle must be a crossing edge.
- Removing f and adding e is also a spanning tree.
- Since weight of e is less than the weight of f, that spanning tree is lower weight.
- Contradiction.



Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until V-1 edges are colored black.

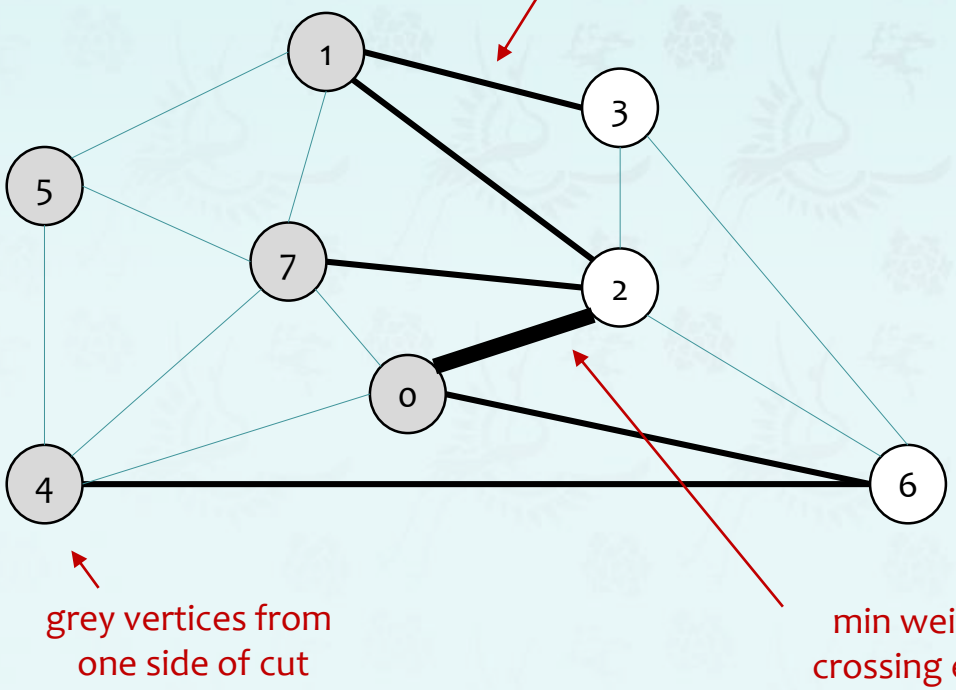


an edge-weighted graph

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.

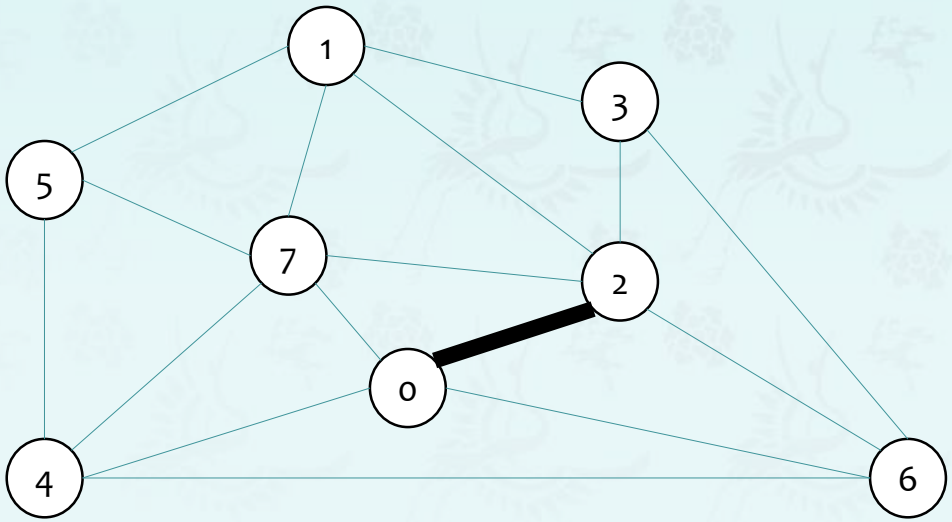


crossing edges (sorted by weight)

in MST		
→	0-2	0.26
	1-3	0.29
	2-7	0.34
	1-2	0.36
	6-0	0.58
	6-4	0.93

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



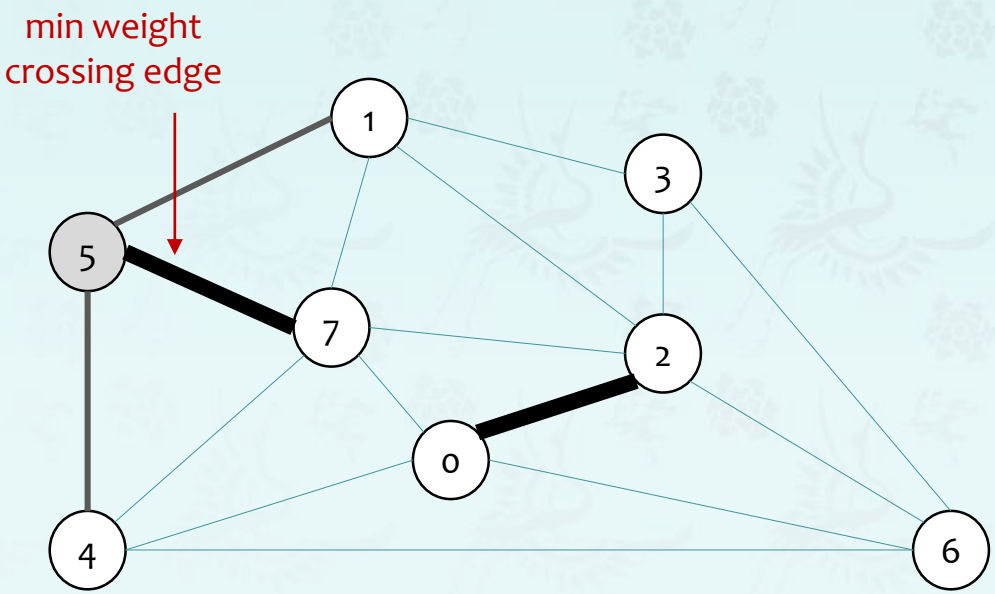
crossing edges (sorted by weight)

in MST →	0-2	0.26
	1-3	0.29
	2-7	0.34
	1-2	0.36
	6-0	0.58
	6-4	0.93

MST edges: 0-2

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



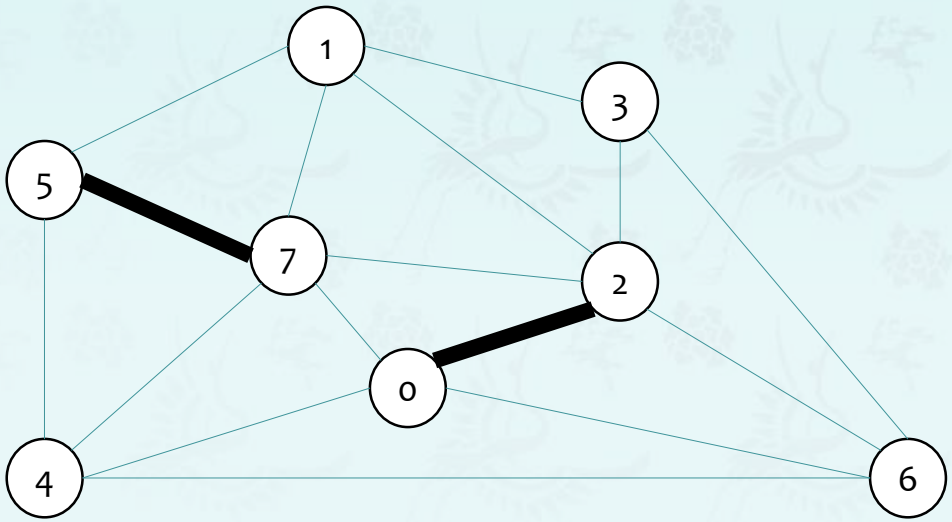
crossing edges
(sorted by weight)

in MST →	5-7	0.28
	1-5	0.32
	4-5	0.35

MST edges: 0-2

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



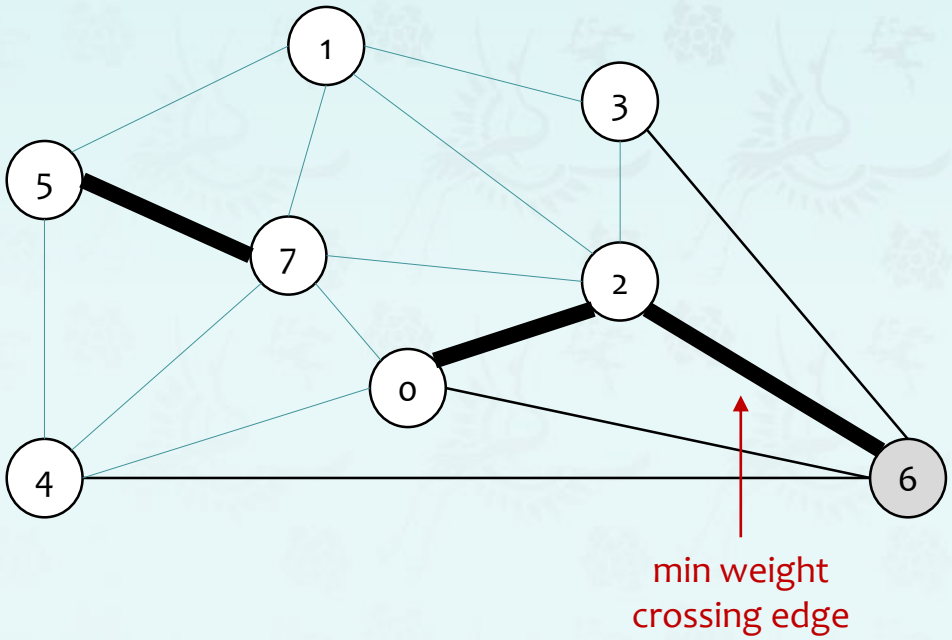
crossing edges (sorted by weight)

in MST		
→	5-7	0.28
	1-5	0.32
	4-5	0.35

MST edges: 0-2 5-7

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



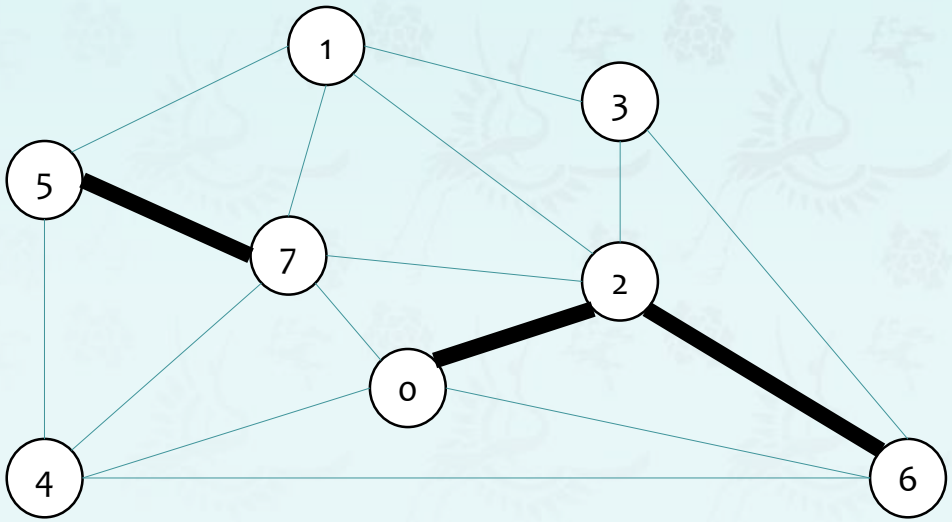
crossing edges
(sorted by weight)

in MST →	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

MST edges: 0-2 5-7

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



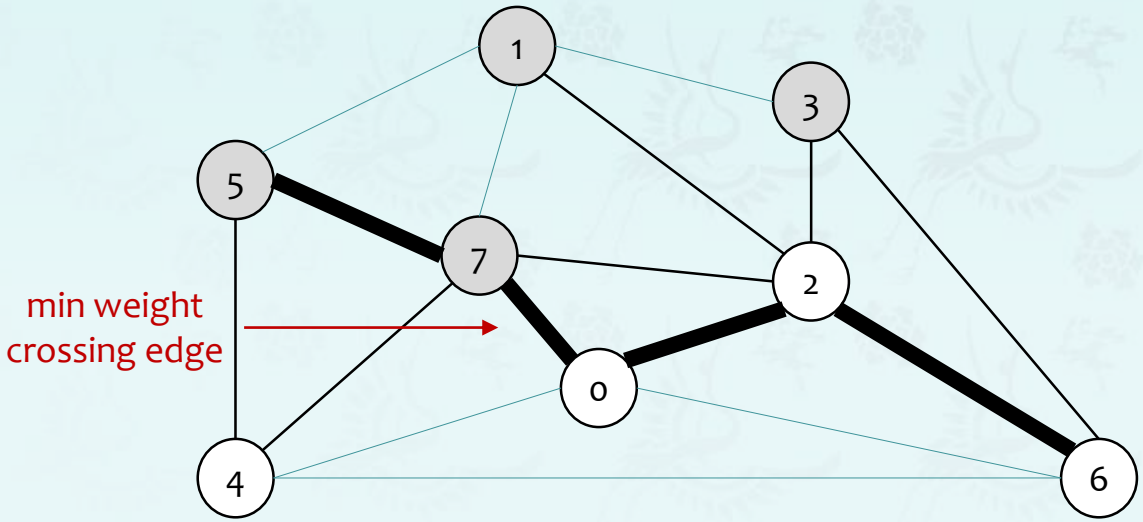
crossing edges (sorted by weight)

in MST		
→	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

MST edges: 0-2 5-7 6-2

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



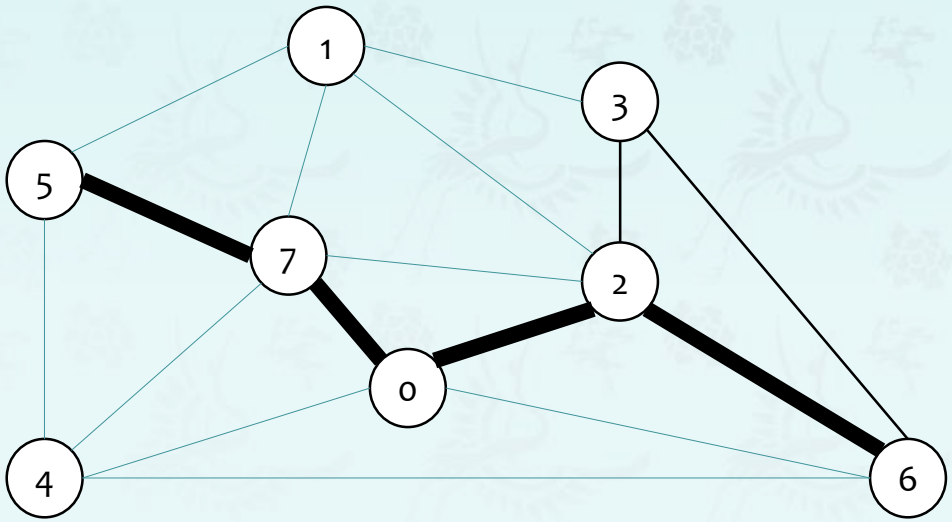
crossing edges
(sorted by weight)

	in MST
0-7	0.16
2-3	0.17
4-5	0.35
1-2	0.36
4-7	0.37
3-6	0.52

MST edges: 0-2 5-7 6-2

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



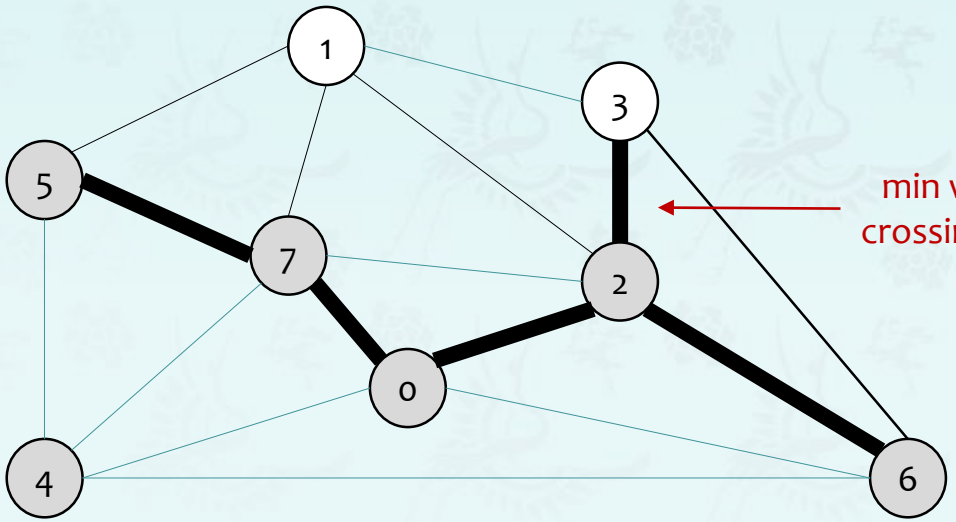
crossing edges (sorted by weight)

in MST		
→	0-7	0.16
	2-3	0.17
	4-5	0.35
	1-2	0.36
	4-7	0.37
	3-6	0.52

MST edges: 0-2 5-7 6-2 0-7

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



min weight crossing edge

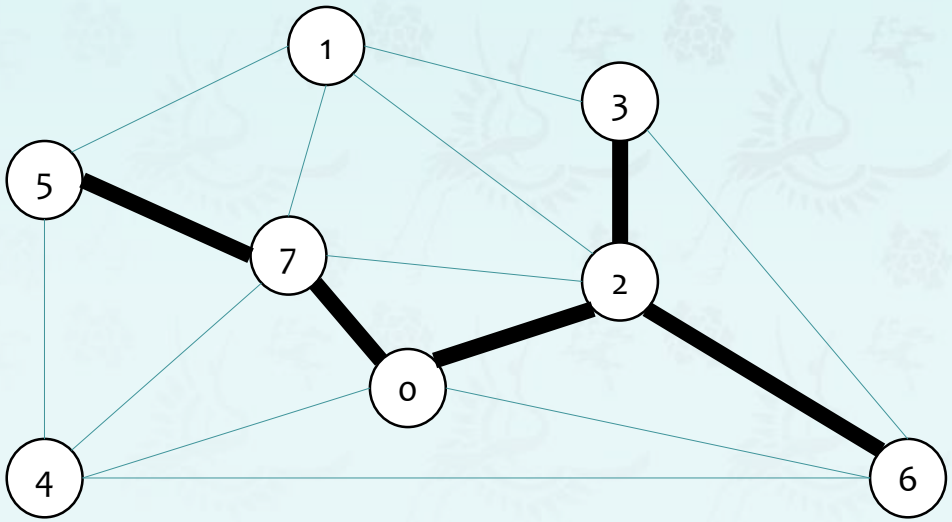
crossing edges (sorted by weight)

	in MST
2-3	0.17
1-7	0.19
1-5	0.32
1-2	0.36
3-6	0.52

MST edges: 0-2 5-7 6-2 0-7

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until V-1 edges are colored black.



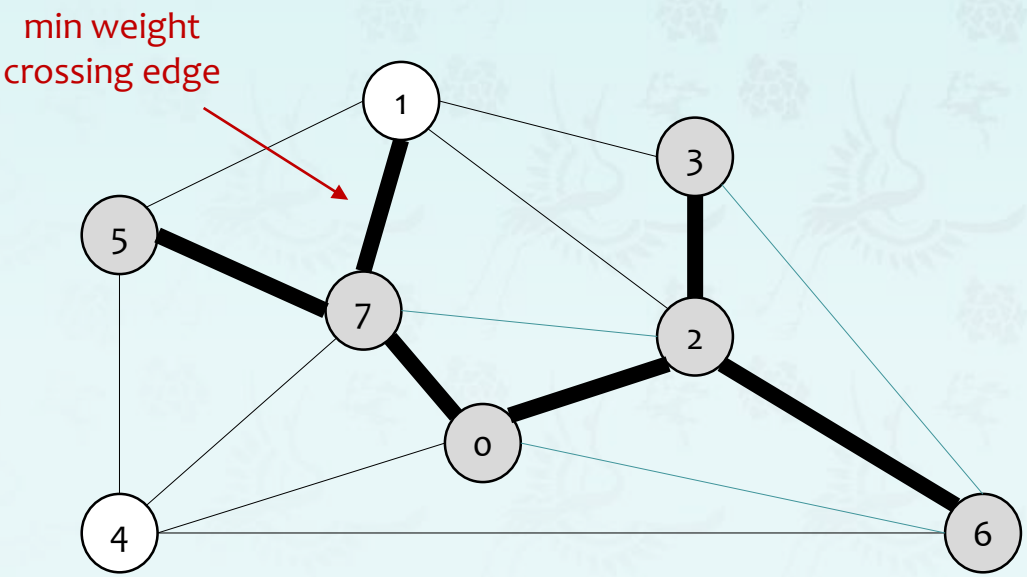
crossing edges (sorted by weight)

in MST		
→	2-3	0.17
	1-7	0.19
	1-5	0.32
	1-2	0.36
	3-6	0.52

MST edges: 0-2 5-7 6-2 0-7 2-3

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



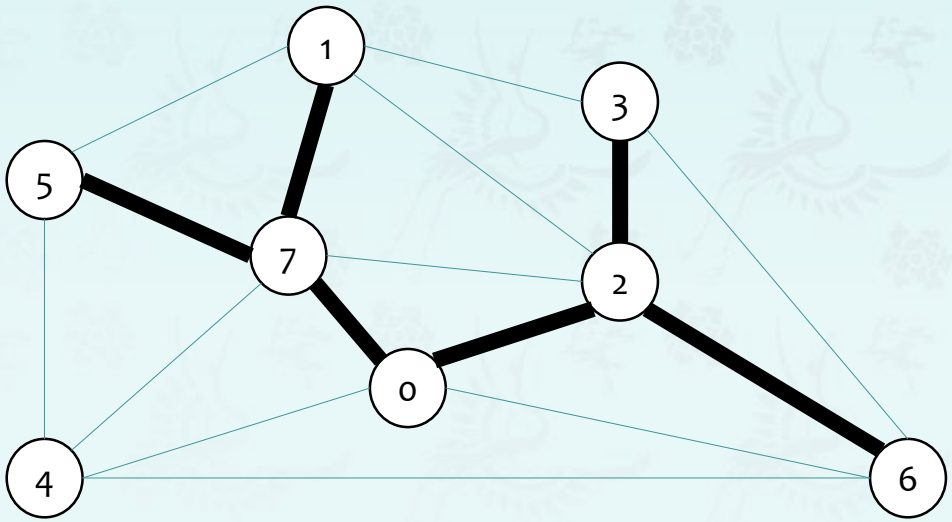
crossing edges
(sorted by weight)

in MST →	1-7	0.19
	1-3	0.29
	1-5	0.32
	1-2	0.36

MST edges: 0-2 5-7 6-2 0-7 2-3

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



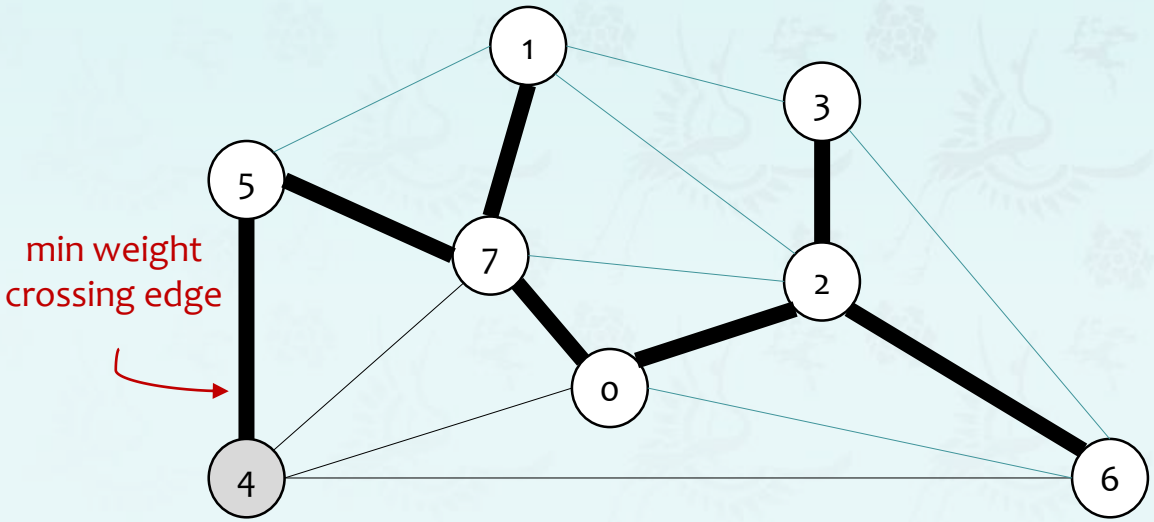
crossing edges (sorted by weight)

in MST		
→	1-7	0.19
	1-3	0.29
	1-5	0.32
	1-2	0.36

MST edges: 0-2 5-7 6-2 0-7 2-3 1-7

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges are colored black.



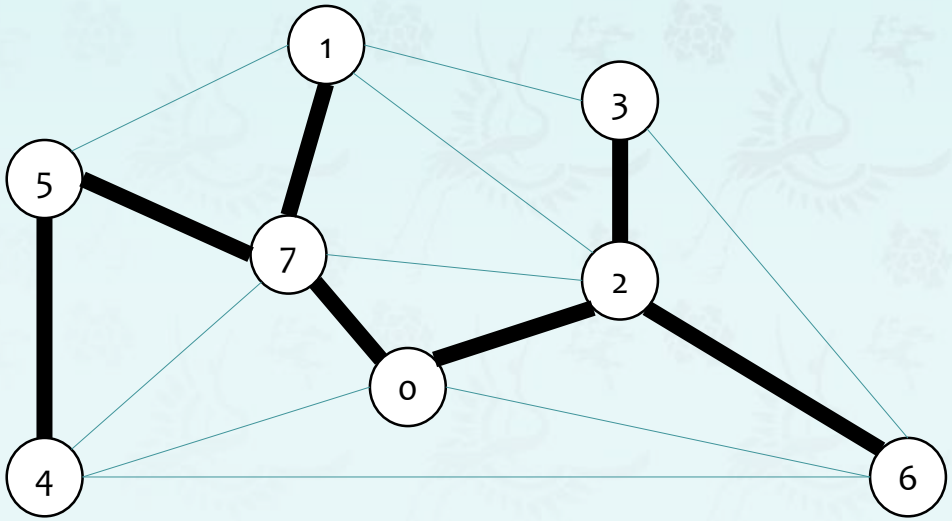
crossing edges
(sorted by weight)

in MST		
→	4-5	0.35
	4-7	0.37
	0-4	0.38
	6-4	0.93

MST edges: 0-2 5-7 6-2 0-7 2-3 1-7

Greedy MST algorithm Demo

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until **V-1** edges **are colored black.**



MST edges: 0-2 5-7 6-2 0-7 2-3 1-7 4-5

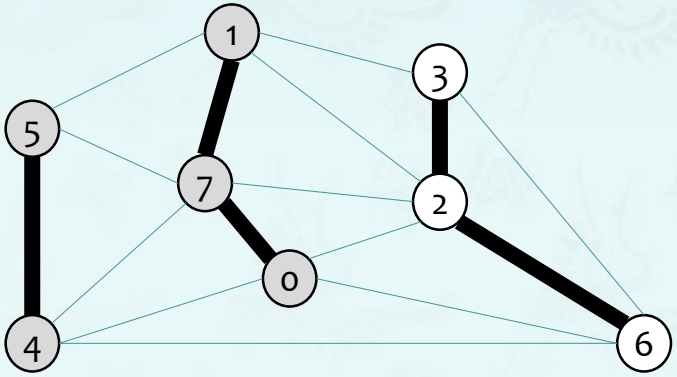
7 (v-1) edges colored black !

Greedy MST algorithm: correctness proof

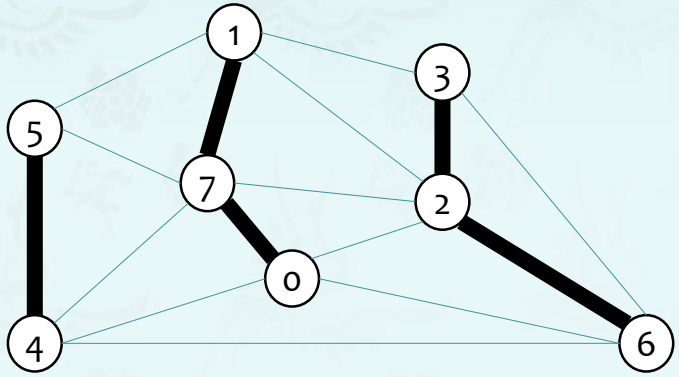
Proposition: The greedy algorithm computes the MST.

Proof:

- Any edge colored black is in the MST (via cut property).
- Fewer than V-1 black edges → cut with no black crossing edges. (consider cut whose vertices are one connected component)



a cut with no black crossing edges



fewer than V-1 edges colored black

Greedy MST algorithm: efficient implementations

Proposition: The greedy algorithm computes the MST.

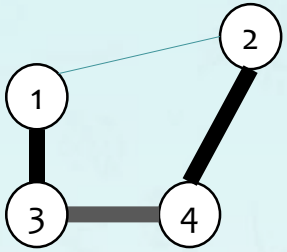
Efficient implementations: Choose cut? Find min-weight edge?

- Ex1: Kruskal's algorithm
- Ex2: Prim's algorithm

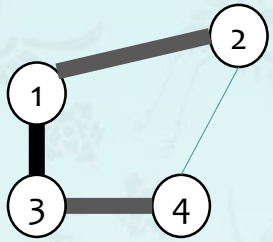
Greedy MST algorithm: Removing two simplifying assumptions

Question: What if edge weights are not all distinct?

Answer: Greedy MST algorithm still correct if equal weights are present!




```
1-2   1.00
1-1   0.50
2-4   1.00
3-4   0.50
```

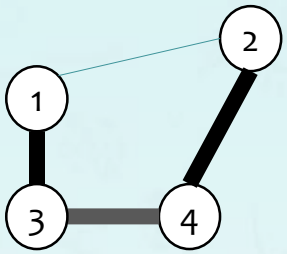



```
1-2   1.00
1-1   0.50
2-4   1.00
3-4   0.50
```

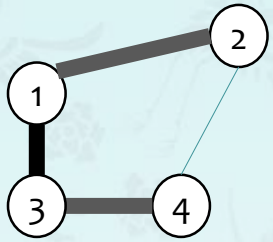
Greedy MST algorithm: Removing two simplifying assumptions

Question: What if edge weights are not all distinct?

Answer: Greedy MST algorithm still correct if equal weights are present!



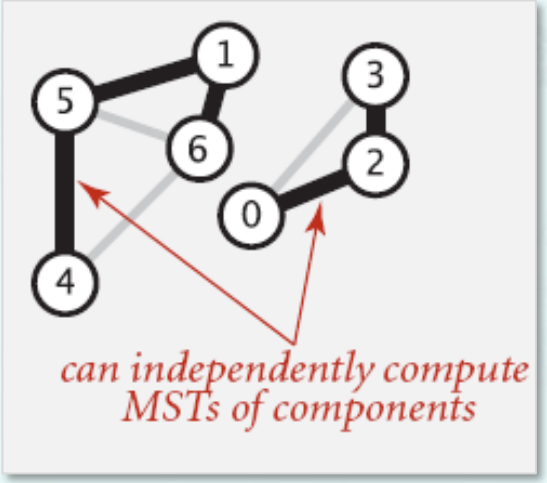

```
1-2   1.00
1-1   0.50
2-4   1.00
3-4   0.50
```




```
1-2   1.00
1-1   0.50
2-4   1.00
3-4   0.50
```

Question: What if graph is not connected ?

Answer: Compute minimum spanning forest = MST of each component.




```
4  5   0.61
4  6   0.62
5  6   0.88
1  5   0.11
2  3   0.35
0  3   0.60
1  6   0.10
0  2   0.22
```

ECE20010 Data Structures

Chapter 6

- *Graph, Digraph*
- *Minimum Spanning Tree (MST)*
  - *Introduction*
  - *greedy algorithm*
  - ***Kruskals's algorithm***
  - *Prim's algorithm*

Major references:

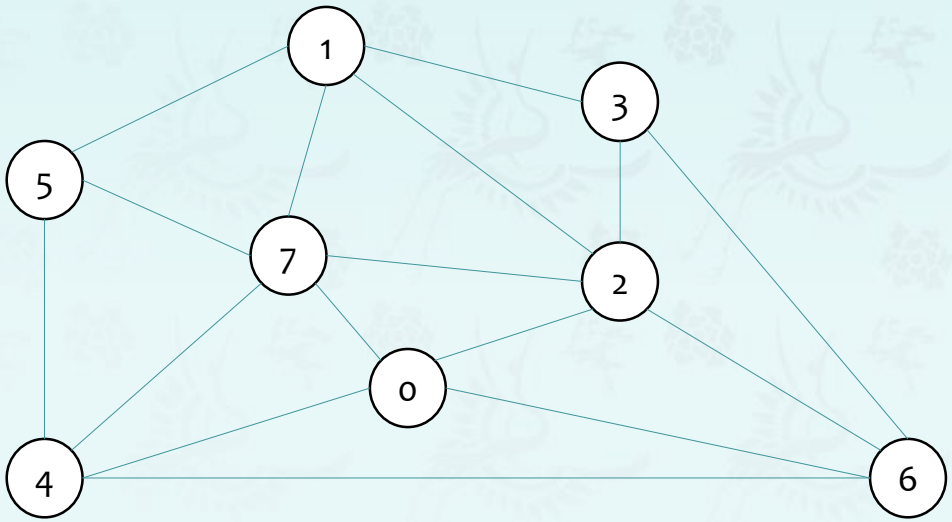
1. Fundamentals of Data Structures by Horowitz, Sahni, Anderson-Freed,
2. Algorithms 4th edition - Part 1 & Part 2 by Robert Sedgewick and Kevin Wayne
3. Wikipedia and many resources available from internet

Prof. Youngsup Kim, idebtor@handong.edu, 2014 Data Structures, CSEE Dept., Handong Global University

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



an edge-weighted graph

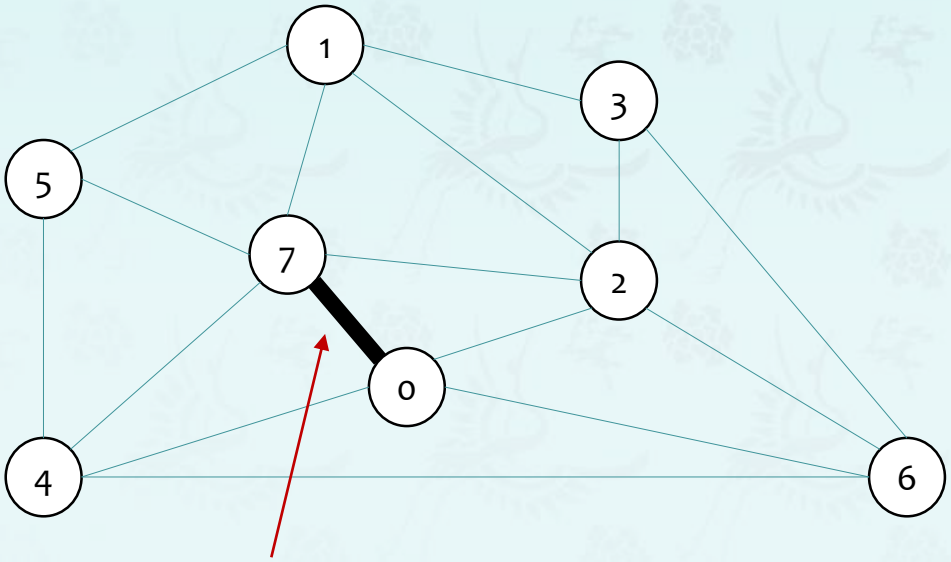
graph edges sorted by weight

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



graph edges sorted by weight

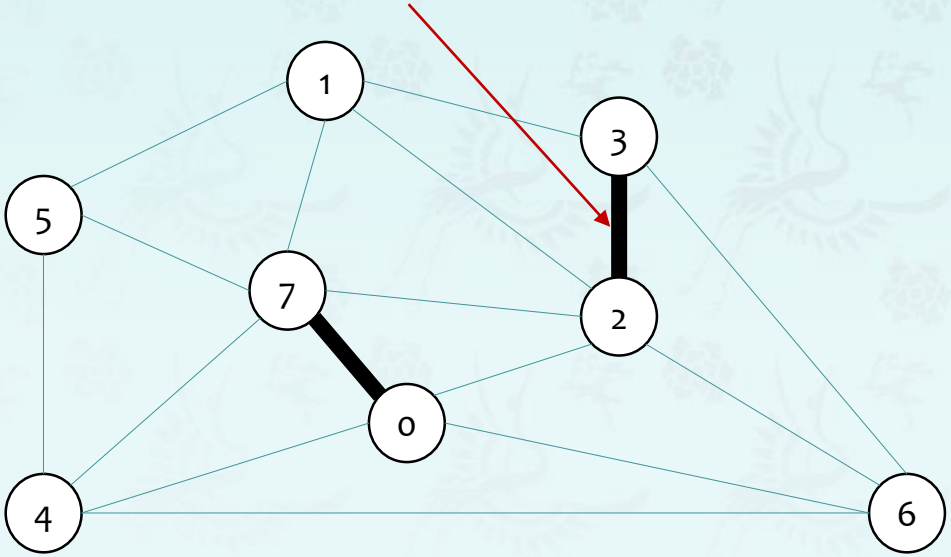
add it in MST →

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



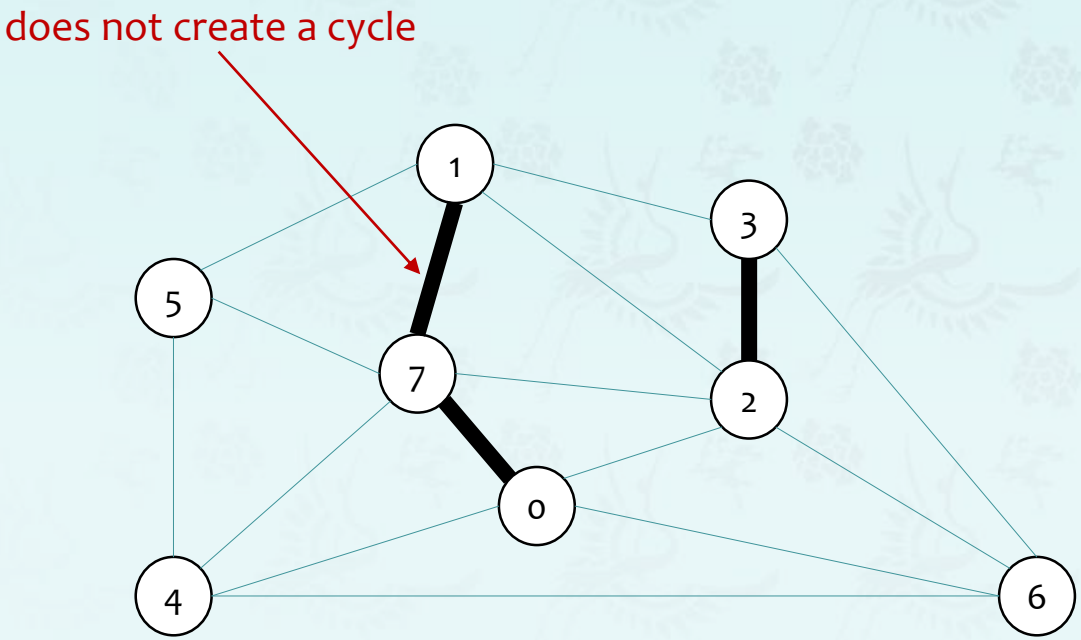
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
	1-7	0.19
	0-2	0.26
	5-7	0.28
	1-3	0.29
	1-5	0.32
	2-7	0.34
	4-5	0.35
	1-2	0.36
	4-7	0.37
	0-4	0.38
	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



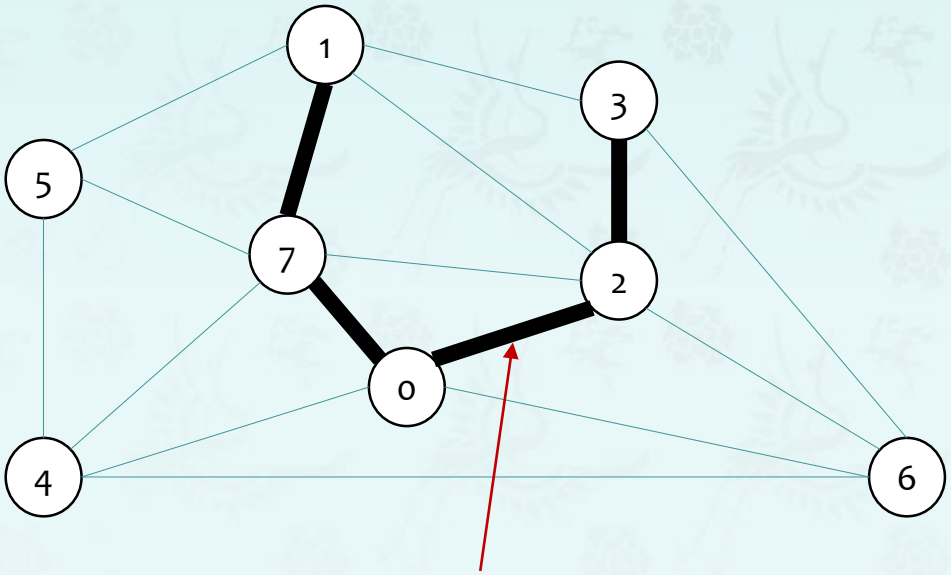
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
	0-2	0.26
	5-7	0.28
	1-3	0.29
	1-5	0.32
	2-7	0.34
	4-5	0.35
	1-2	0.36
	4-7	0.37
	0-4	0.38
	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



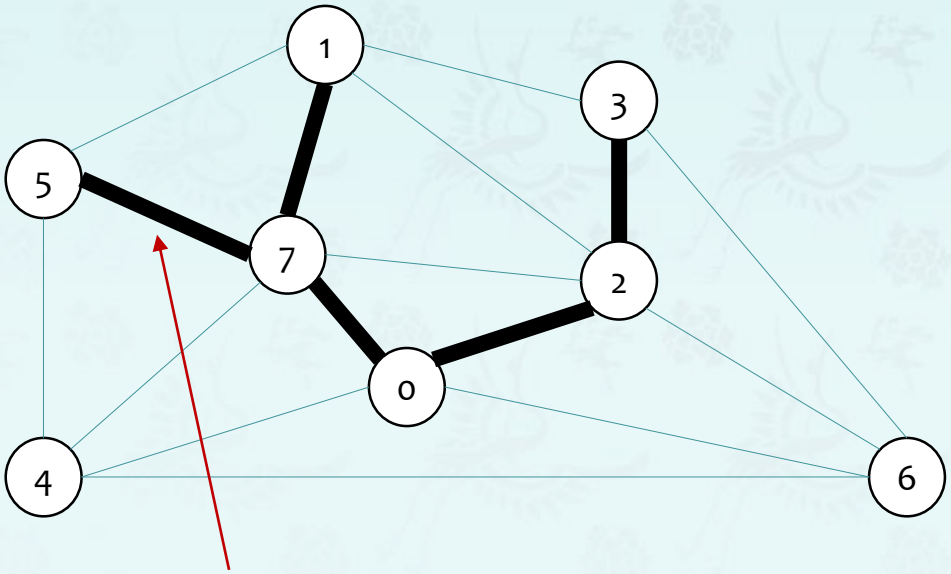
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
	5-7	0.28
	1-3	0.29
	1-5	0.32
	2-7	0.34
	4-5	0.35
	1-2	0.36
	4-7	0.37
	0-4	0.38
	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



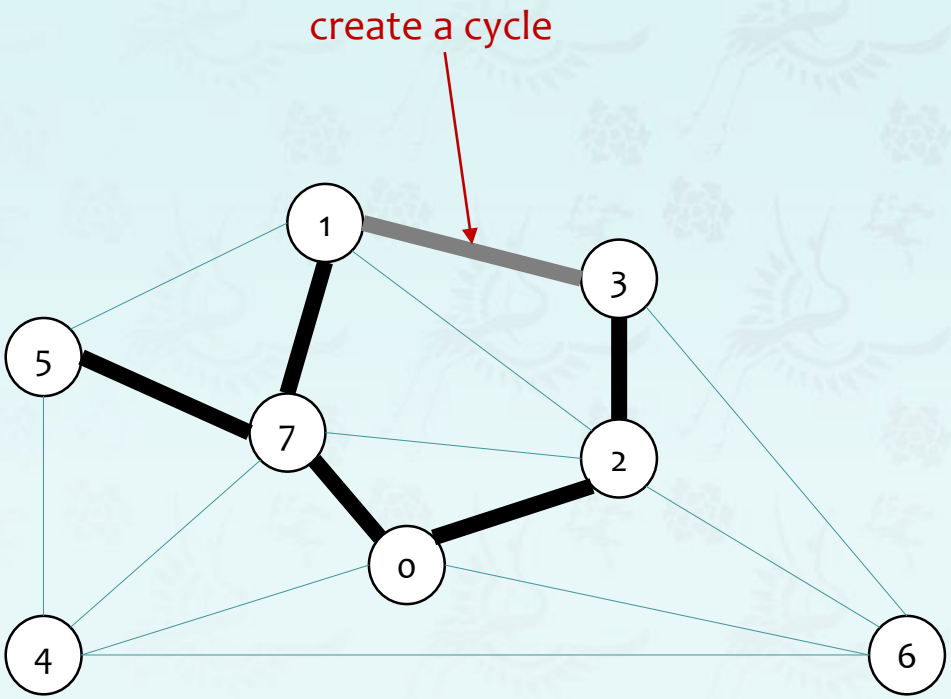
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
	1-3	0.29
	1-5	0.32
	2-7	0.34
	4-5	0.35
	1-2	0.36
	4-7	0.37
	0-4	0.38
	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



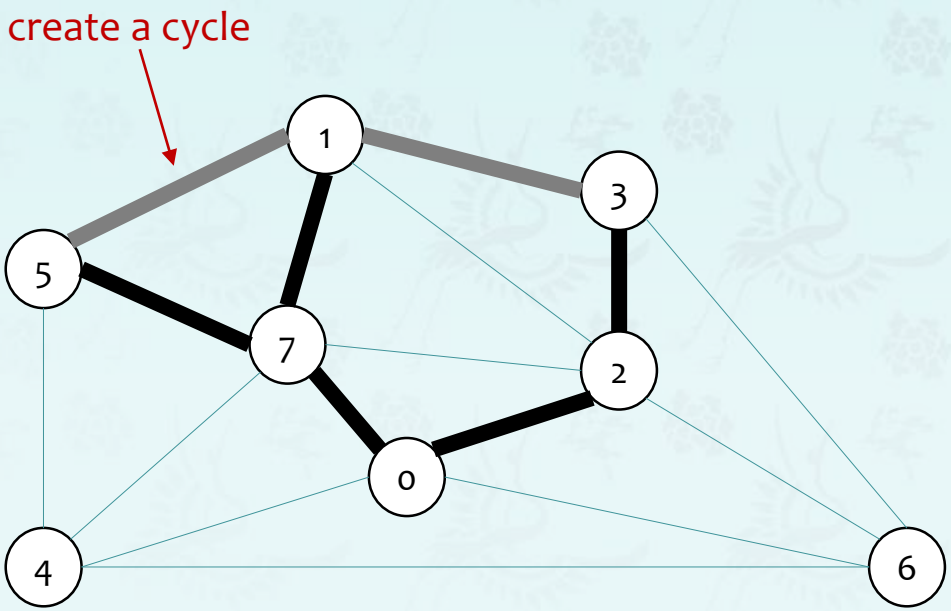
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
NOT in MST	1-3	0.29
	1-5	0.32
	2-7	0.34
	4-5	0.35
	1-2	0.36
	4-7	0.37
	0-4	0.38
	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



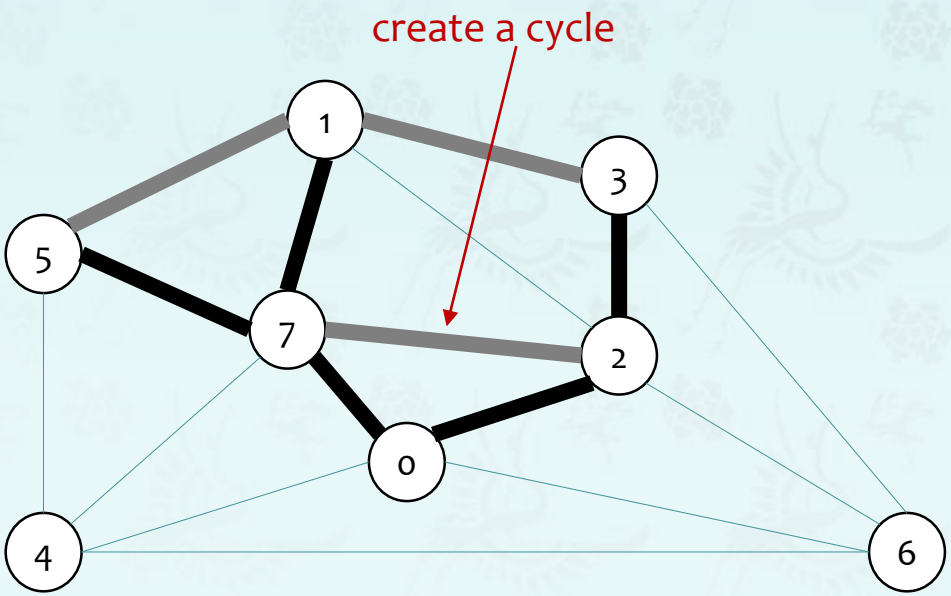
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
NOT in MST	1-3	0.29
NOT in MST	1-5	0.32
	2-7	0.34
	4-5	0.35
	1-2	0.36
	4-7	0.37
	0-4	0.38
	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



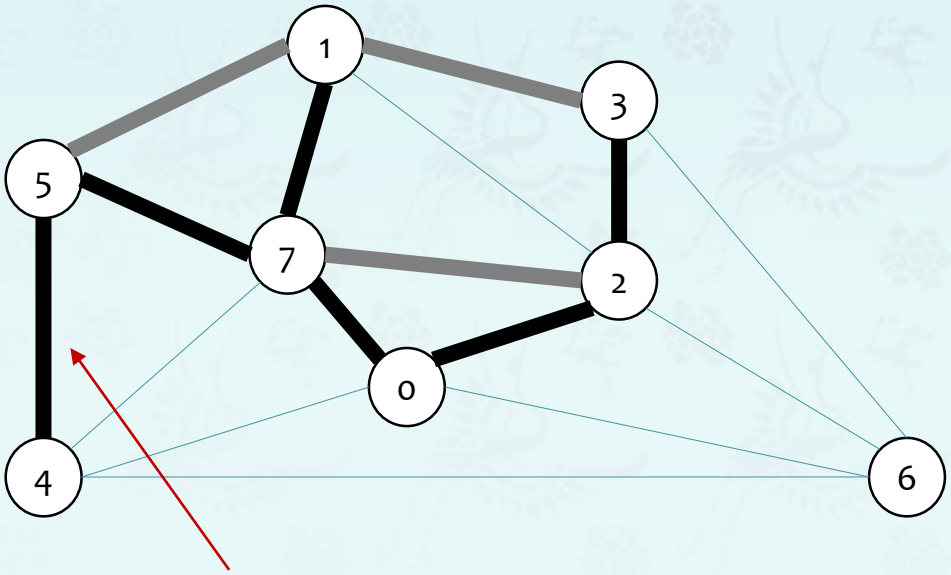
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
NOT in MST	1-3	0.29
NOT in MST	1-5	0.32
NOT in MST	2-7	0.34
	4-5	0.35
	1-2	0.36
	4-7	0.37
	0-4	0.38
	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



does not create a cycle

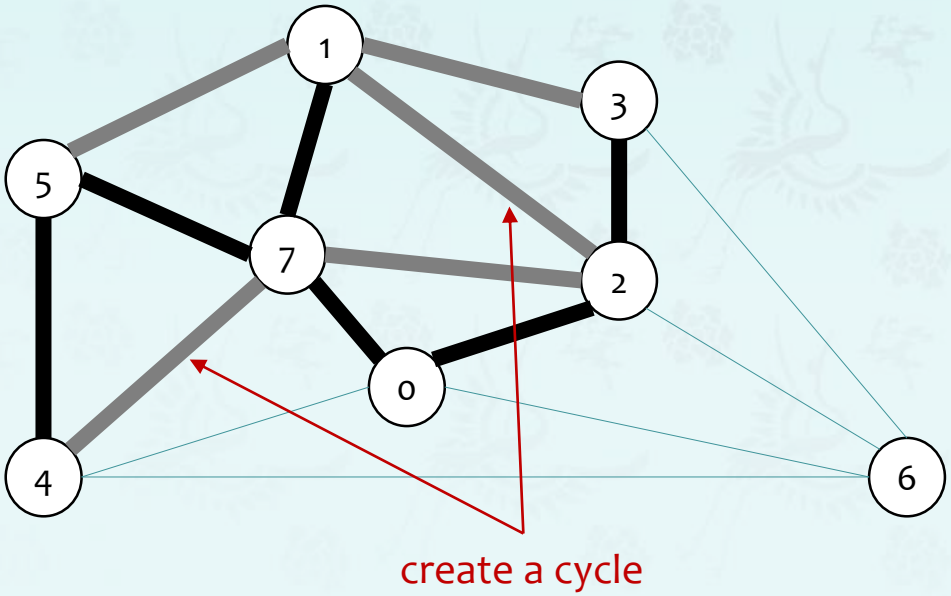
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
NOT in MST	1-3	0.29
NOT in MST	1-5	0.32
NOT in MST	2-7	0.34
add it in MST →	4-5	0.35
	1-2	0.36
	4-7	0.37
	0-4	0.38
	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



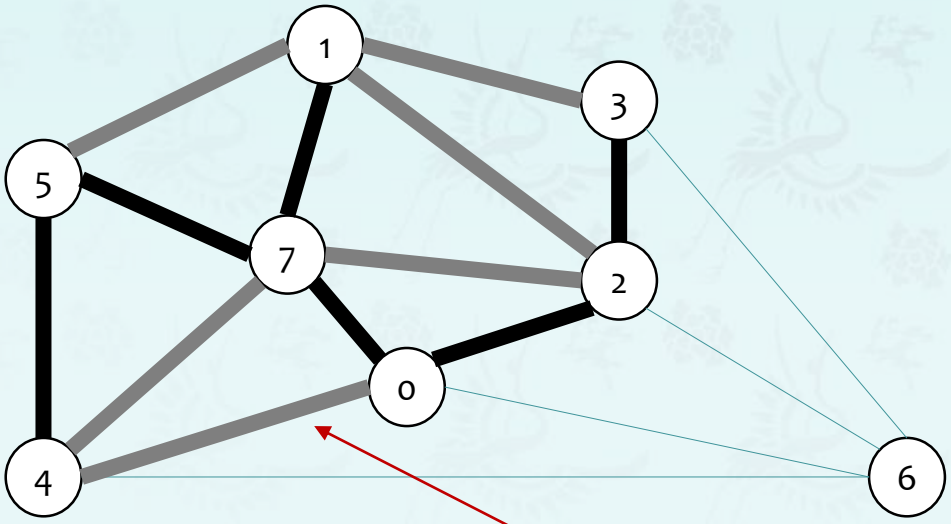
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
NOT in MST	1-3	0.29
NOT in MST	1-5	0.32
NOT in MST	2-7	0.34
add it in MST →	4-5	0.35
NOT in MST	1-2	0.36
NOT in MST	4-7	0.37
	0-4	0.38
	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



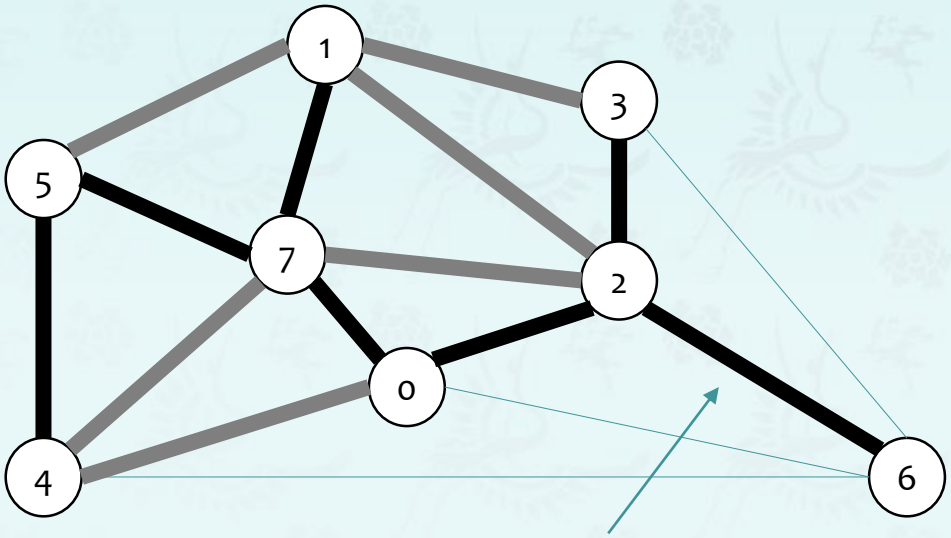
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
NOT in MST	1-3	0.29
NOT in MST	1-5	0.32
NOT in MST	2-7	0.34
add it in MST →	4-5	0.35
NOT in MST	1-2	0.36
NOT in MST	4-7	0.37
NOT in MST	0-4	0.38
	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



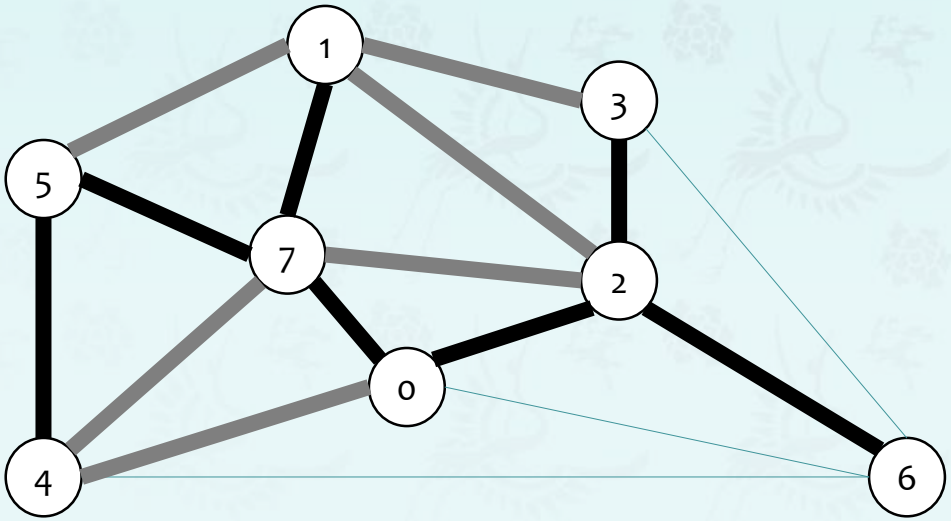
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
NOT in MST	1-3	0.29
NOT in MST	1-5	0.32
NOT in MST	2-7	0.34
add it in MST →	4-5	0.35
NOT in MST	1-2	0.36
NOT in MST	4-7	0.37
NOT in MST	0-4	0.38
add it in MST →	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



create a cycle

graph edges sorted by weight

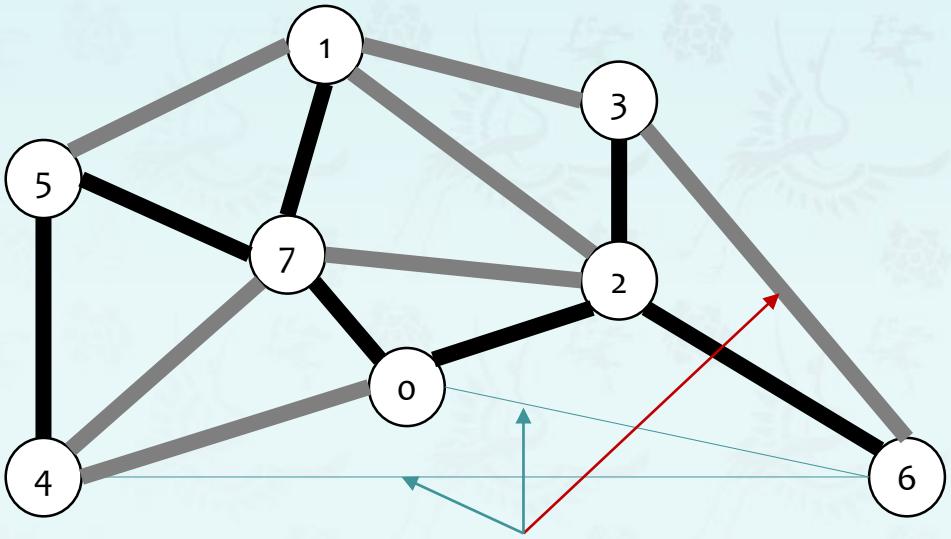
add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
NOT in MST	1-3	0.29
NOT in MST	1-5	0.32
NOT in MST	2-7	0.34
add it in MST →	4-5	0.35
NOT in MST	1-2	0.36
NOT in MST	4-7	0.37
NOT in MST	0-4	0.38
add it in MST →	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

- **7 (v-1) edges – It is a MST. You may continue on.**

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



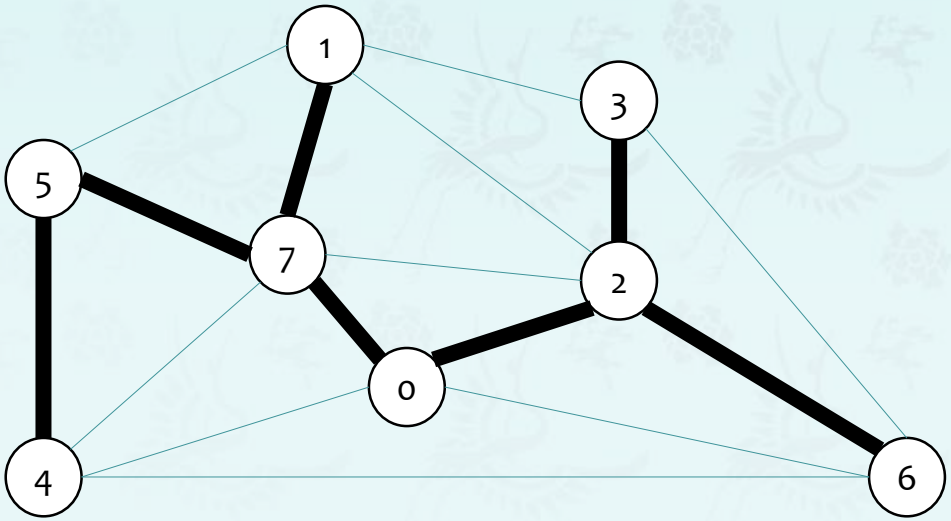
graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
NOT in MST	1-3	0.29
NOT in MST	1-5	0.32
NOT in MST	2-7	0.34
add it in MST →	4-5	0.35
NOT in MST	1-2	0.36
NOT in MST	4-7	0.37
NOT in MST	0-4	0.38
add it in MST →	6-2	0.40
NOT in MST	3-6	0.52
NOT in MST	6-0	0.58
NOT in MST	6-4	0.93

Kruskal's algorithm Demo

Consider edges in ascending order of weight.

- **Add next edge to tree T unless doing so would create a cycle.**



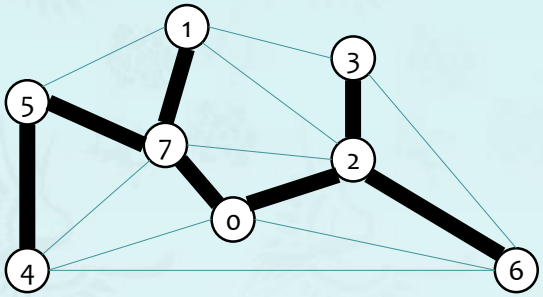
a minimum spanning tree

graph edges sorted by weight

add it in MST →	0-7	0.16
add it in MST →	2-3	0.17
add it in MST →	1-7	0.19
add it in MST →	0-2	0.26
add it in MST →	5-7	0.28
NOT in MST		
NOT in MST		
NOT in MST		
add it in MST →	4-5	0.35
NOT in MST		
NOT in MST		
NOT in MST		
add it in MST →	6-2	0.40
NOT in MST		
NOT in MST		
NOT in MST		

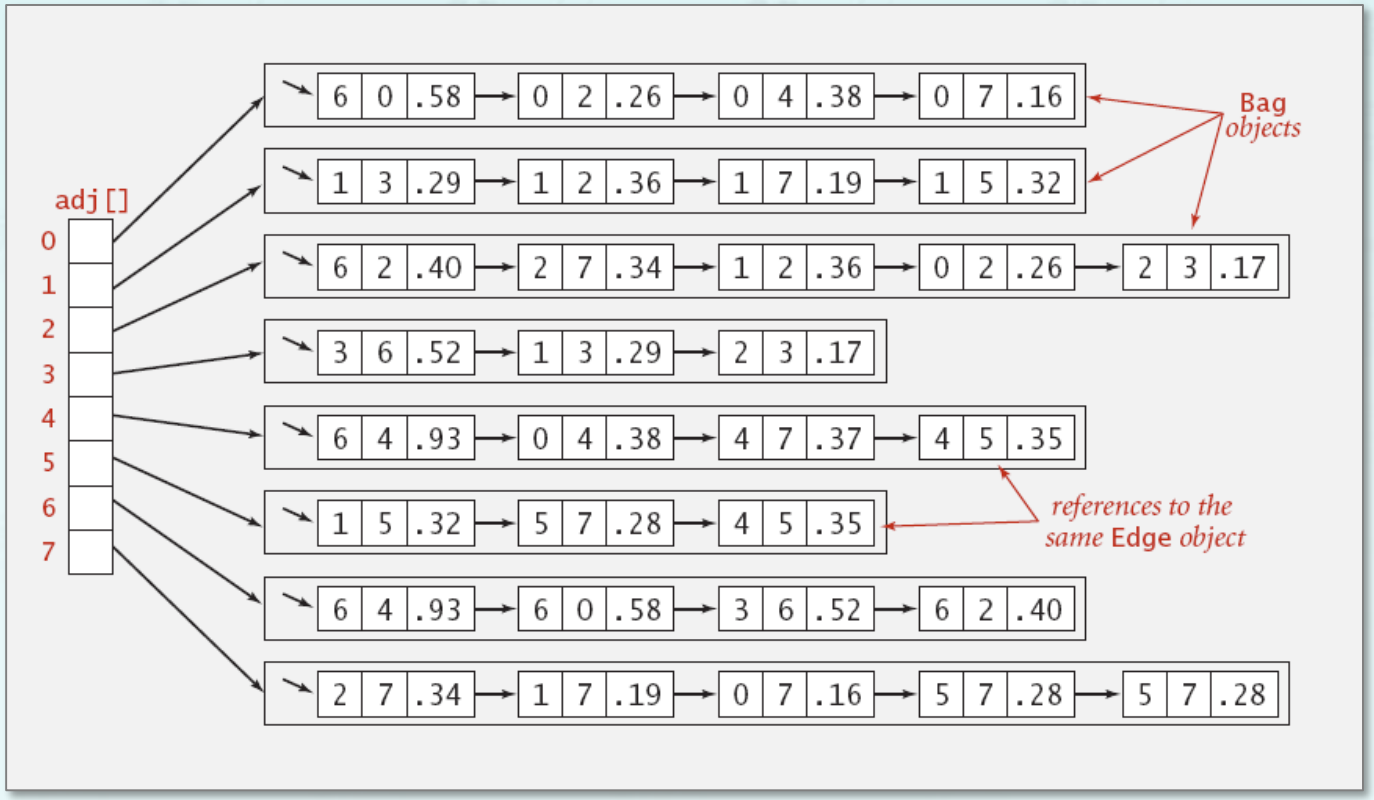
Edge-weighted graph: adjacency-lists representation

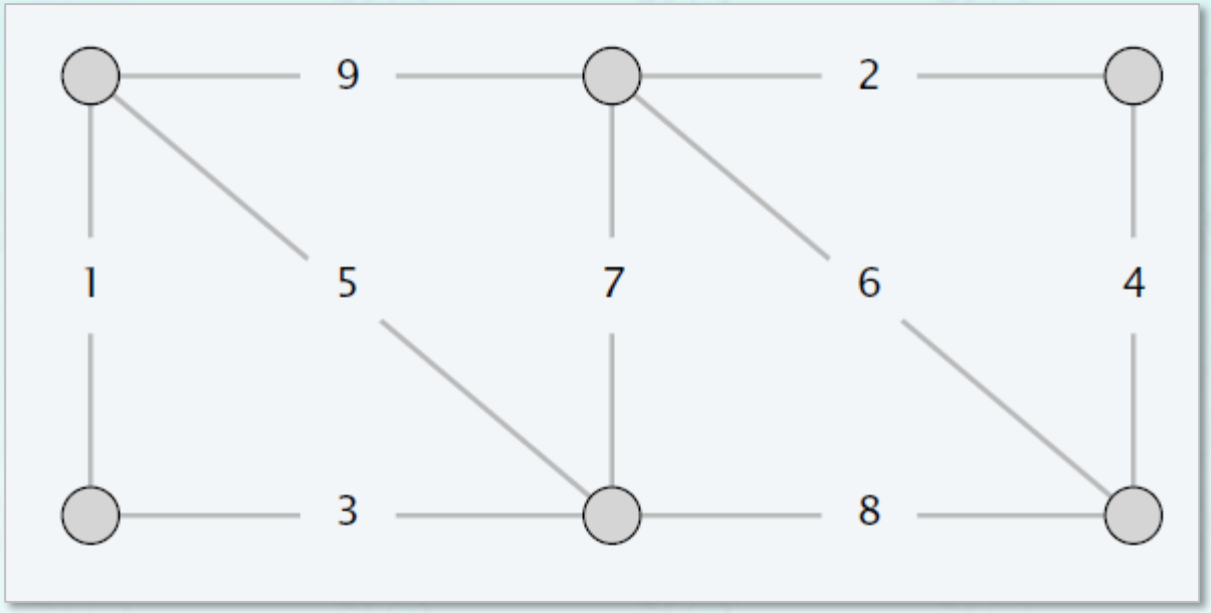
Maintain vertex-indexed array of Edge lists.



GraphEW.txt

```
8          <- V
16         <- E
4 5  0.35
4 7  0.37
5 7  0.28
0 7  0.16
1 5  0.32
0 4  0.38
2 3  0.17
1 7  0.19
0 2  0.26
1 2  0.36
1 3  0.29
2 7  0.34
6 2  0.40
3 6  0.52
6 0  0.58
6 4  0.93
```




9
2
1
5
7
6
4
3
8

ITP20001/ECE20010 Data Structures

Chapter 6

- *Minimum Spanning Tree (MST)*
  - *Introduction*
  - *greedy algorithm*
  - *Kruskals's algorithm*
  - ***Prim's algorithm***

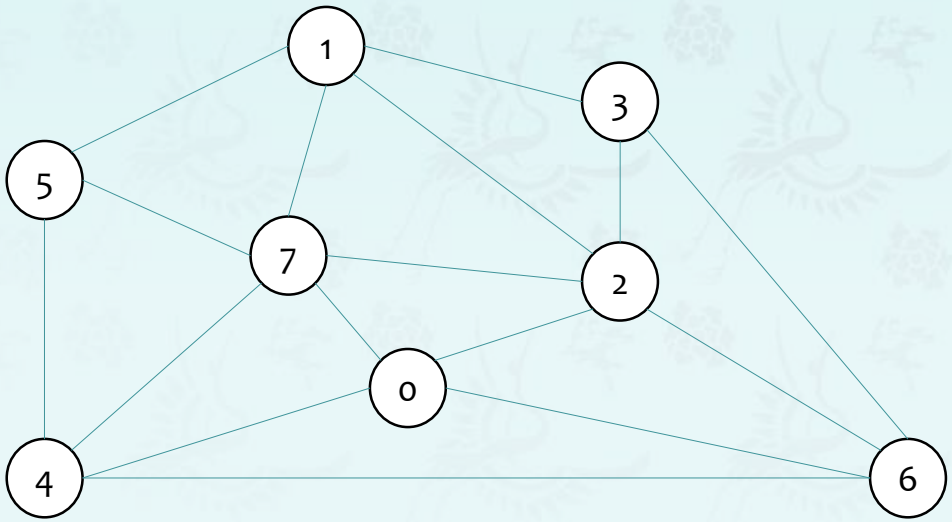
Major references:

1. Fundamentals of Data Structures by Horowitz, Sahni, Anderson-Freed,
2. Algorithms 4th edition - Part 1 & Part 2 by Robert Sedgewick and Kevin Wayne
3. Wikipedia and many resources available from internet

Prof. Youngsup Kim, idebtor@gmail.com, Data Structures, CSEE Dept., Handong Global University

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



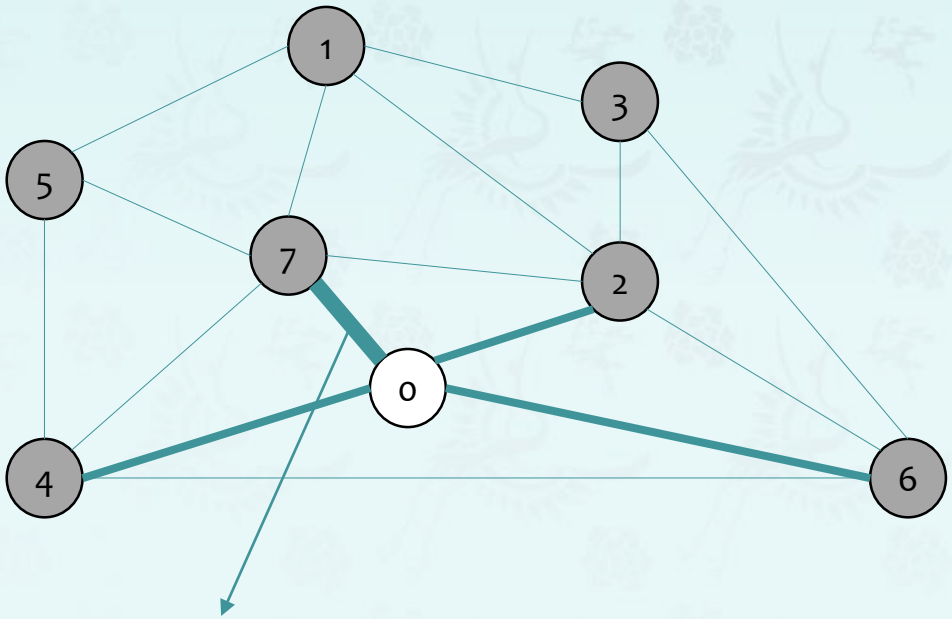
an edge-weighted graph

graph edges sorted by weight

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



edges with exactly one endpoint in T (sorted by weight)

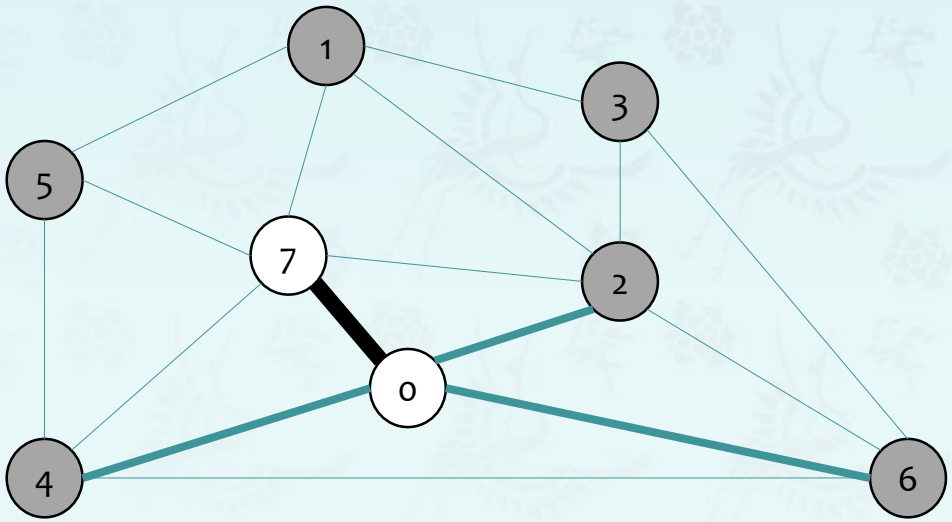
in MST →

0-7	0.16
0-2	0.26
0-4	0.38
6-0	0.58

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



edges with exactly one endpoint in T (sorted by weight)

in MST →

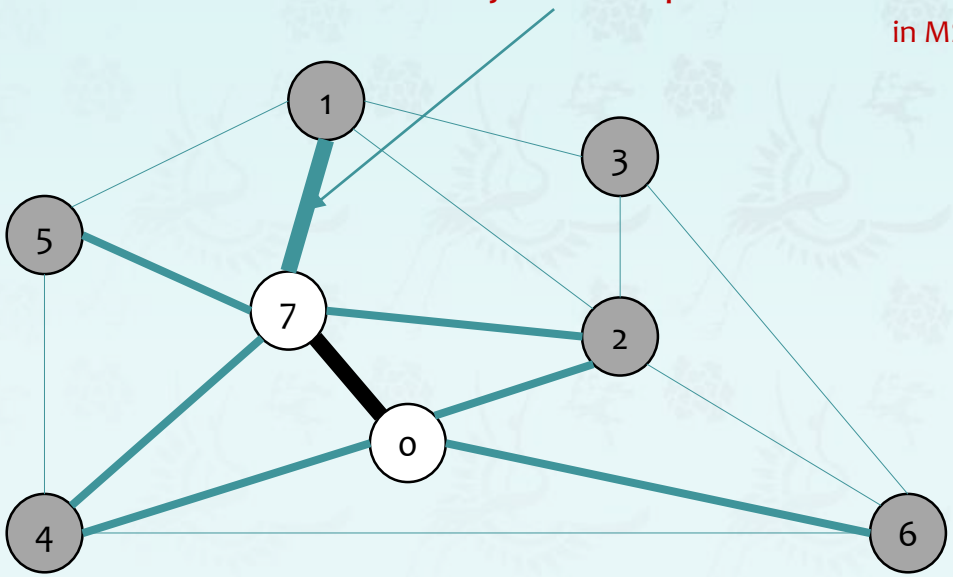
0-7	0.16
0-2	0.26
0-4	0.38
6-0	0.58

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

MST edges: 0-7

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



edges with exactly one endpoint in T (sorted by weight)

in MST →

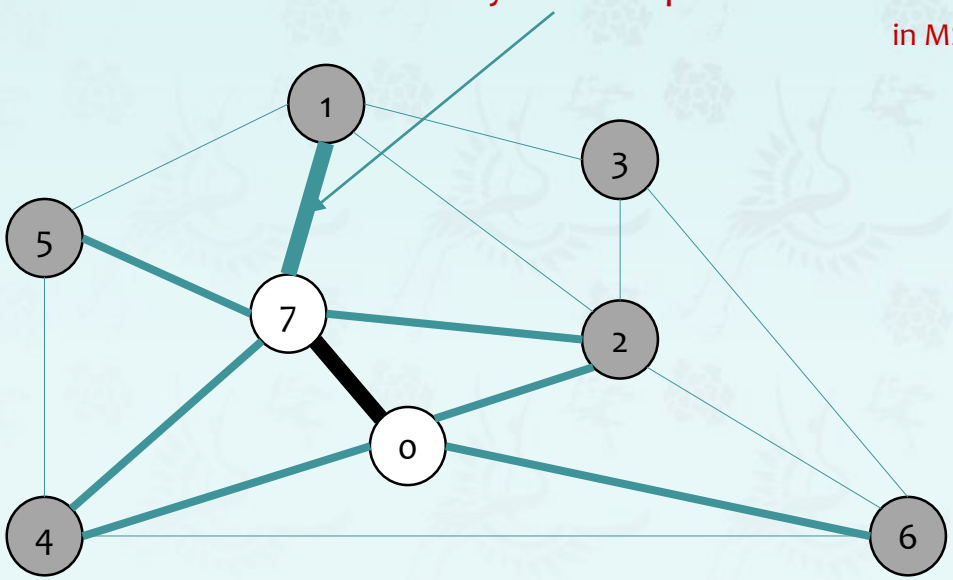
1-7	0.19
0-2	0.26
5-7	0.28
2-7	0.34
4-7	0.37
0-4	0.38
6-0	0.58

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

MST edges: 0-7

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



edges with exactly one endpoint in T (sorted by weight)

in MST →

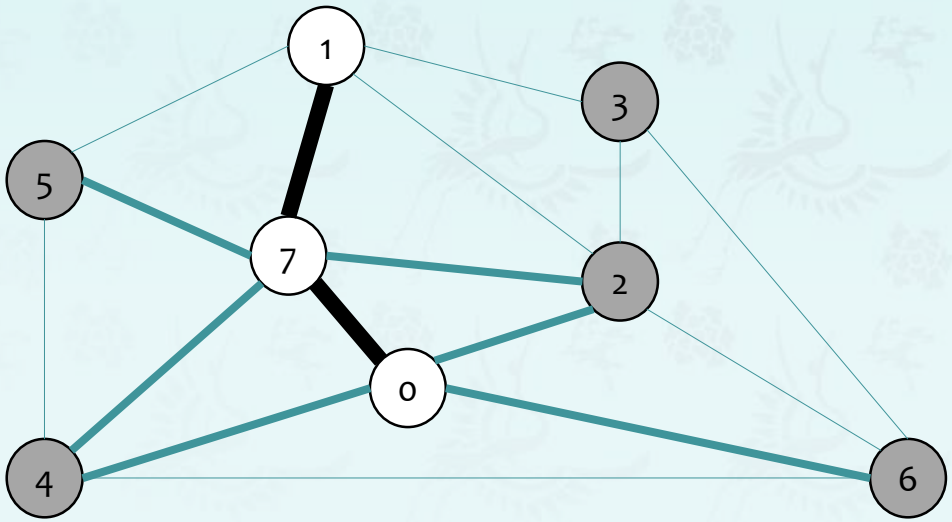
1-7	0.19
0-2	0.26
5-7	0.28
2-7	0.34
4-7	0.37
0-4	0.38
6-0	0.58

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

MST edges: 0-7

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



edges with exactly one endpoint in T (sorted by weight)

in MST →

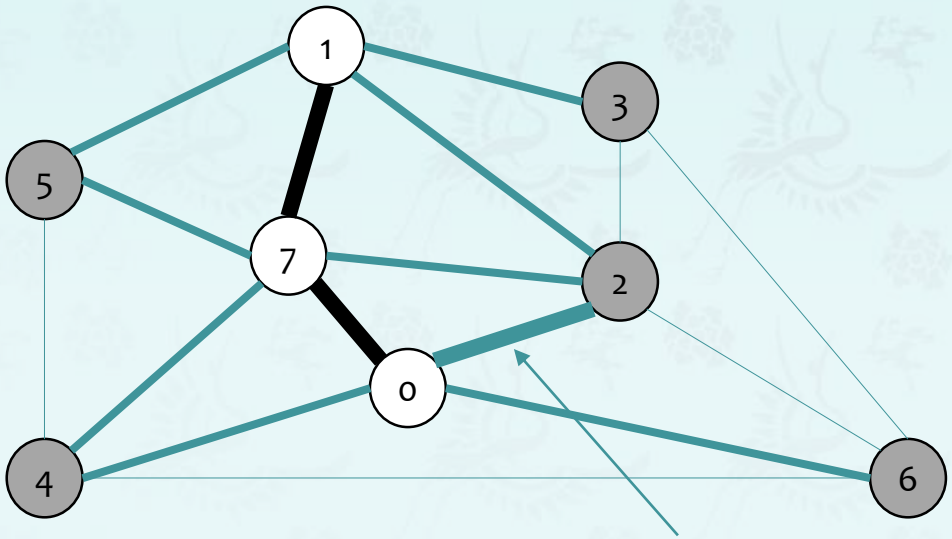
1-7	0.19
0-2	0.26
5-7	0.28
2-7	0.34
4-7	0.37
0-4	0.38
6-0	0.58

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

MST edges: 0-7 1-7

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



min weight edge with exactly one endpoint in T

edges with exactly one endpoint in T (sorted by weight)

in MST →

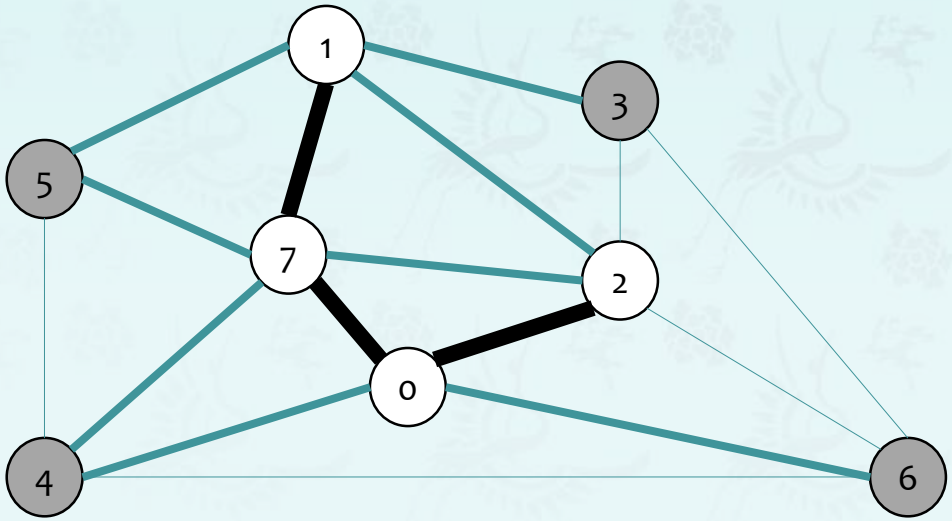
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-0	0.58

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

MST edges: 0-7 1-7

Prim's algorithm Demo

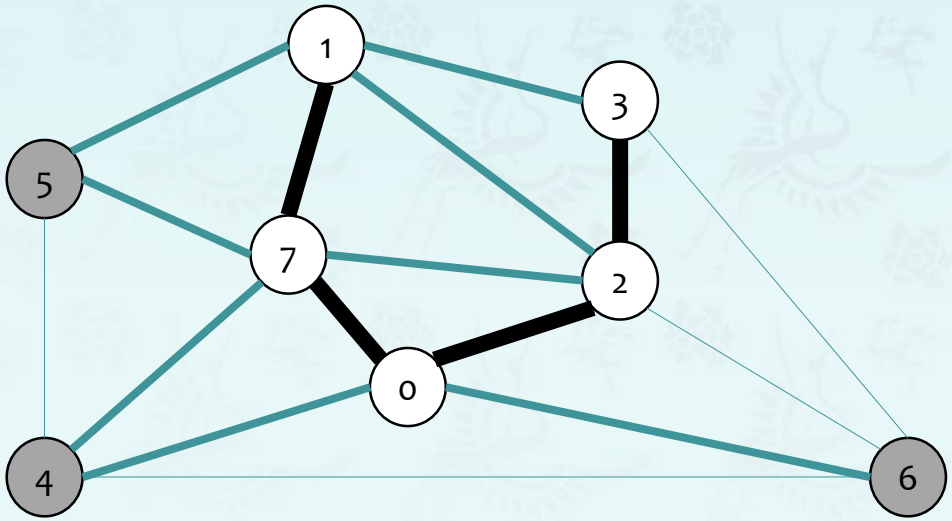
- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



MST edges: 0-7 1-7 0-2

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.

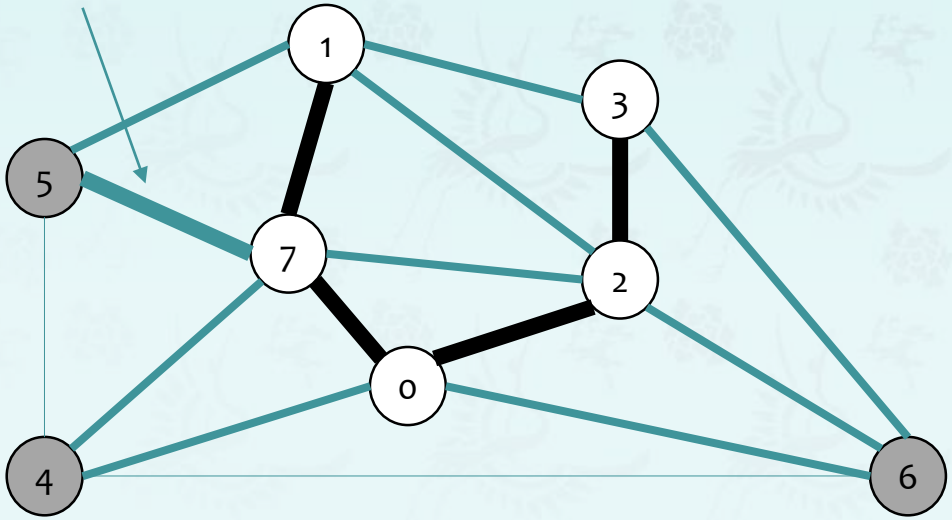


MST edges: 0-7 1-7 0-2 2-3

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.

min weight edge with exactly one endpoint in T



edges with exactly one endpoint in T (sorted by weight)

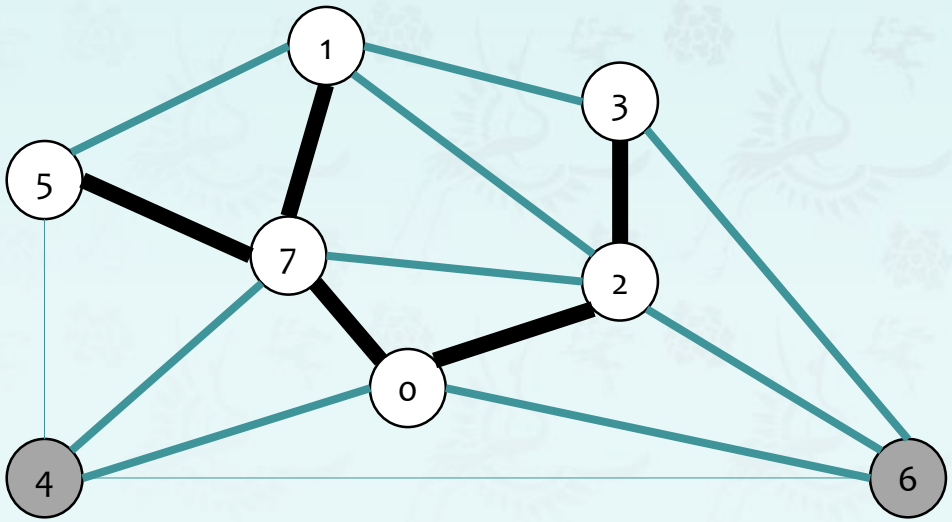
in MST →

5-7	0.28
1-5	0.32
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

MST edges: 0-7 1-7 0-2 2-3

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



edges with exactly one endpoint in T (sorted by weight)

in MST →

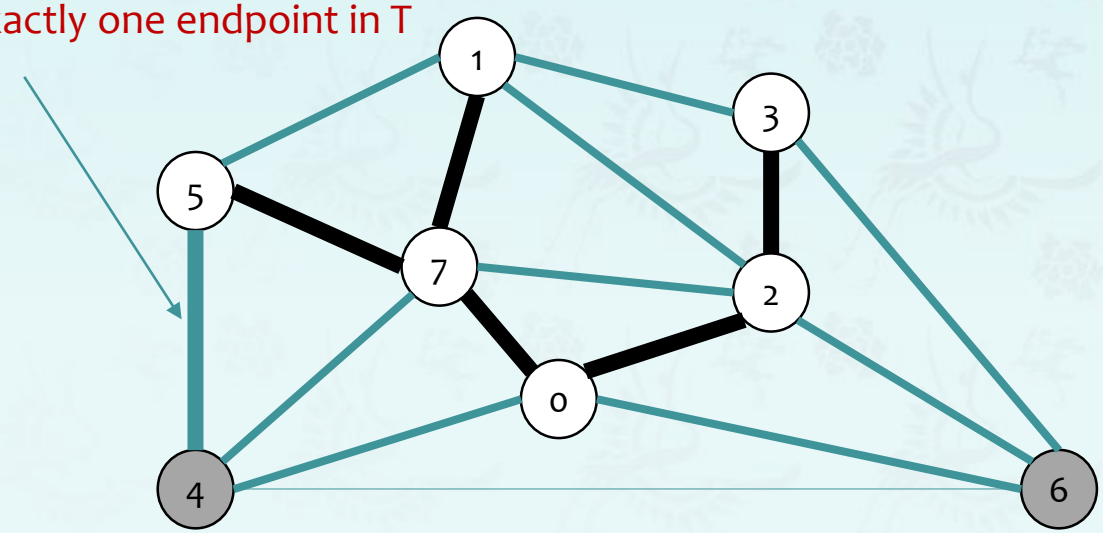
5-7	0.28
1-5	0.32
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

MST edges: 0-7 1-7 0-2 2-3 5-7

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.

min weight edge with exactly one endpoint in T



edges with exactly one endpoint in T (sorted by weight)

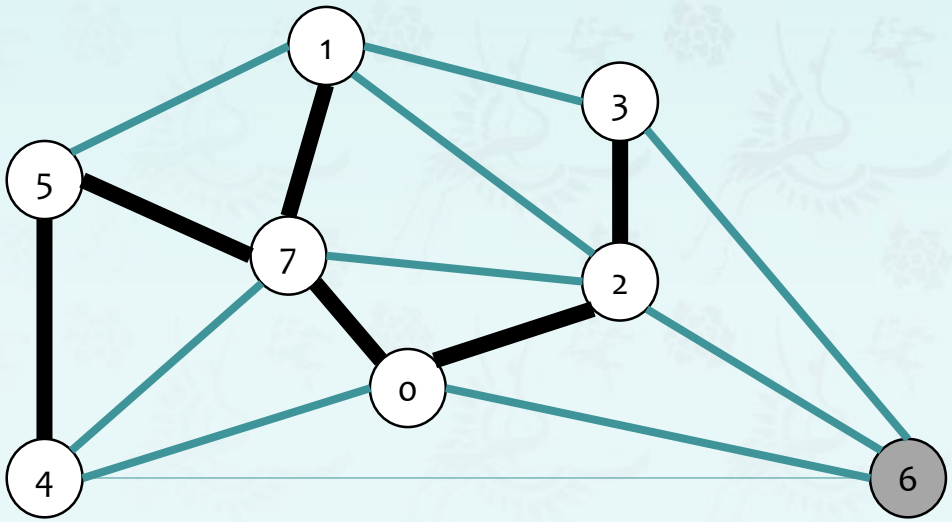
in MST →

4-5	0.35
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

MST edges: 0-7 1-7 0-2 2-3 5-7

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



edges with exactly one endpoint in T (sorted by weight)

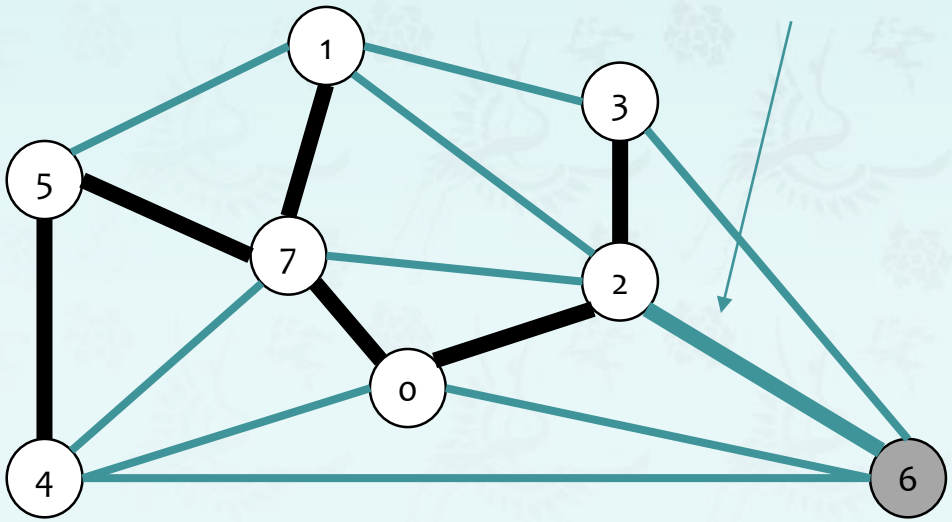
in MST →

4-5	0.35
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

MST edges: 0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



edges with exactly one endpoint in T (sorted by weight)

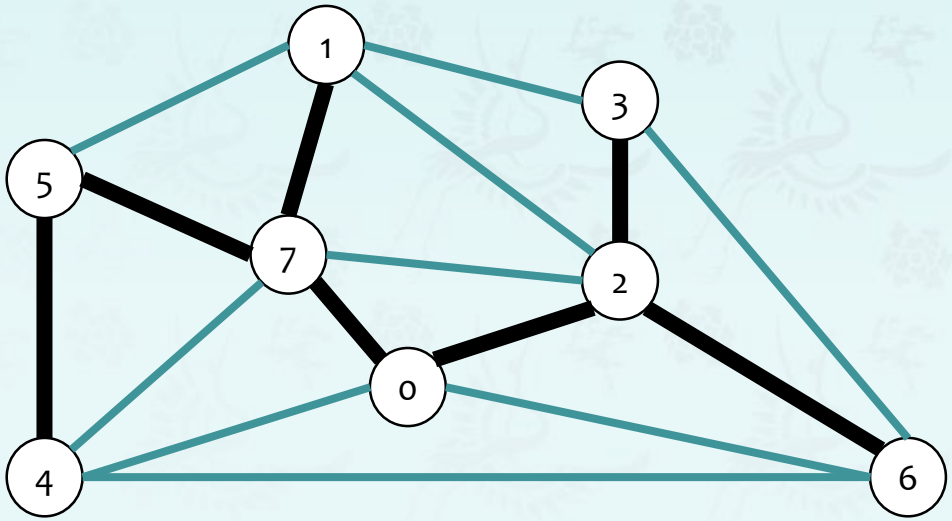
in MST →

6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

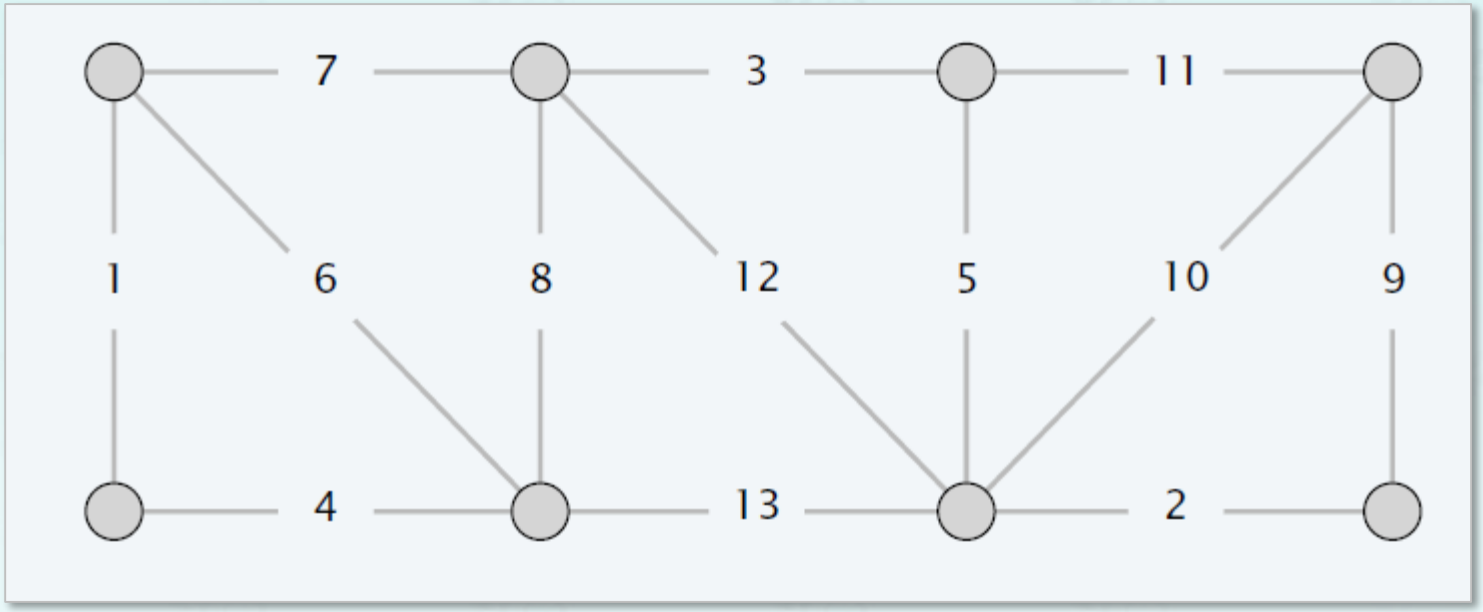
MST edges: 0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm Demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until V-1 edges.



MST edges: 0-7 1-7 0-2 2-3 5-7 4-5


7
3
11
1
6
8
12
5
10
9
4
13
2

ECE20010 Data Structures

Chapter 6

- *Graph, Digraph*
- *Minimum Spanning Tree (MST)*
  - *Introduction*
  - *greedy algorithm*
  - *Kruskals's algorithm*
  - *Prim's algorithm*

Major references:
1. Fundamentals of Data Structures by Horowitz, Sahni, Anderson-Freed,
2. Algorithms 4th edition - Part 1 & Part 2 by Robert Sedgewick and Kevin Wayne
3. Wikipedia and many resources available from internet

Prof. Youngsup Kim, idebtor@handong.edu, 2014 Data Structures, CSEE Dept., Handong Global University