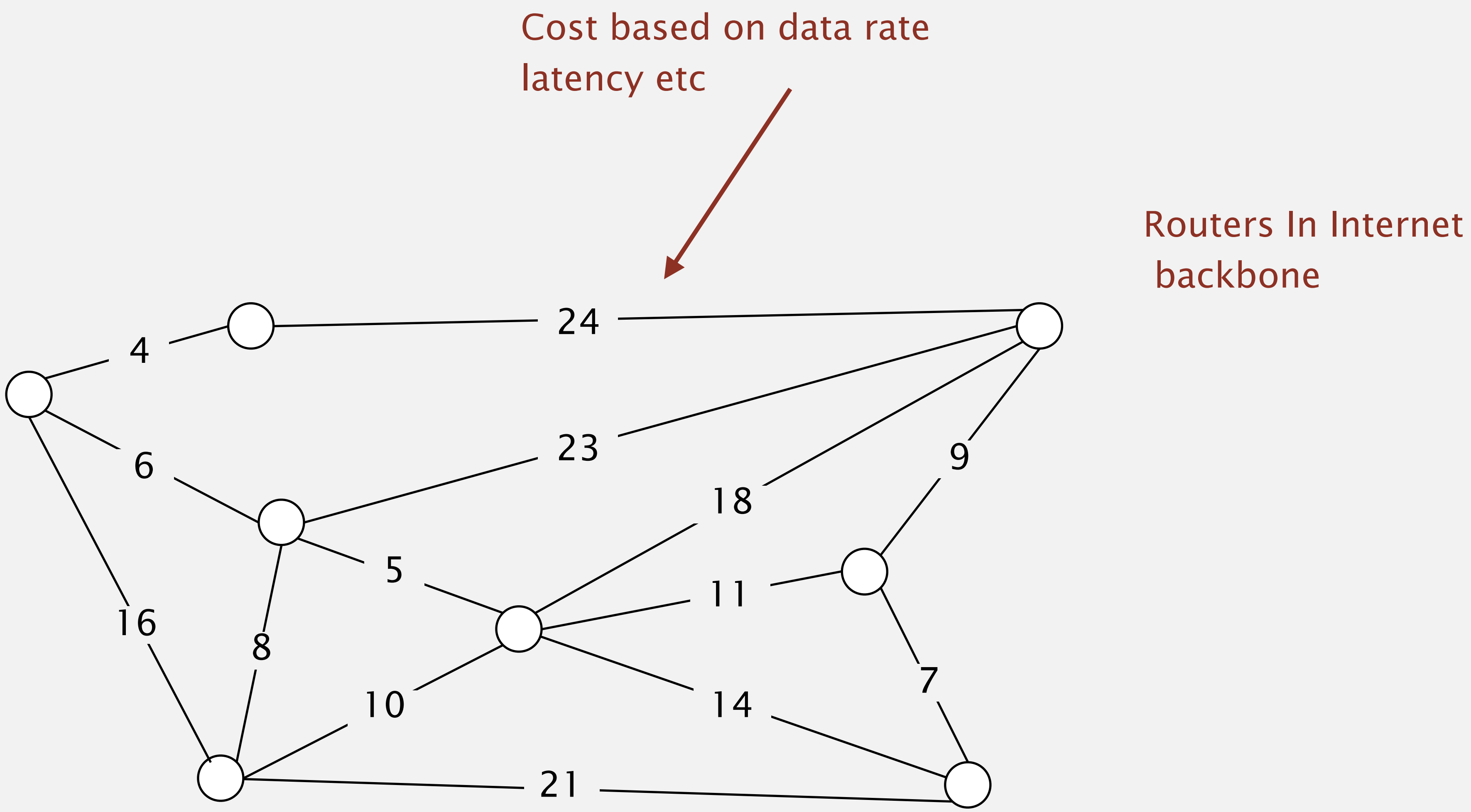
Deciding on routing

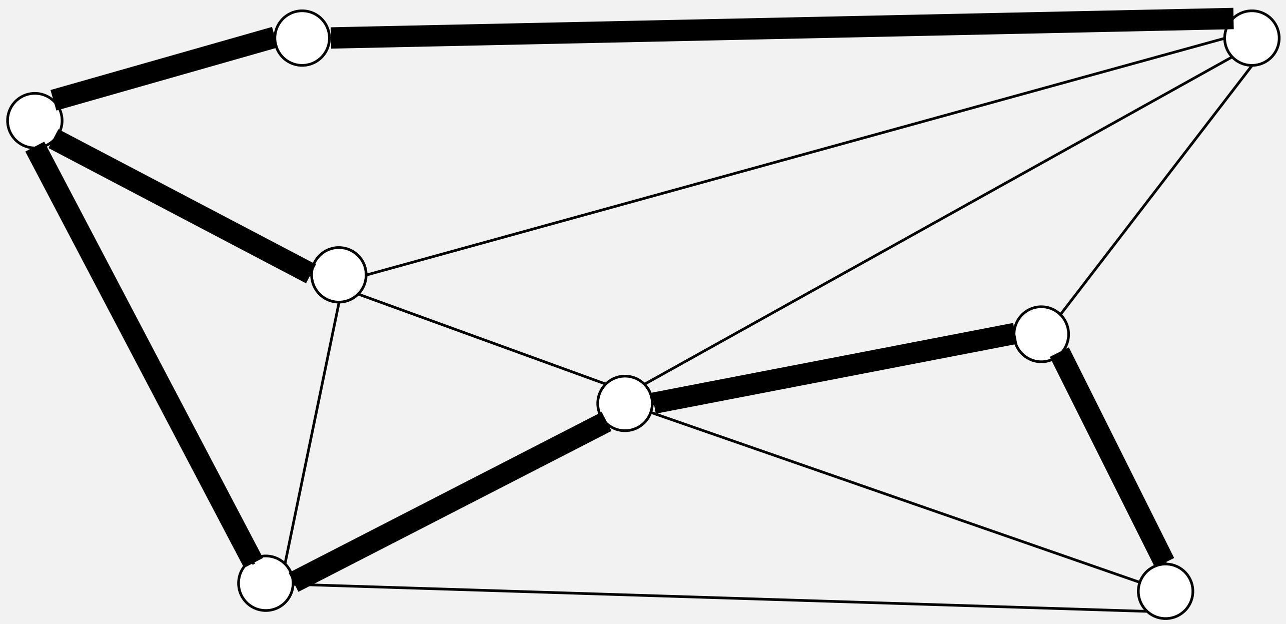


Problem find the collection of forwarding paths that allow each router to reach every other router with the least cost

Minimum spanning tree

Def. A spanning tree of G is a subgraph T that is:

- Connected.
- Acyclic.
- Includes all of the vertices.

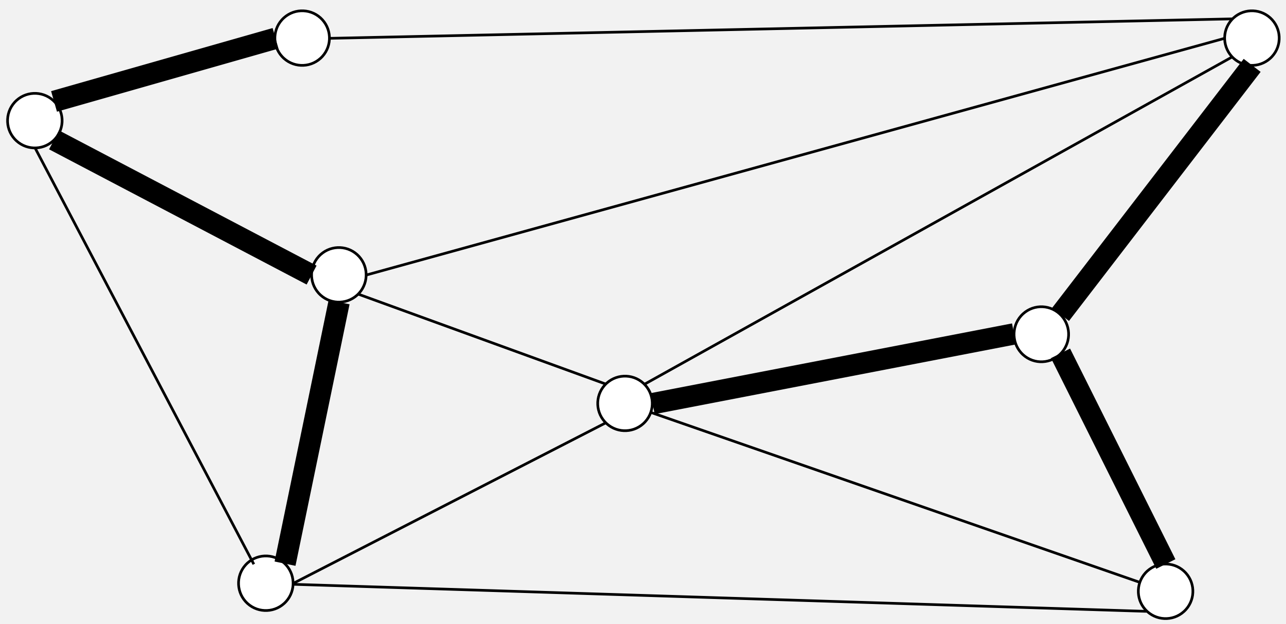


graph G

Minimum spanning tree

Def. A spanning tree of G is a subgraph T that is:

- Connected.
- Acyclic.
- Includes all of the vertices.

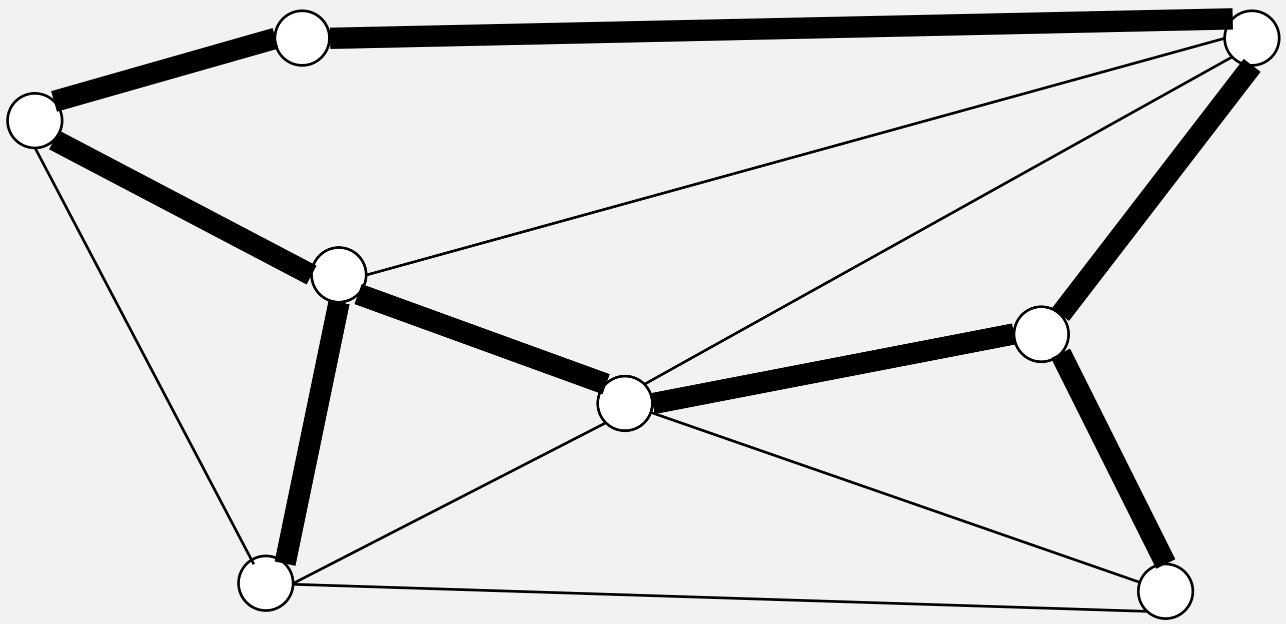


not connected

Minimum spanning tree

Def. A spanning tree of G is a subgraph T that is:

- Connected.
- Acyclic.
- Includes all of the vertices.

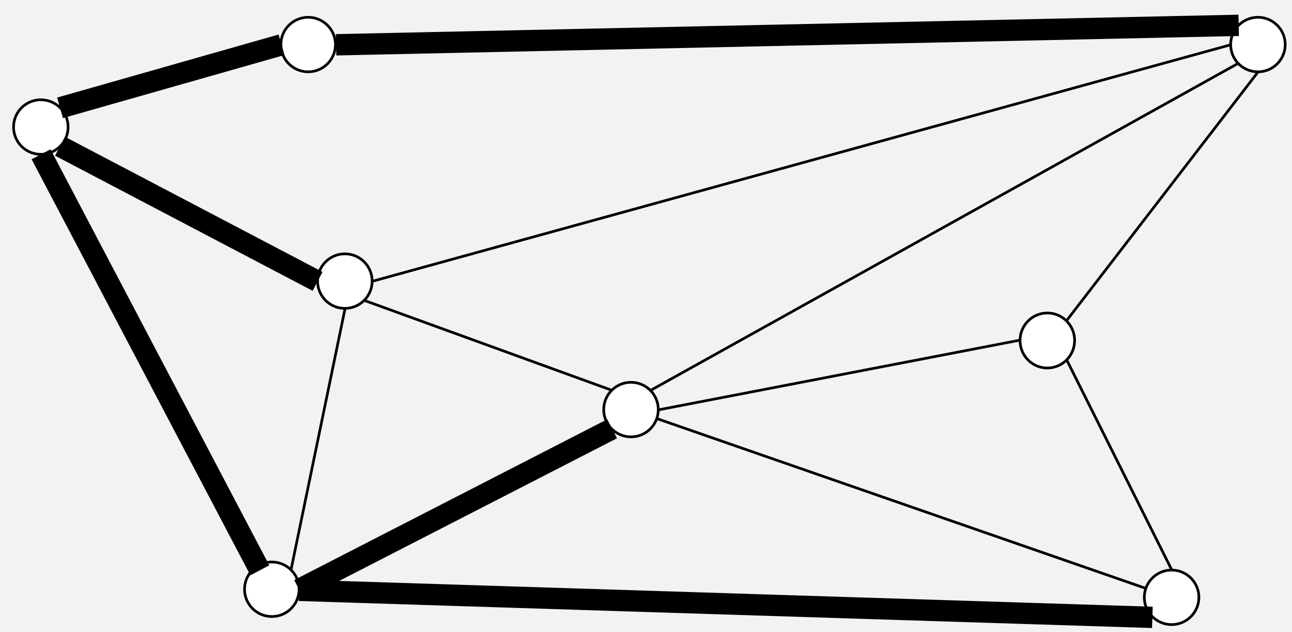


not acyclic

Minimum spanning tree

Def. A spanning tree of G is a subgraph T that is:

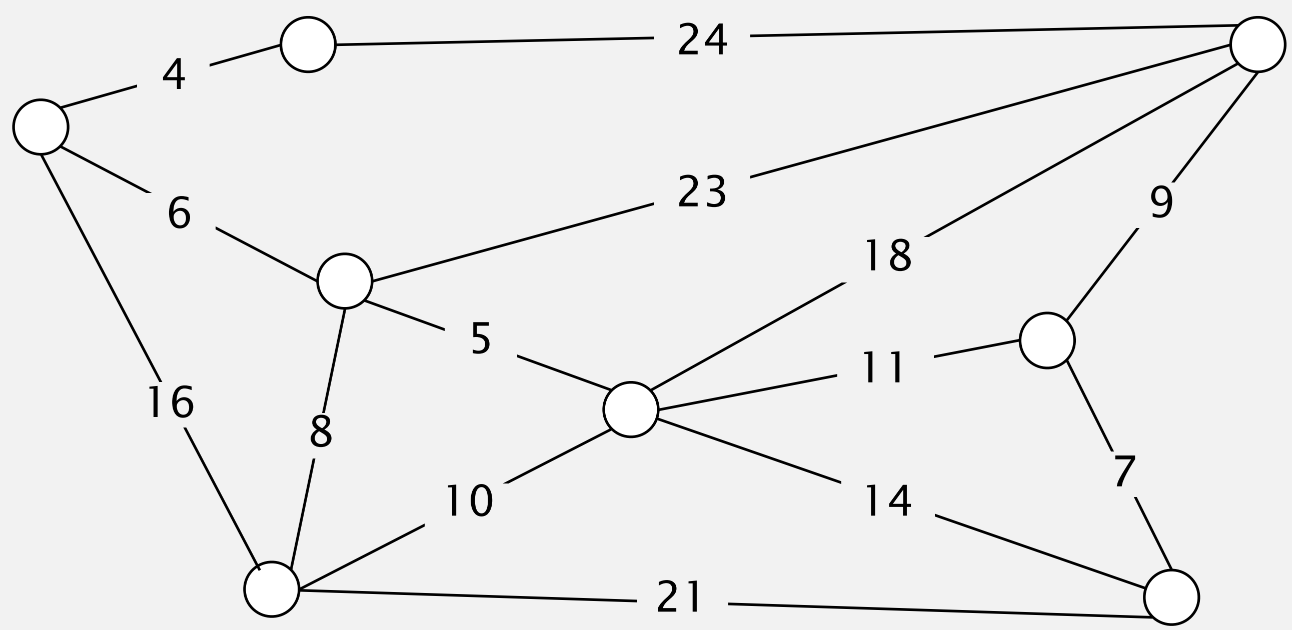
- Connected.
- Acyclic.
- Includes all of the vertices.



not spanning

Minimum spanning tree problem

Input. Connected, undirected graph G with positive edge weights.

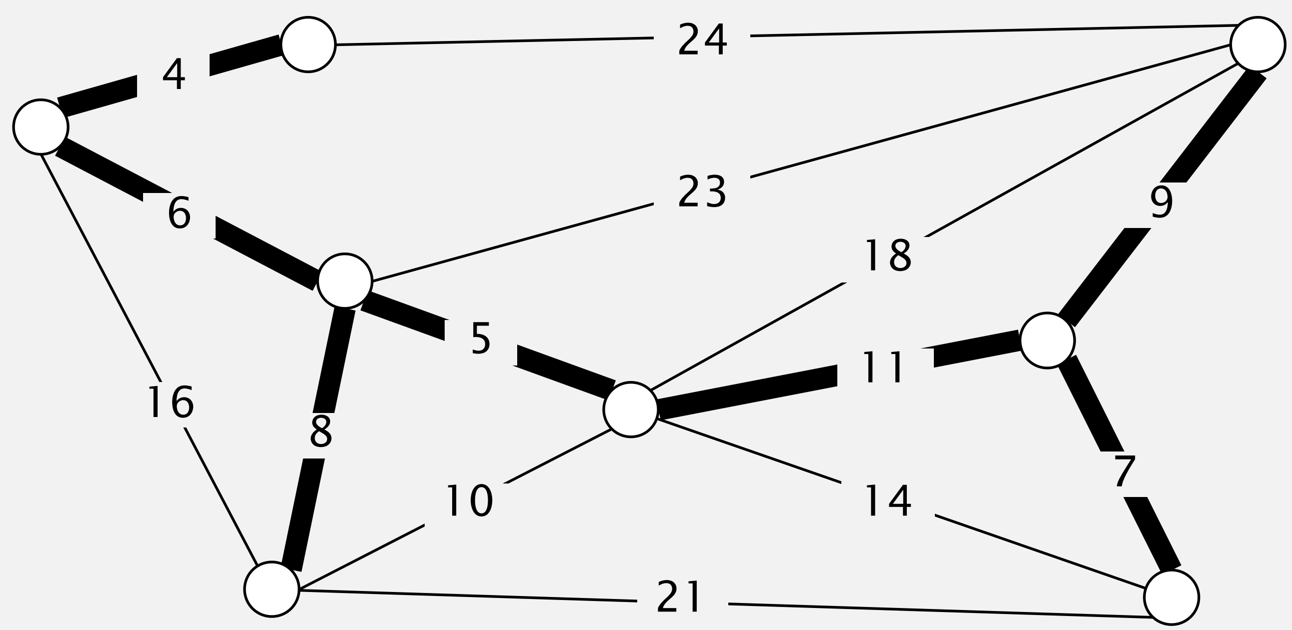


edge-weighted graph G

Minimum spanning tree problem

Input. Connected, undirected graph G with positive edge weights.

Output. A min weight spanning tree.

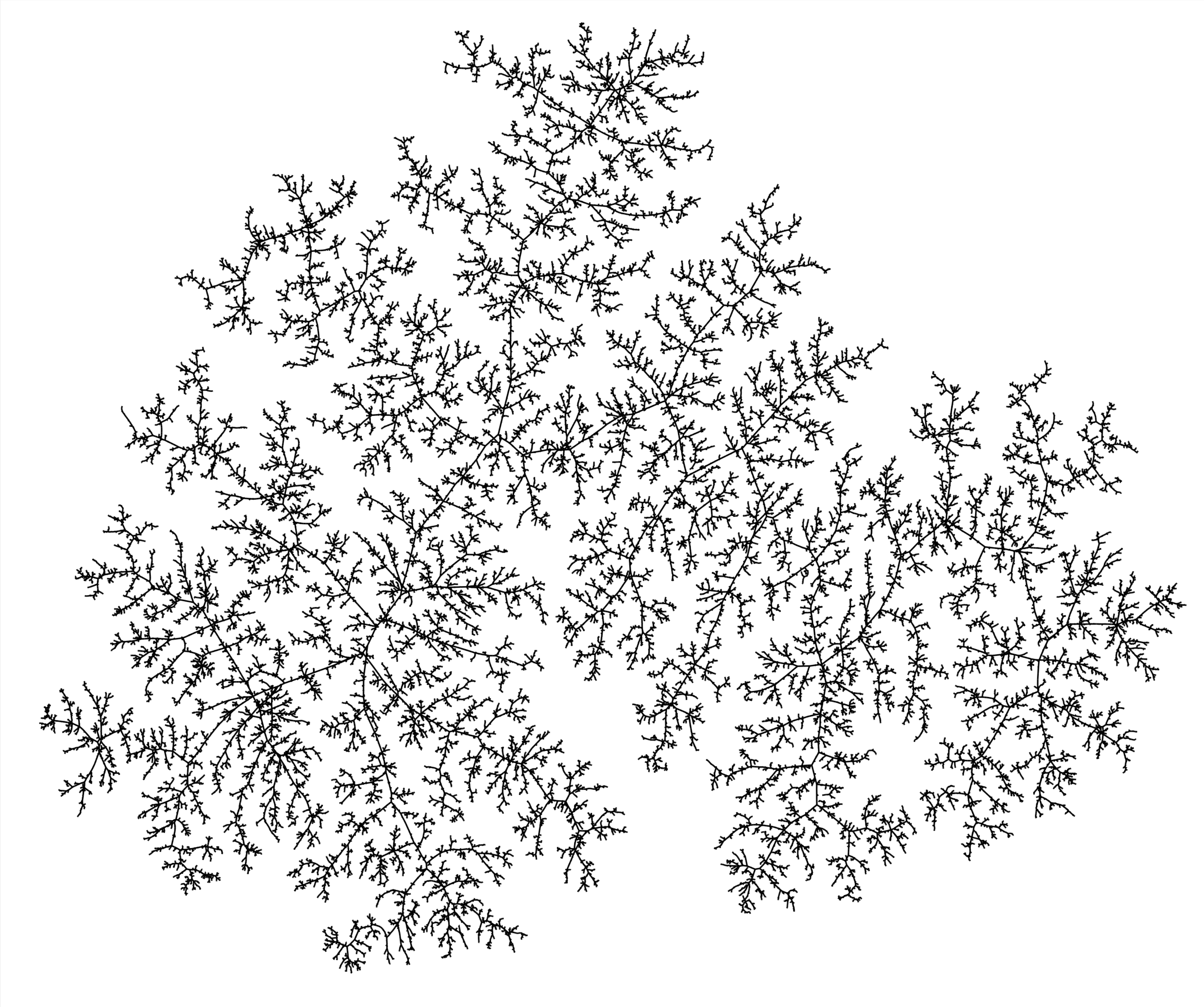


minimum spanning tree T
(weight = 50 = 4 + 6 + 8 + 5 + 11 + 9 + 7)

Brute force. Try all spanning trees?

Models of nature

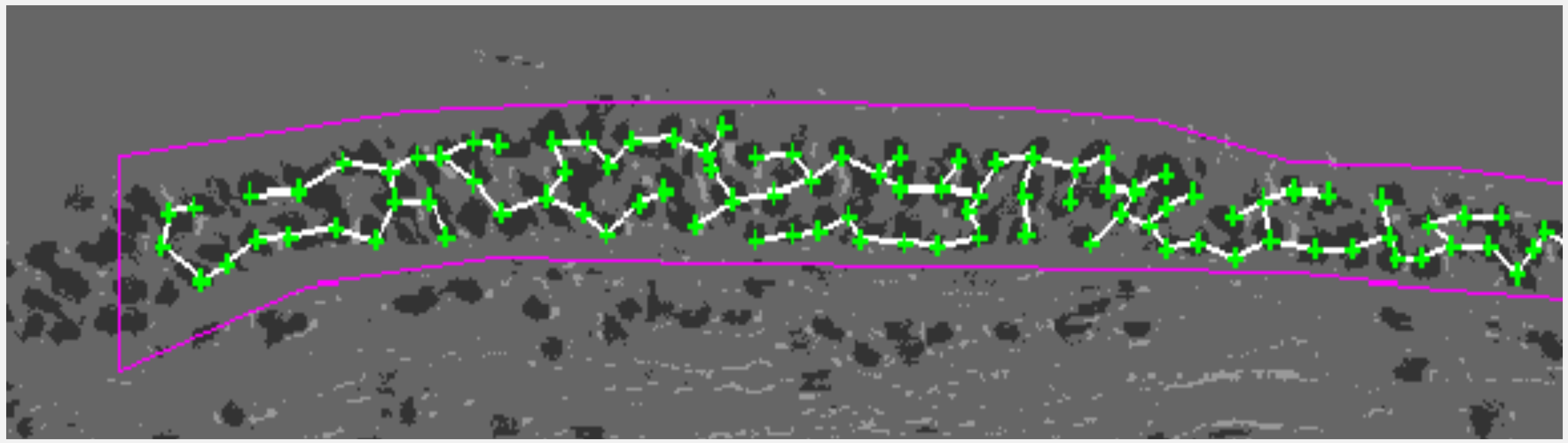
MST of random graph

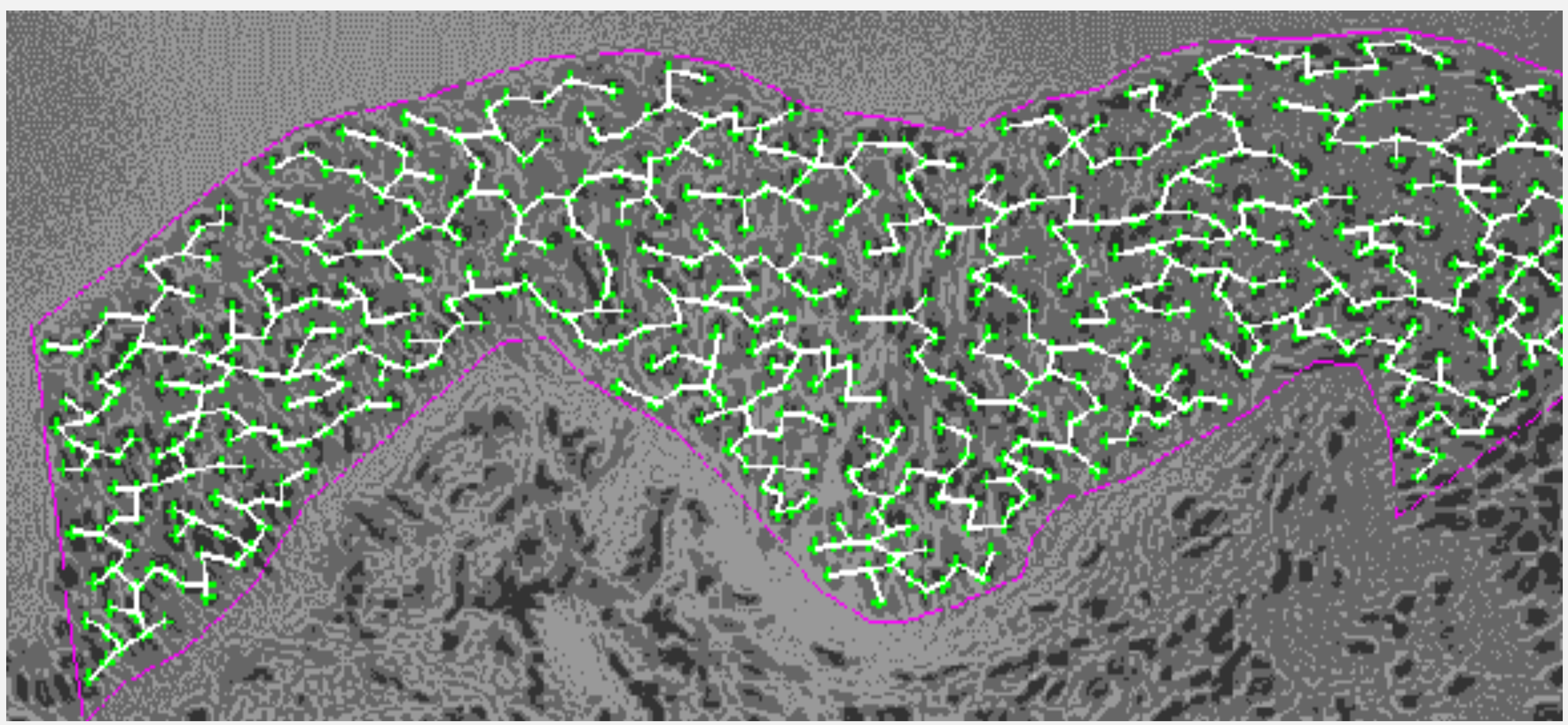


http://algo.inria.fr/broutin/gallery.html

Medical image processing

MST describes arrangement of nuclei in the epithelium for cancer research





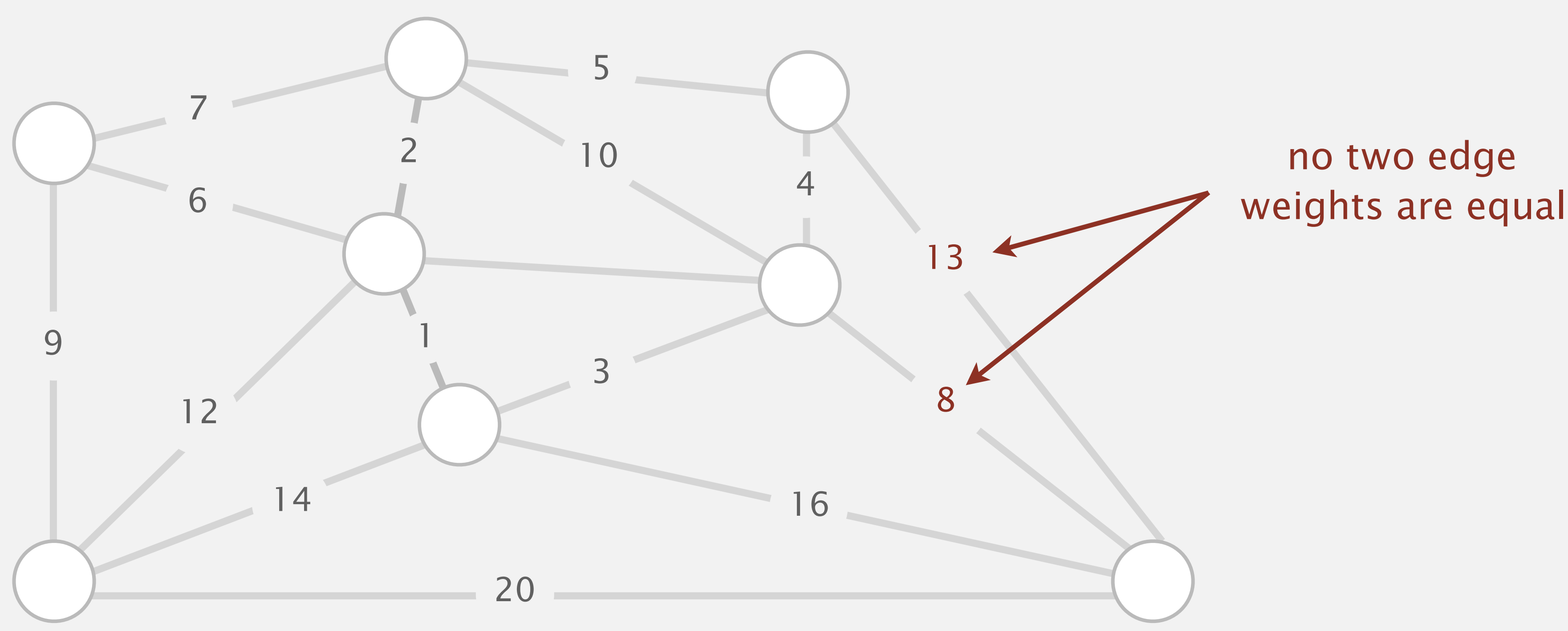
http://www.bccrc.ca/ci/ta01_archlevel.html

Simplifying assumptions

- Graph is connected.
- Edge weights are distinct.

Consequence. MST exists and is unique.

Pf. Exercise for the bored.

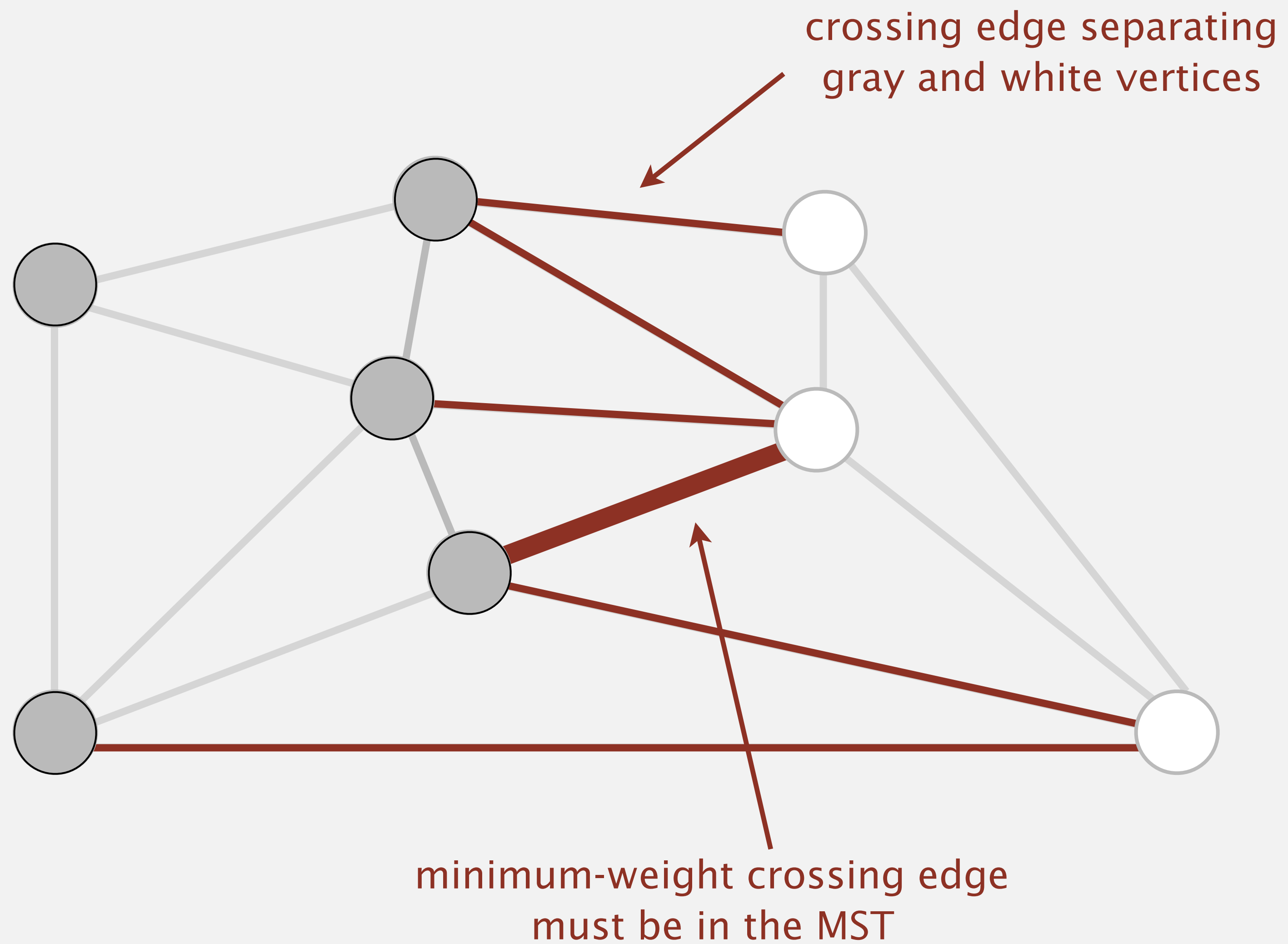


Cut property

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.

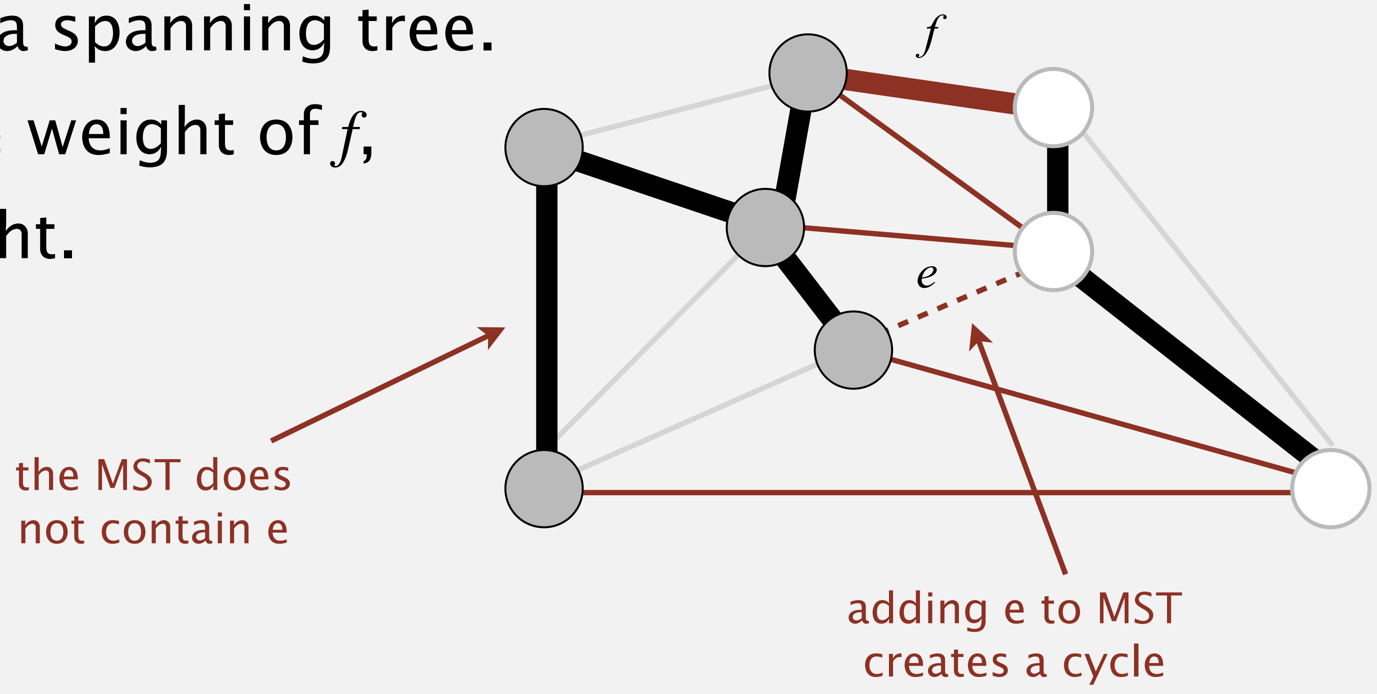


Cut property: correctness proof

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.

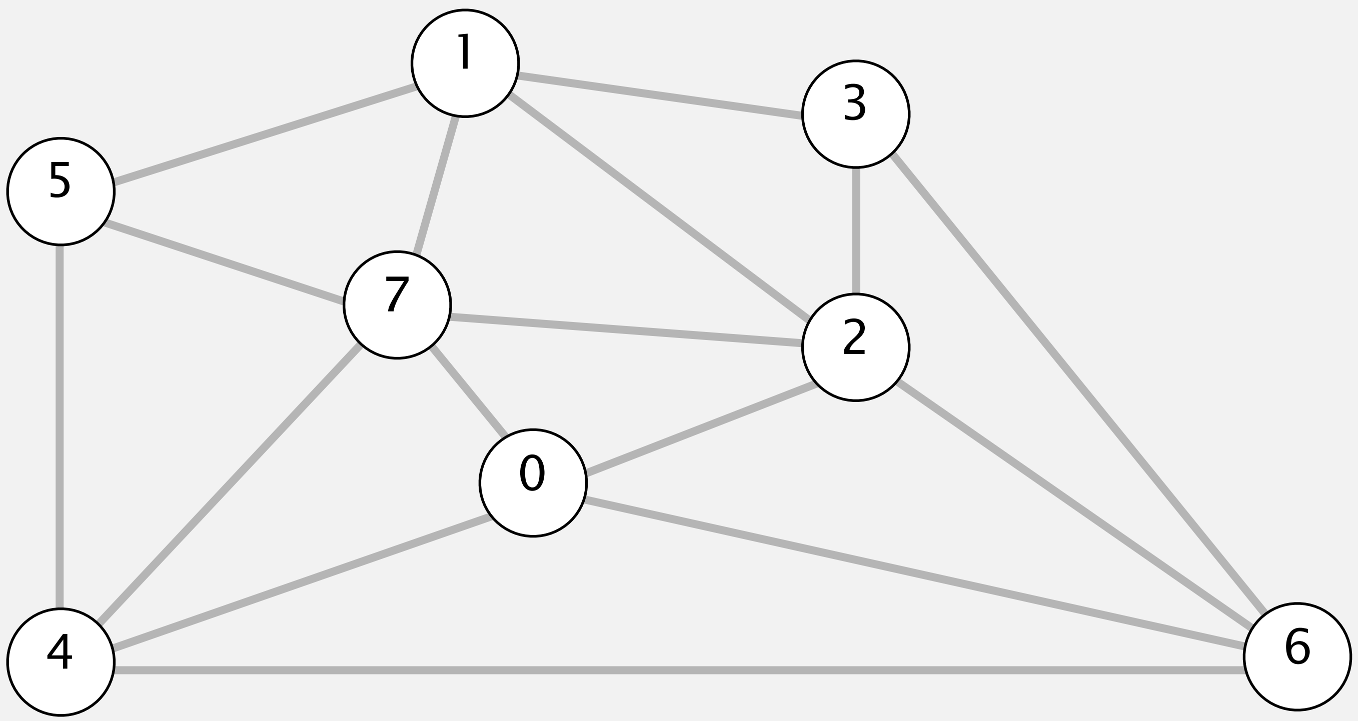
Cut property. Given any cut, the crossing edge of min weight is in the MST.
Pf (By Contradition). Suppose min-weight crossing edge e is not in the MST.

- Adding e to the MST creates 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.

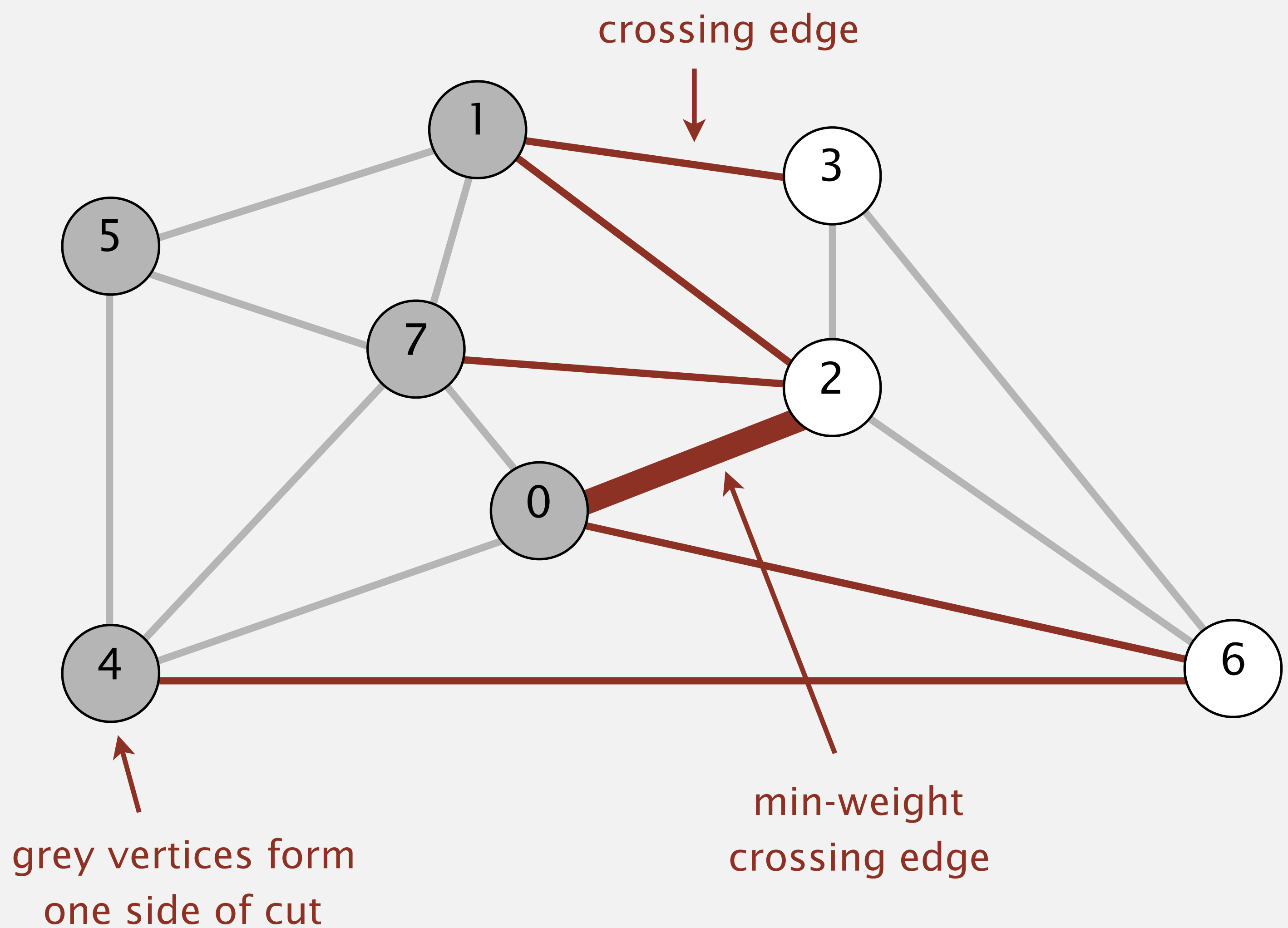


an edge-weighted graph

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.

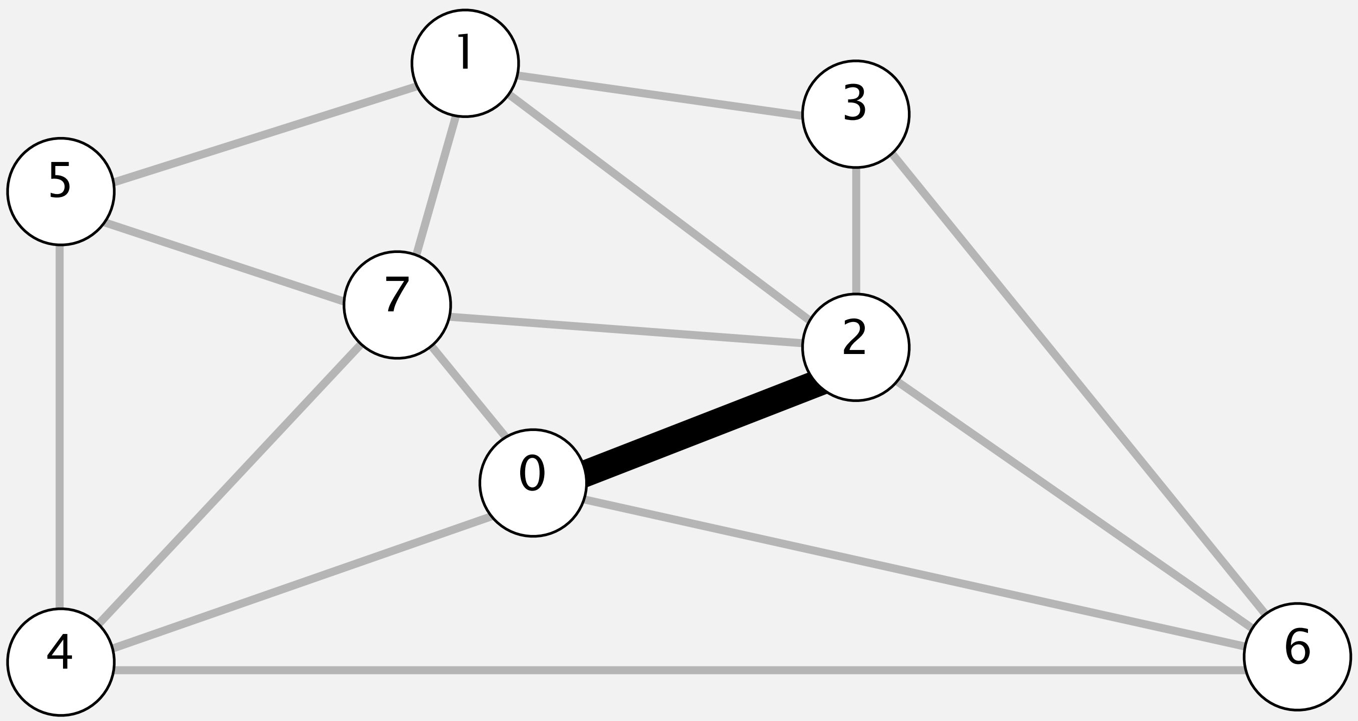


crossing edges
(sorted by weight)

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.

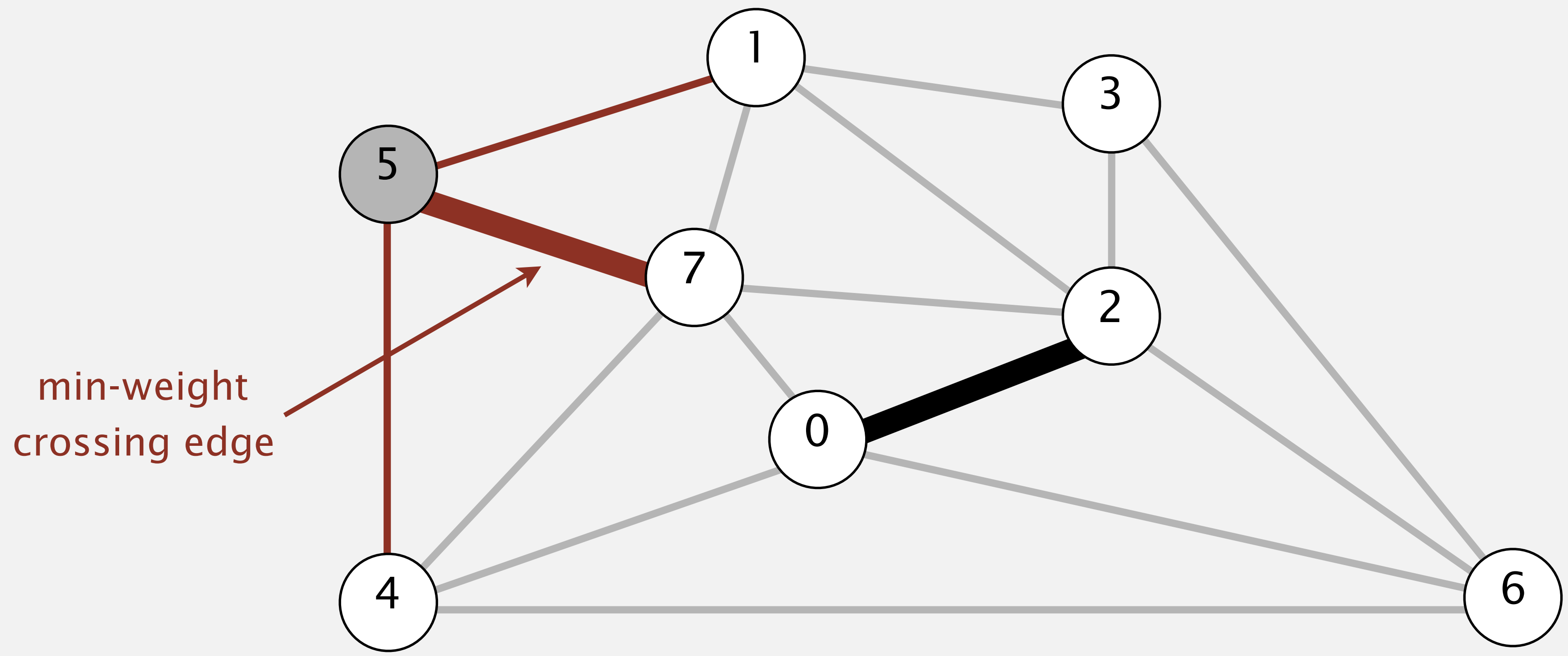


MST edges

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)

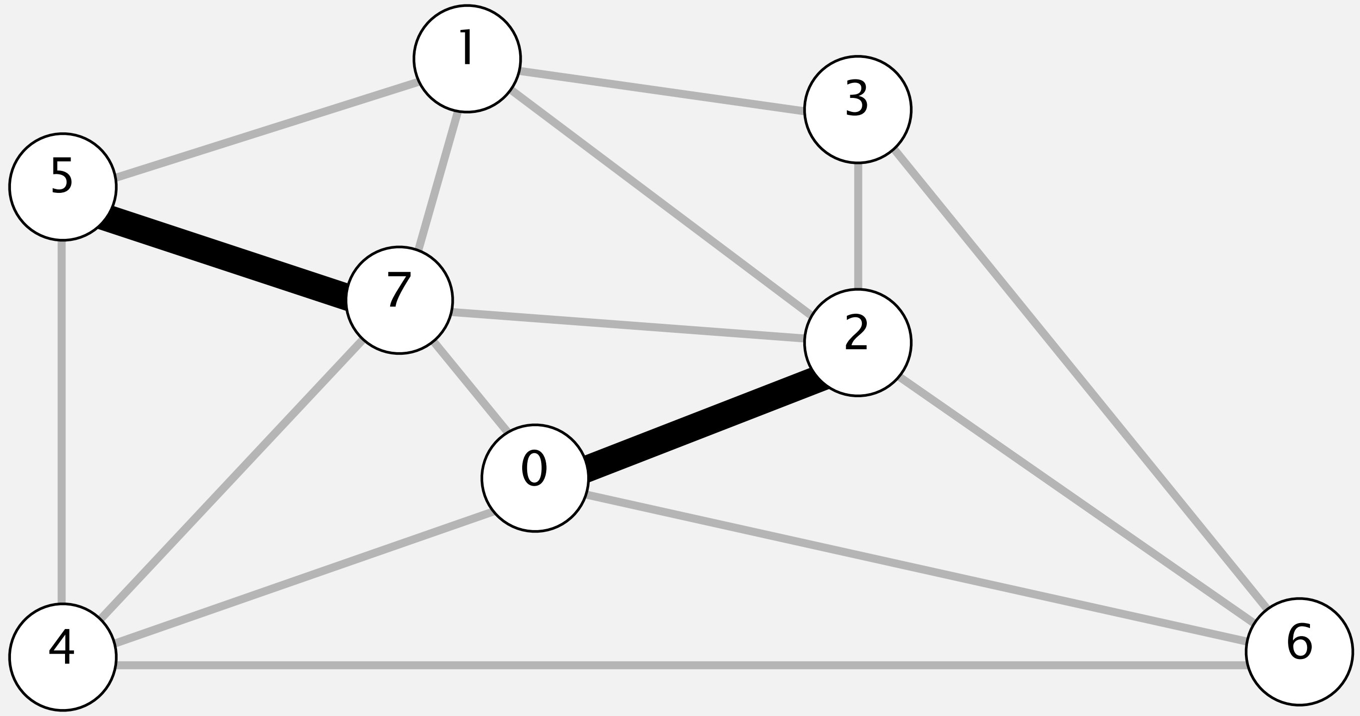
in MST →	5-7	0.28
	1-5	0.32
	4-5	0.35

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.

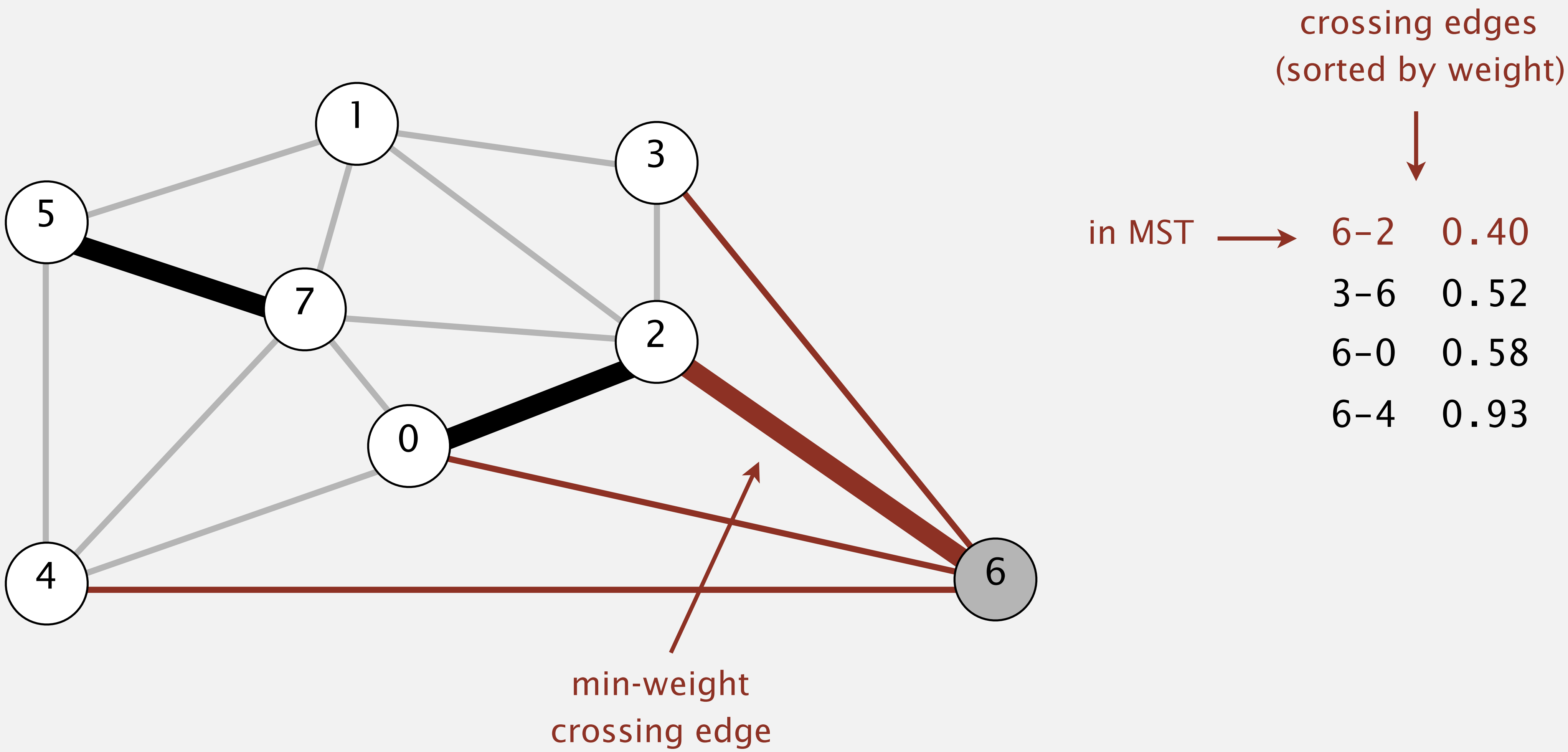


MST edges

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.

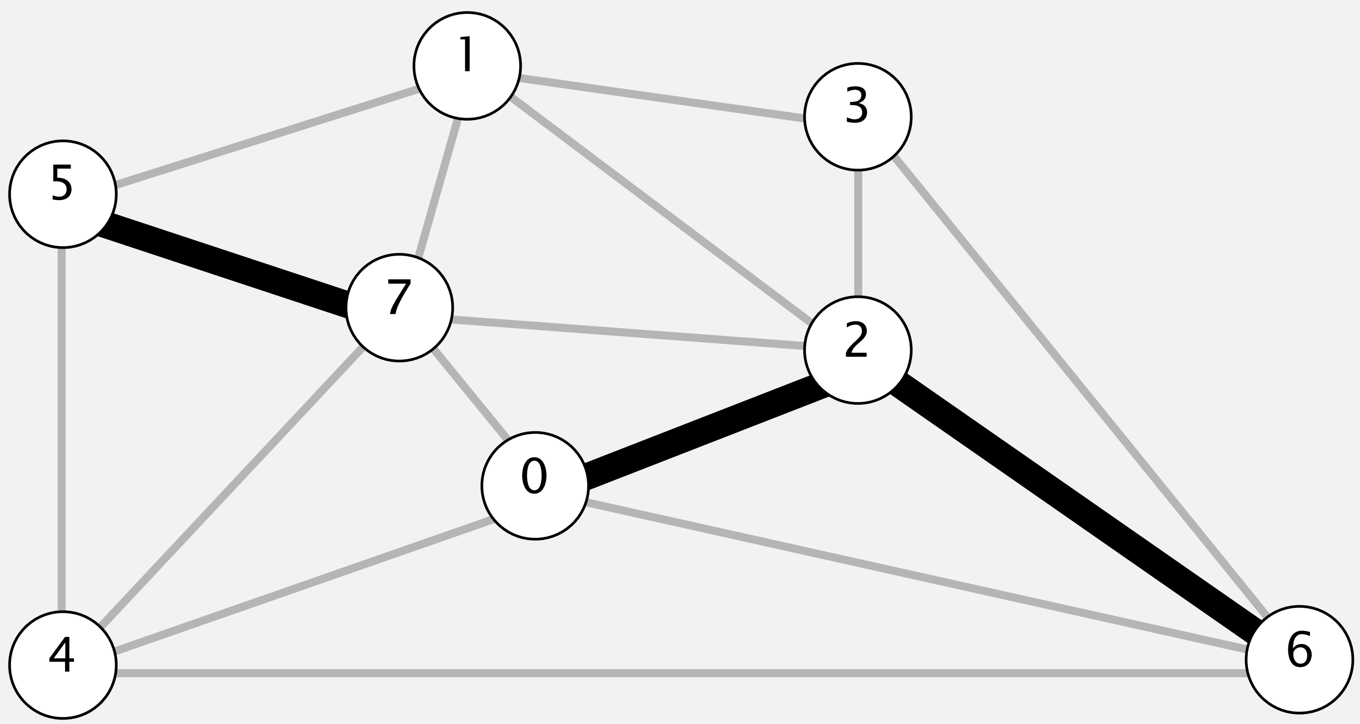


MST edges

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.

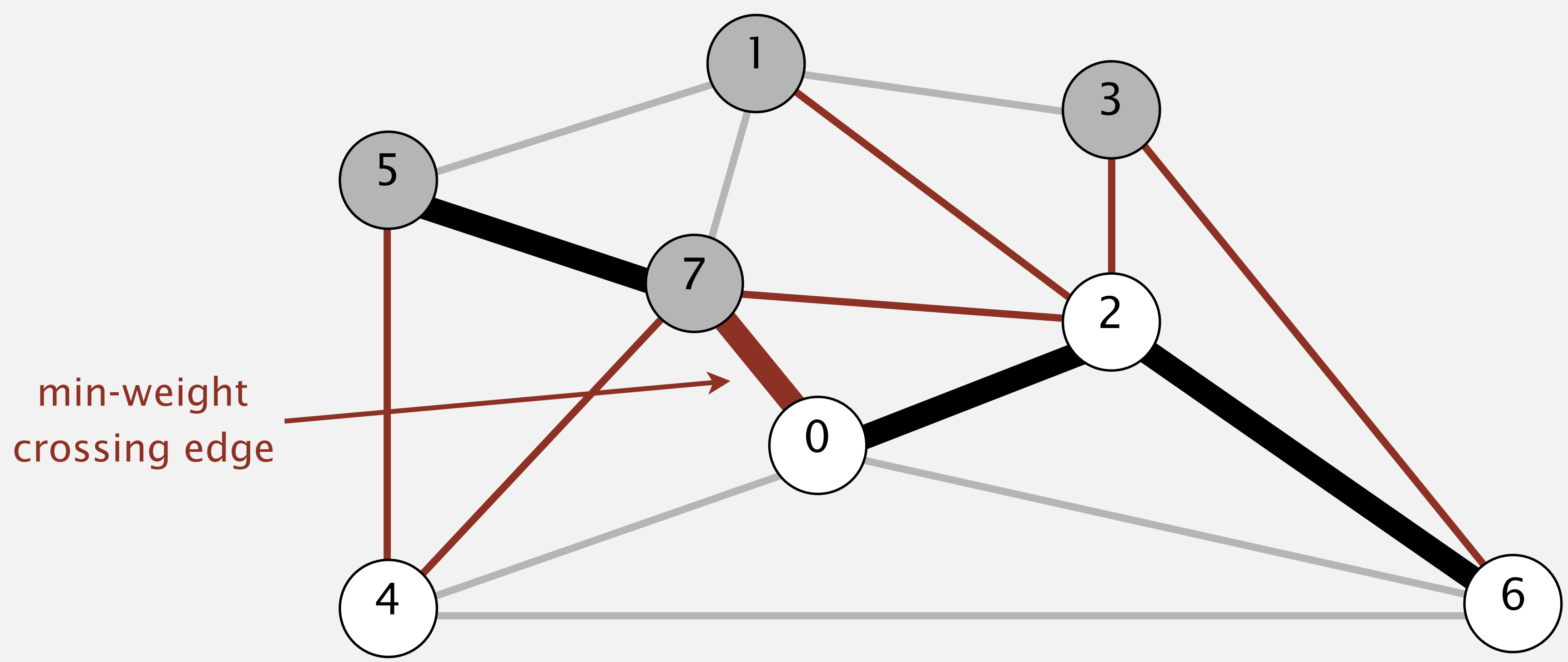


MST edges

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)

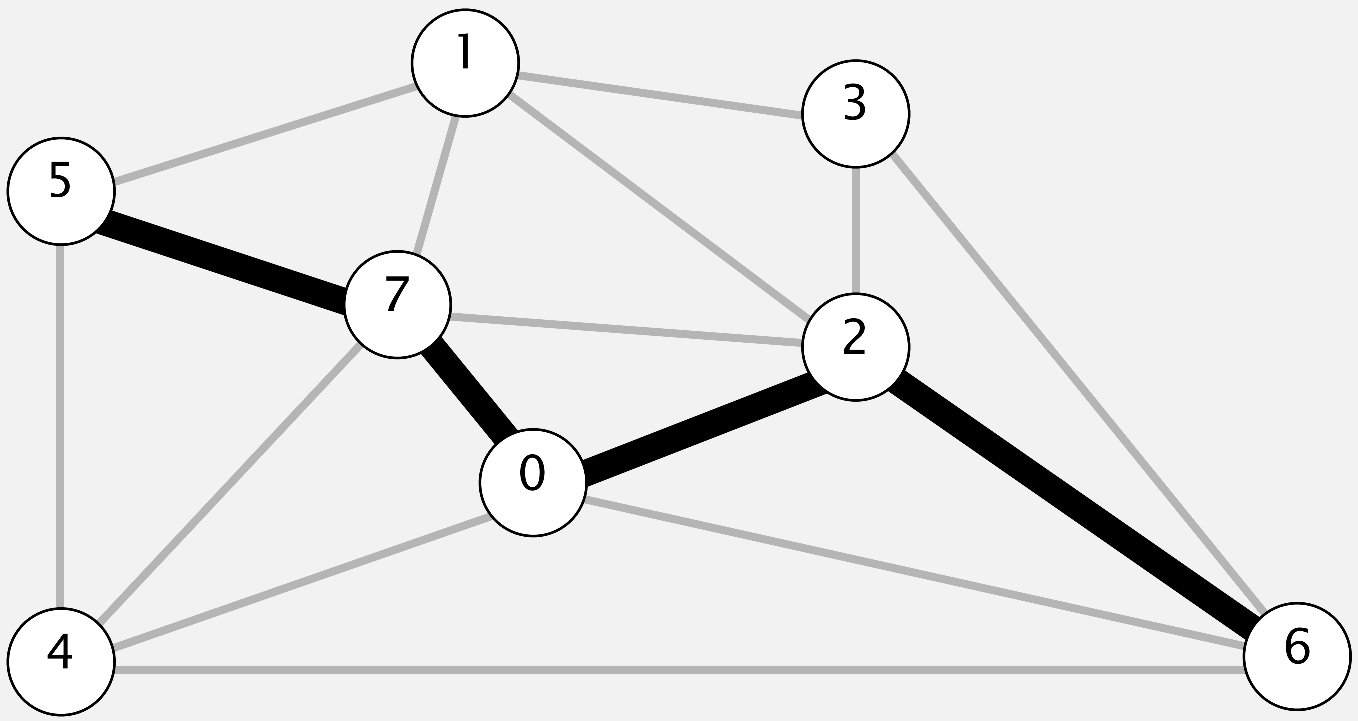
in MST →	0-7	0.16
	2-3	0.17
	2-7	0.34
	4-5	0.35
	1-2	0.36
	4-7	0.37
	3-6	0.52

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.

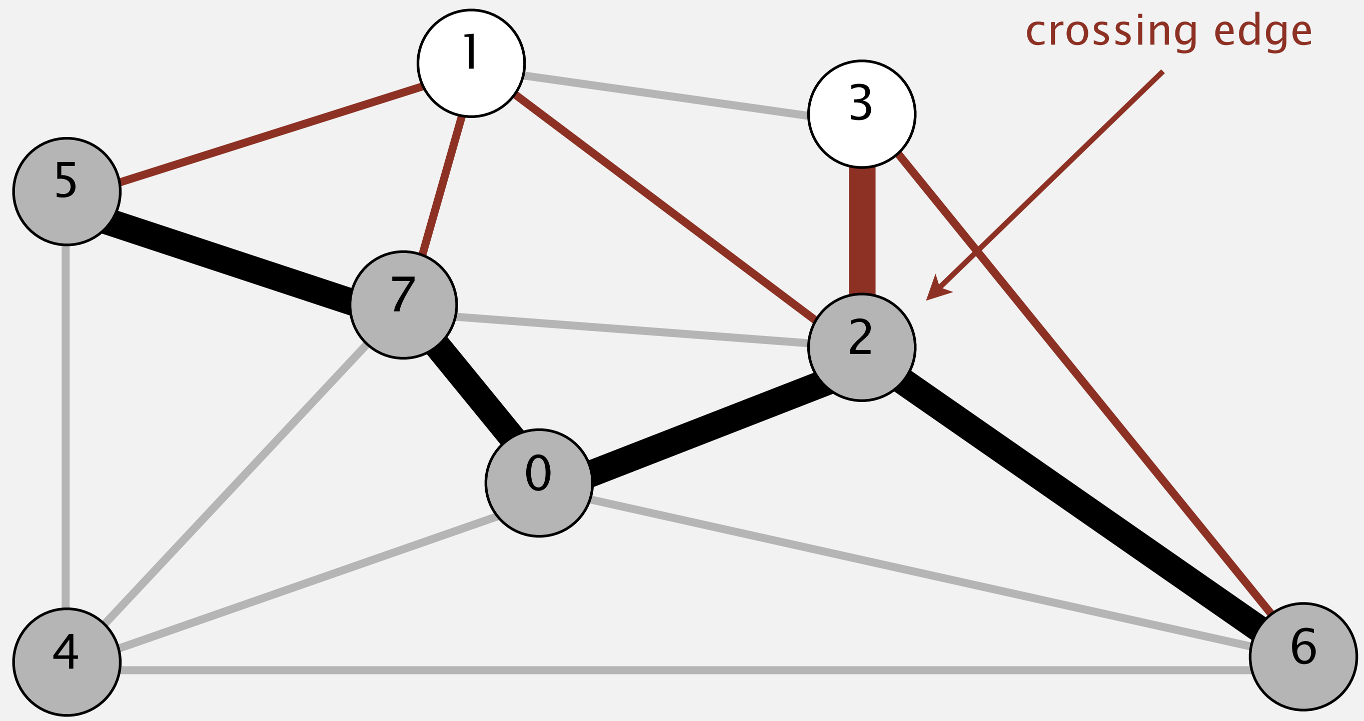


MST edges

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)

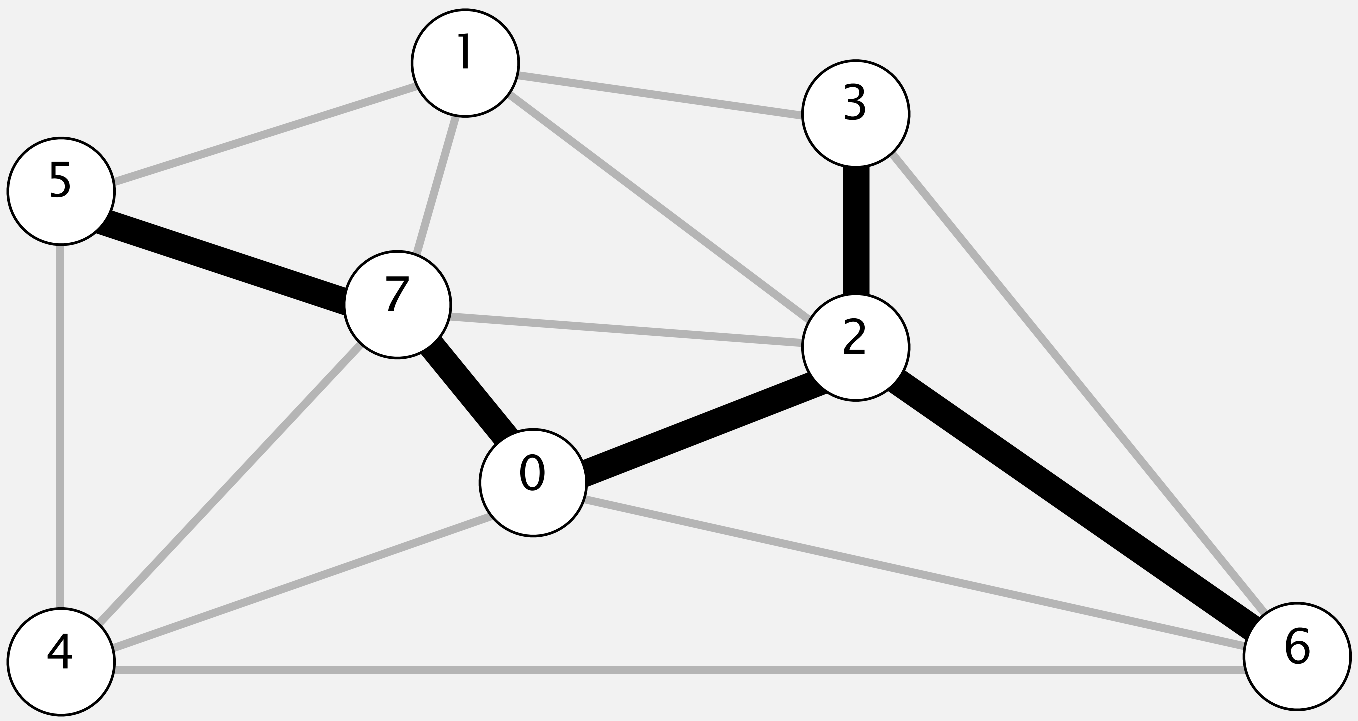
in MST →	2-3	0.17
	1-7	0.19
	1-5	0.32
	1-2	0.36

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.

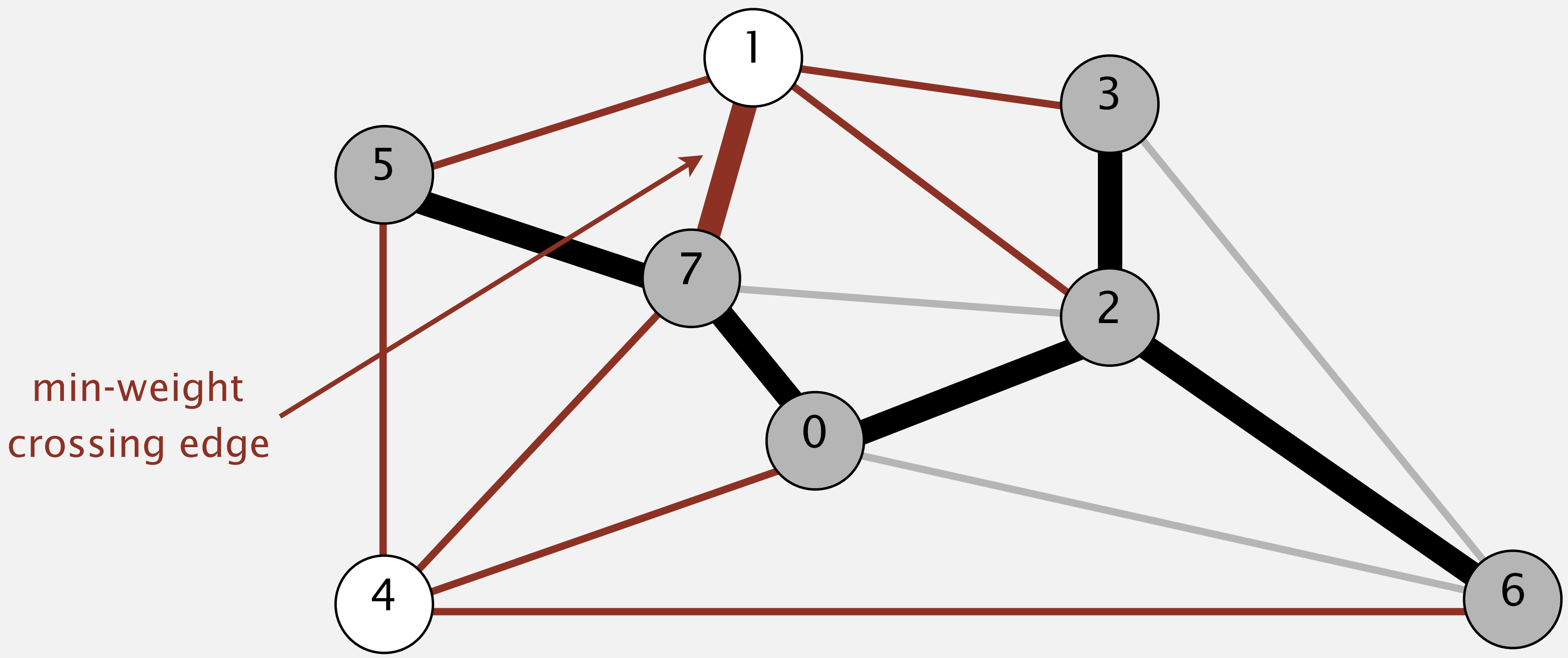


MST edges

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)

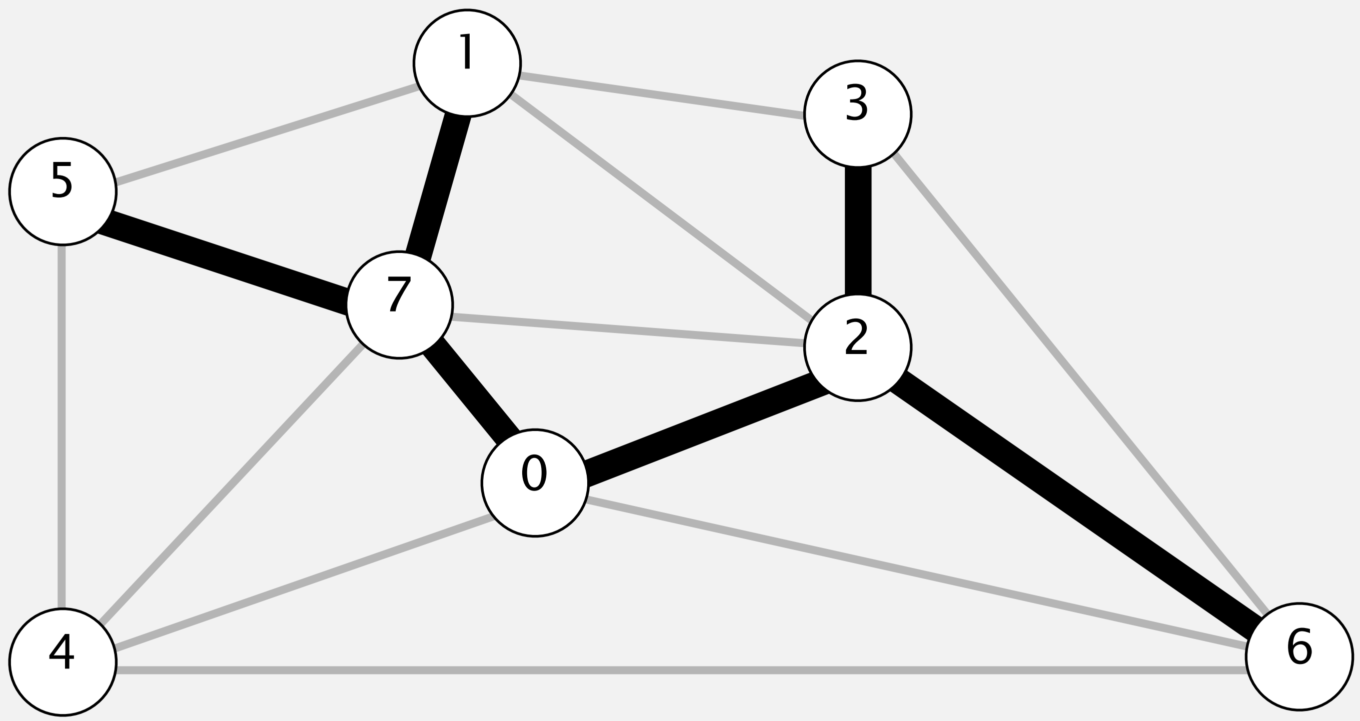
in MST	1-7	0.19
	1-3	0.29
	1-5	0.32
	4-5	0.35
	1-2	0.36
	4-7	0.37
	0-4	0.38
	6-4	0.93

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.

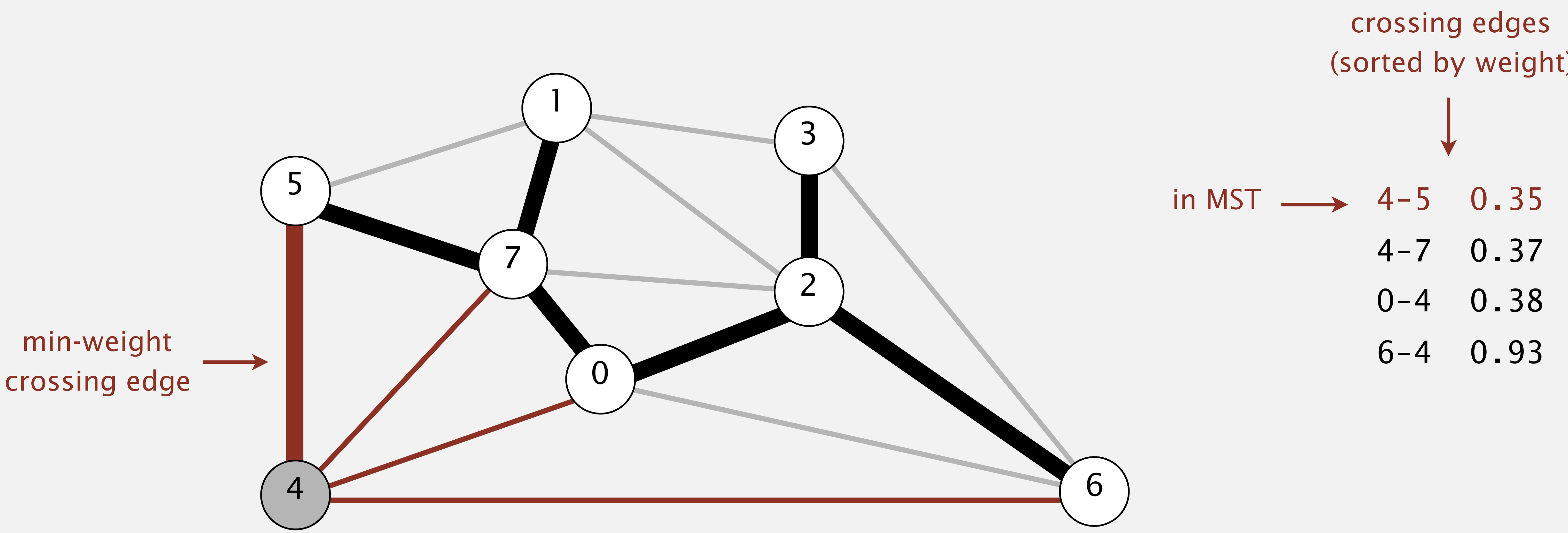


MST edges

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.

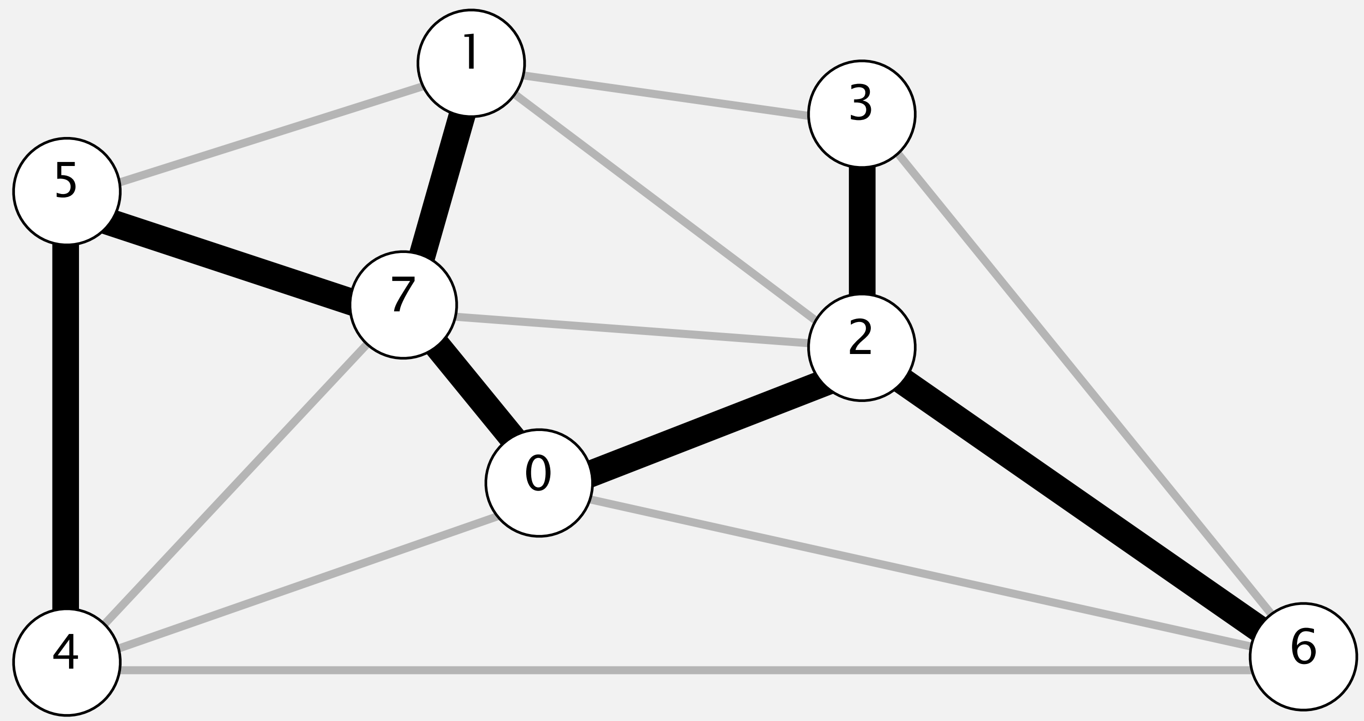


MST edges

0-2 5-7 6-2 0-7 2-3 1-7

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.

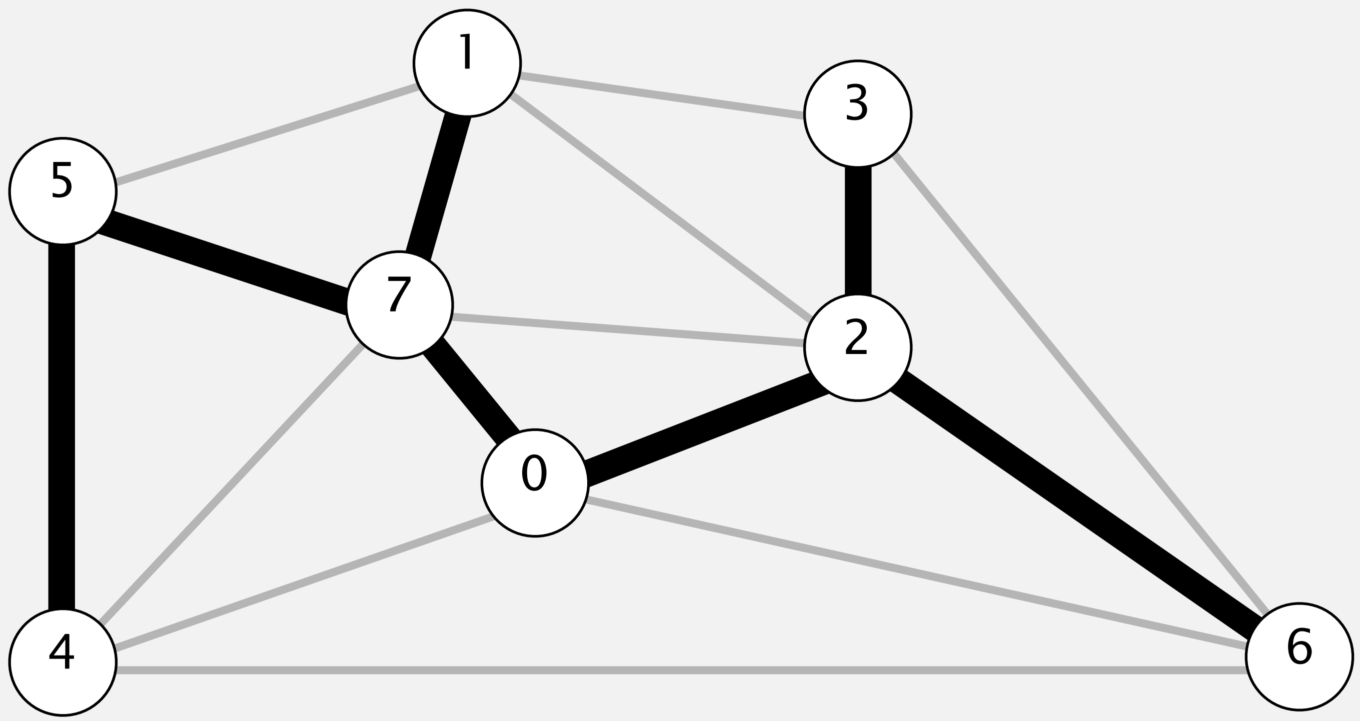


MST edges

0-2 5-7 6-2 0-7 2-3 1-7 4-5

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

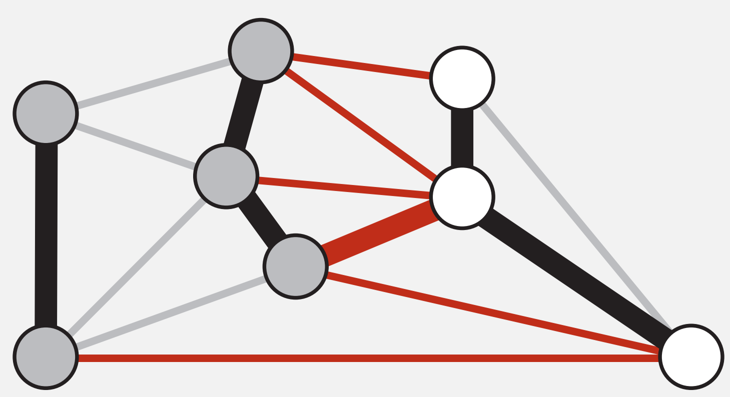
0-2 5-7 6-2 0-7 2-3 1-7 4-5

Greedy MST algorithm: correctness proof

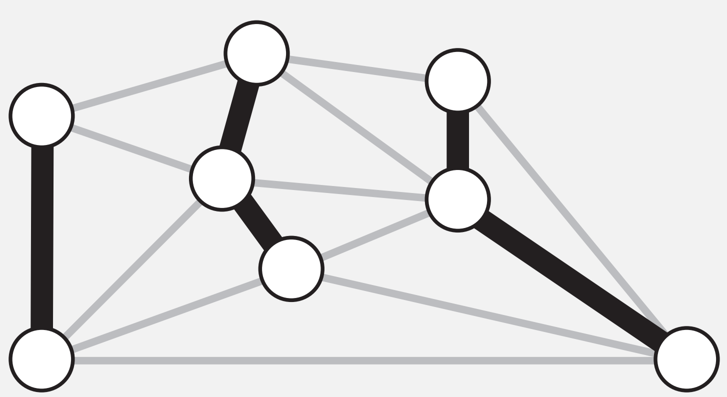
Proposition. The greedy algorithm computes the MST.

Pf.

- Any edge colored black is in the MST (via cut property).
- Fewer than $V-1$ black edges $\Rightarrow$ cut with no black crossing edges.



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?

Ex 1. Kruskal's algorithm. [stay tuned]

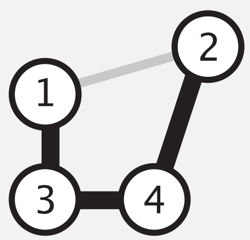
Ex 2. Prim's algorithm. [stay tuned]

Removing two simplifying assumptions

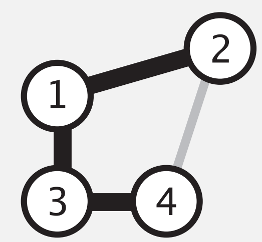
Q. What if edge weights are not all distinct?

A. Greedy MST algorithm still correct if equal weights are present!

(our correctness proof fails, but that can be fixed)



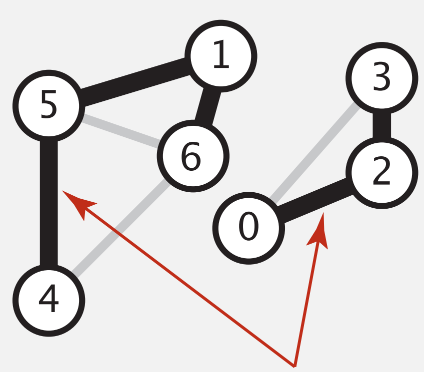
```
1 2  1.00
1 3  0.50
2 4  1.00
3 4  0.50
```



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

Q. What if graph is not connected?

A. Compute minimum spanning forest = MST of each component.



can independently compute MSTs of components

```
4 5  0.61
4 6  0.62
5 6  0.88
1 5  0.11
2 3  0.35
0 3  0.6
1 6  0.10
0 2  0.22
```

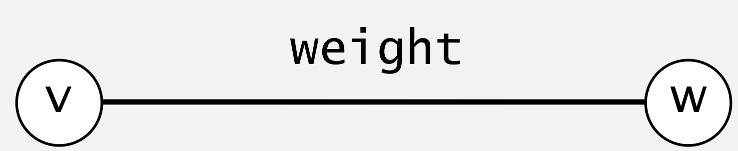
Greed is good

Gordon Gecko (Michael Douglas) address to Teldar Paper Stockholders in Wall Street (1986)

Weighted edge API

Edge abstraction needed for weighted edges.

`public class Edge implements Comparable<Edge>`		
	`Edge(int v, int w, double weight)`	*create a weighted edge v-w*
`int`	`either()`	*either endpoint*
`int`	`other(int v)`	*the endpoint that's not v*
`int`	`compareTo(Edge that)`	*compare this edge to that edge*
`double`	`weight()`	*the weight*
`String`	`toString()`	*string representation*



Idiom for processing an edge `e`: `int v = e.either(), w = e.other(v);`

Weighted edge: Java implementation

```
public class Edge implements Comparable<Edge>
{
   private final int v, w;
   private final double weight;

   public Edge(int v, int w, double weight)
   {
      this.v = v;
      this.w = w;
      this.weight = weight;
   }

   public int either()
   {  return v;  }

   public int other(int vertex)
   {
      if (vertex == v) return w;
      else return v;
   }

   public int compareTo(Edge that)
   {
      if      (this.weight < that.weight) return -1;
      else if (this.weight > that.weight) return +1;
      else                                return  0;
   }
}
```

constructor

either endpoint

other endpoint

compare edges by weight

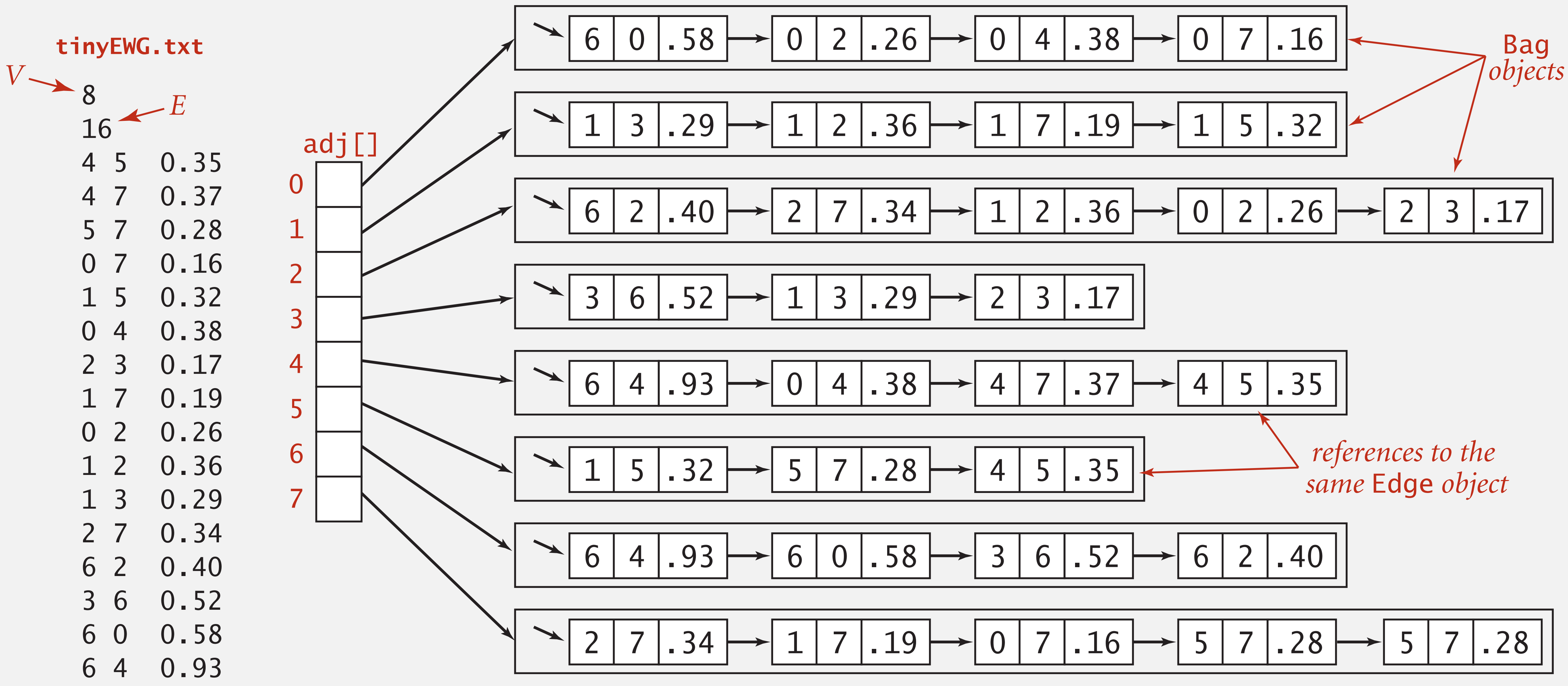
Edge-weighted graph API

	public class EdgeWeightedGraph	
	EdgeWeightedGraph(int V)	*create an empty graph with V vertices*
	EdgeWeightedGraph(In in)	*create a graph from input stream*
void	addEdge(Edge e)	*add weighted edge e to this graph*
Iterable<Edge>	adj(int v)	*edges incident to v*
Iterable<Edge>	edges()	*all edges in this graph*
int	V()	*number of vertices*
int	E()	*number of edges*
String	toString()	*string representation*

Conventions. Allow self-loops and parallel edges.

Edge-weighted graph: adjacency-lists representation

Maintain vertex-indexed array of Edge lists.



Edge-weighted graph: adjacency-lists implementation

```
public class EdgeWeightedGraph
{
   private final int V;
   private final Bag<Edge>[] adj;

   public EdgeWeightedGraph(int V)
   {
      this.V = V;
      adj = (Bag<Edge>[]) new Bag[V];
      for (int v = 0; v < V; v++)
         adj[v] = new Bag<Edge>();
   }

   public void addEdge(Edge e)
   {
      int v = e.either(), w = e.other(v);
      adj[v].add(e);
      adj[w].add(e);
   }

   public Iterable<Edge> adj(int v)
   {  return adj[v];  }
}
```

same as Graph, but adjacency lists of Edges instead of integers

constructor

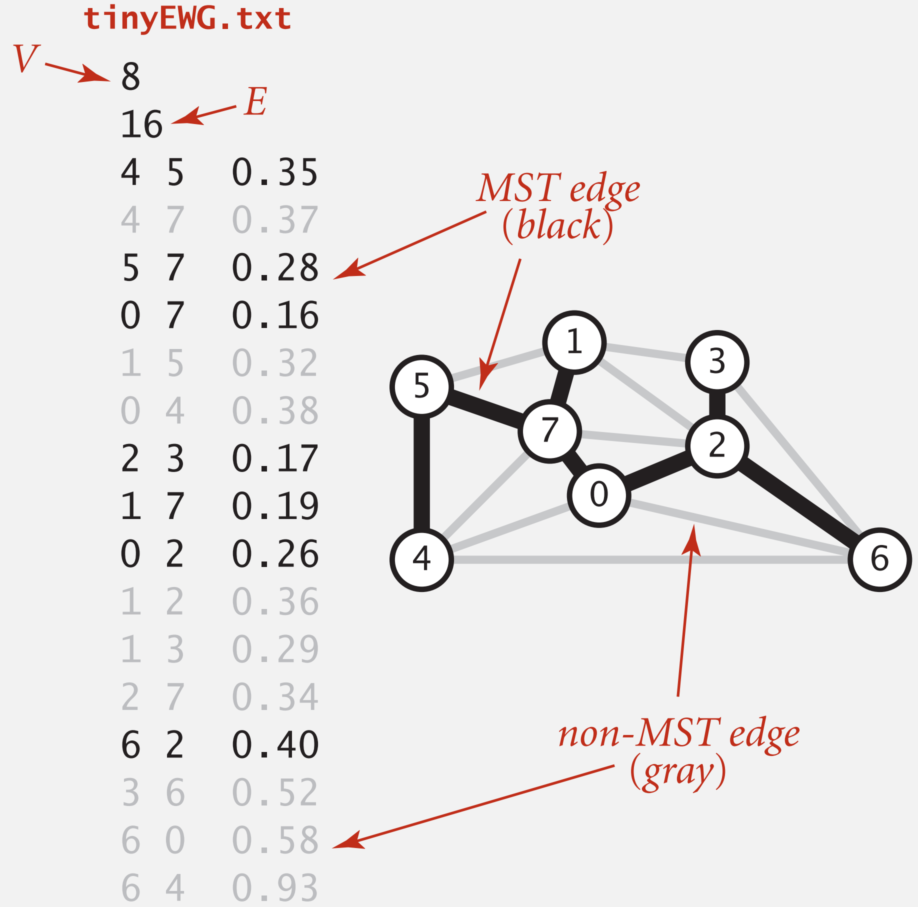
add edge to both adjacency lists

Minimum spanning tree API

Q. How to represent the MST?

```
public class MST
---------------------------------------------------------------
                  MST(EdgeWeightedGraph G)      constructor
Iterable<Edge>    edges()                       edges in MST
        double    weight()                      weight of MST
```

tinyEWG.txt




```
% java MST tinyEWG.txt
0-7 0.16
1-7 0.19
0-2 0.26
2-3 0.17
5-7 0.28
4-5 0.35
6-2 0.40
1.81
```

Minimum spanning tree API

Q. How to represent the MST?

public class MST		
	MST(EdgeWeightedGraph G)	*constructor*
Iterable<Edge>	edges()	*edges in MST*
double	weight()	*weight of MST*

```
public static void main(String[] args)
{
   In in = new In(args[0]);
   EdgeWeightedGraph G = new EdgeWeightedGraph(in);
   MST mst = new MST(G);
   for (Edge e : mst.edges())
      StdOut.println(e);
   StdOut.printf("%.2f\n", mst.weight());
}
```

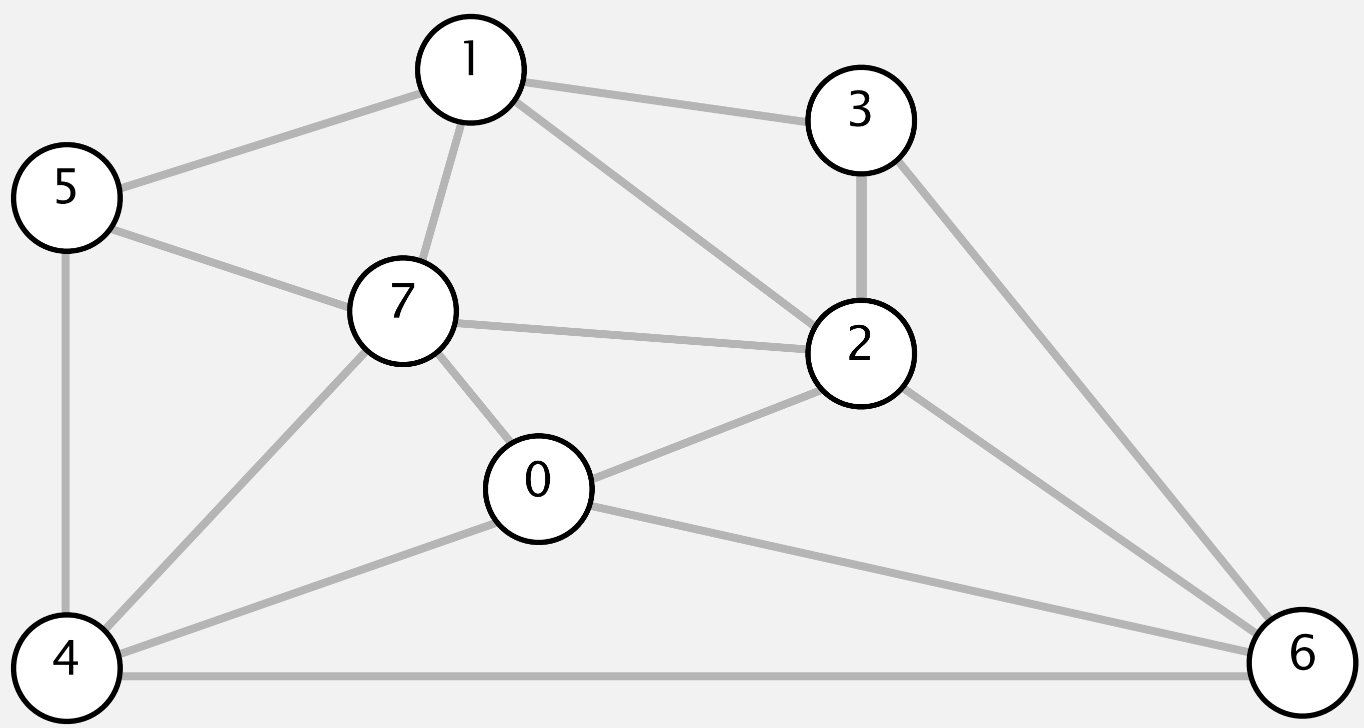
```
% java MST tinyEWG.txt
0-7 0.16
1-7 0.19
0-2 0.26
2-3 0.17
5-7 0.28
4-5 0.35
6-2 0.40
1.81
```

KRUSKAL'S ALGORITHM

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

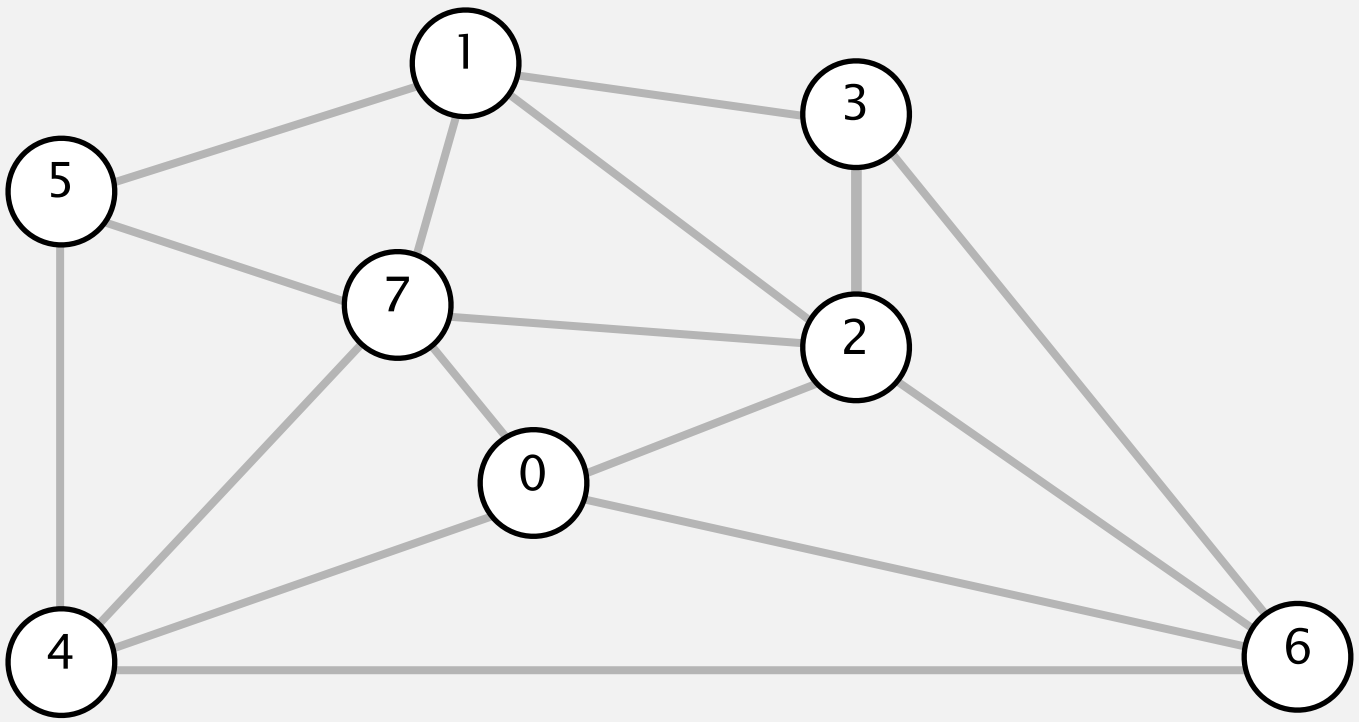
graph edges sorted by weight

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.



an edge-weighted graph

graph edges sorted by weight

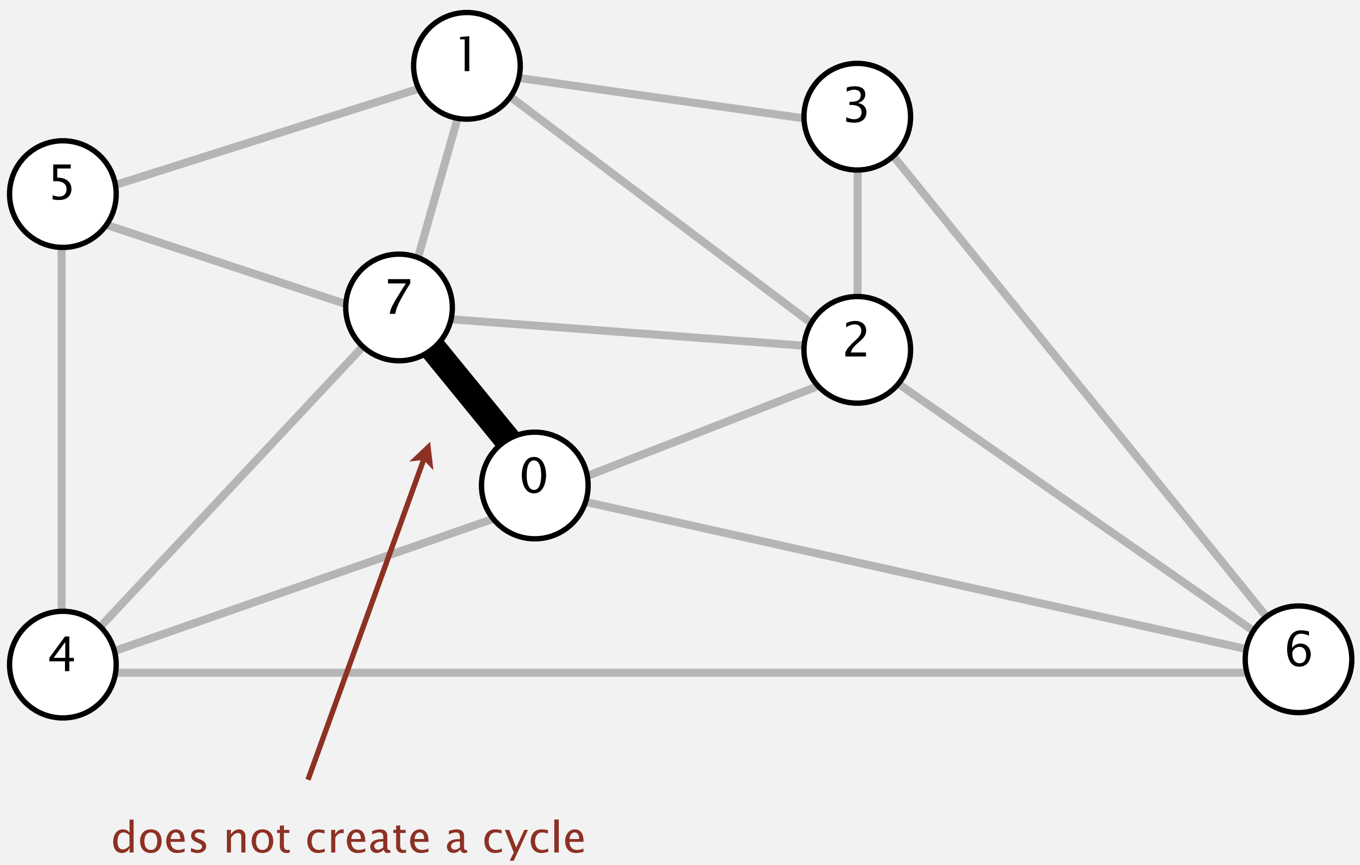
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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

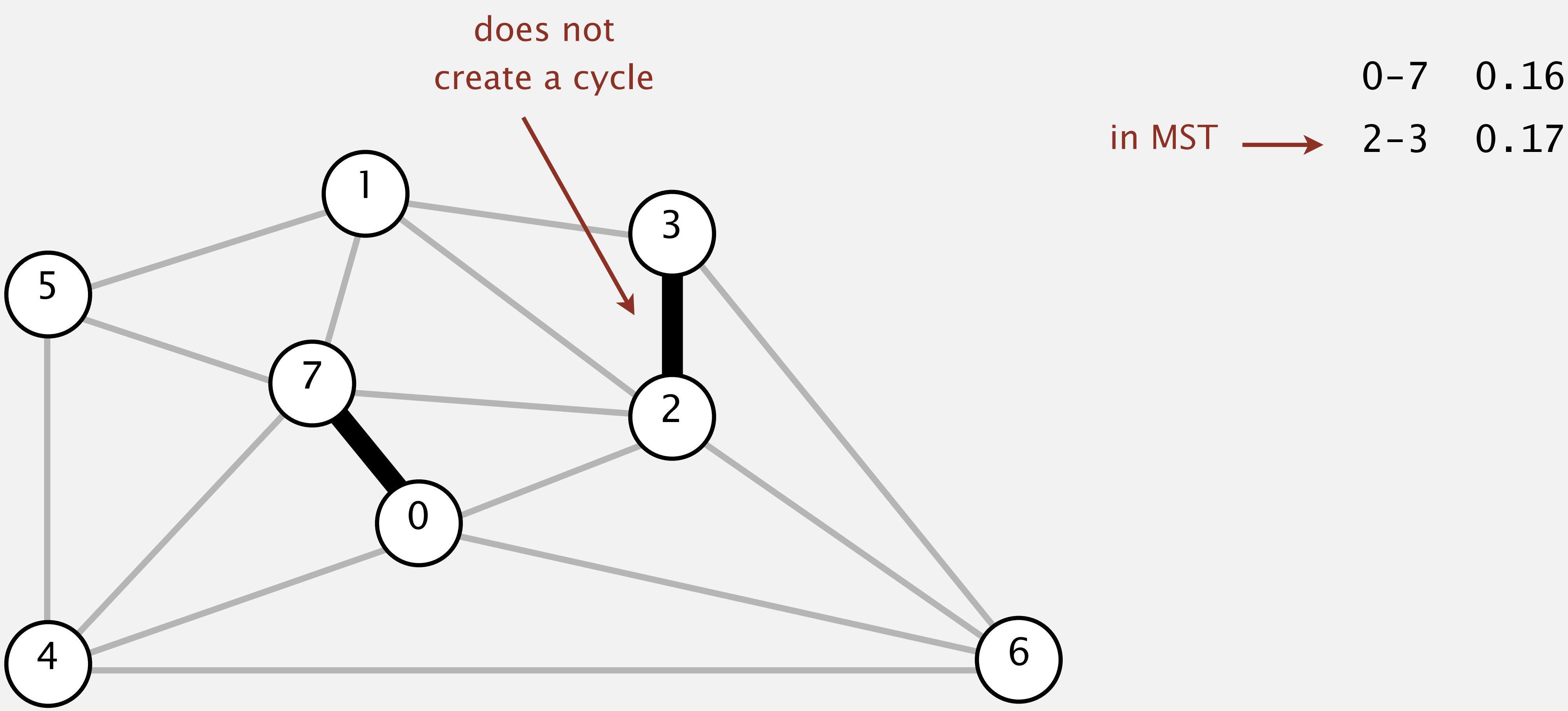
in MST → `0-7 0.16`



Kruskal's algorithm demo

Consider edges in ascending order of weight.

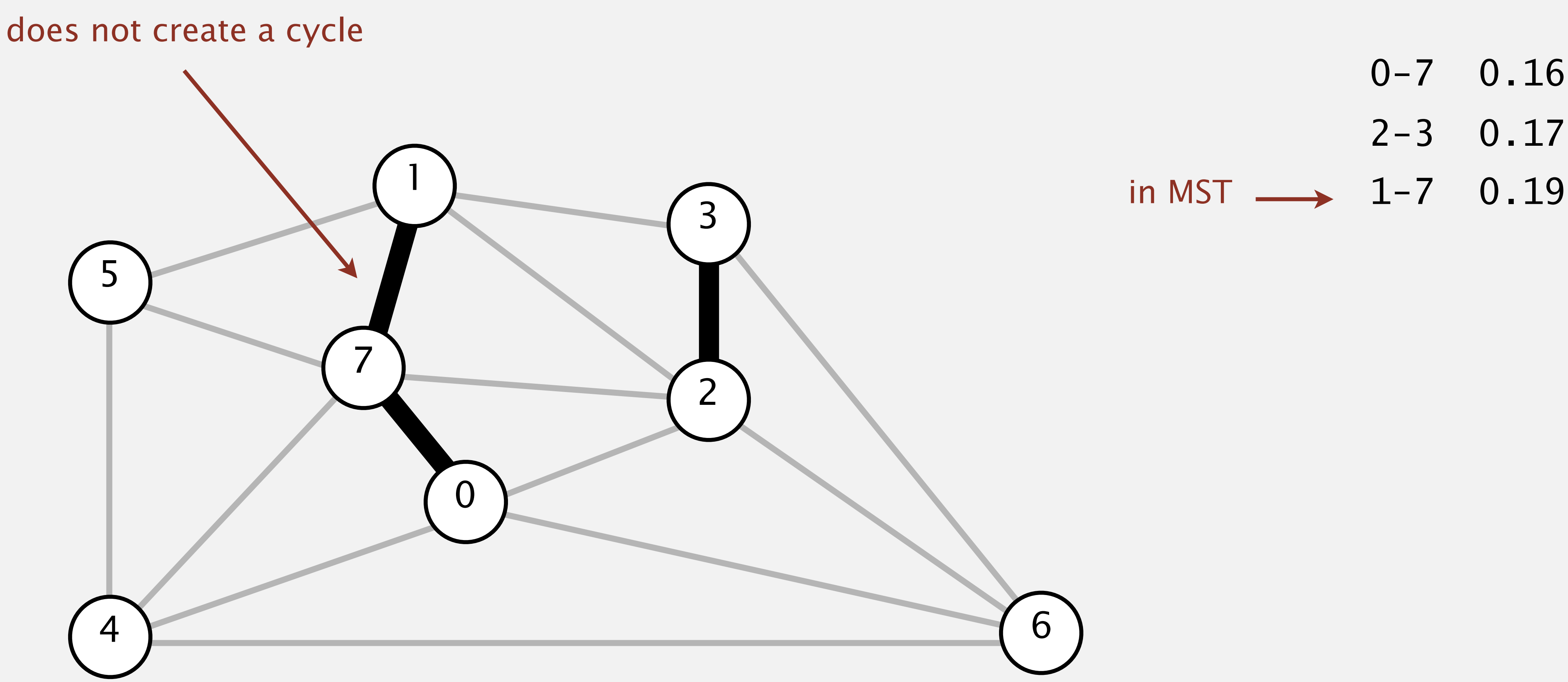
- Add next edge to tree T unless doing so would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight.

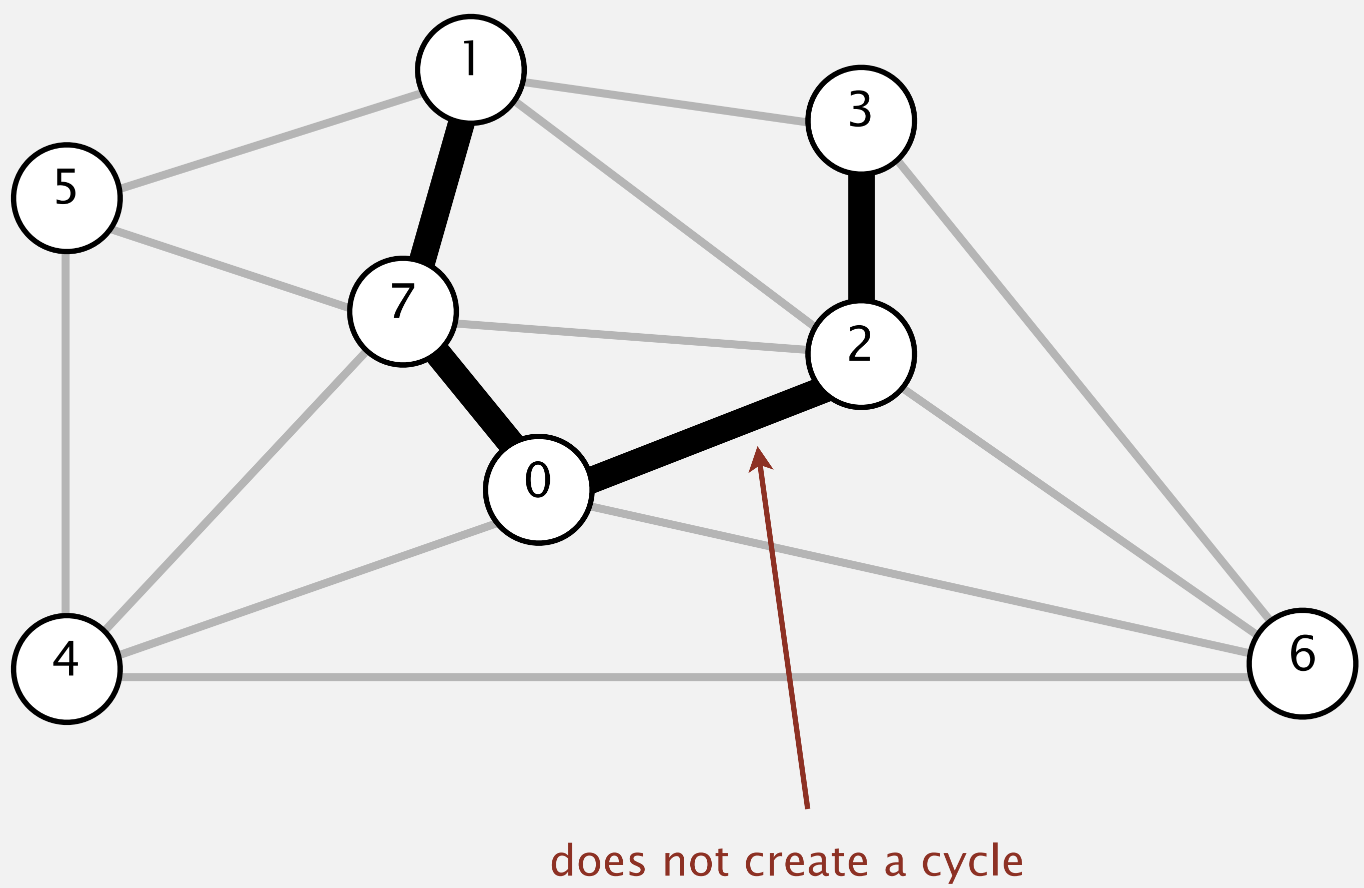
- Add next edge to tree T unless doing so would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight.

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

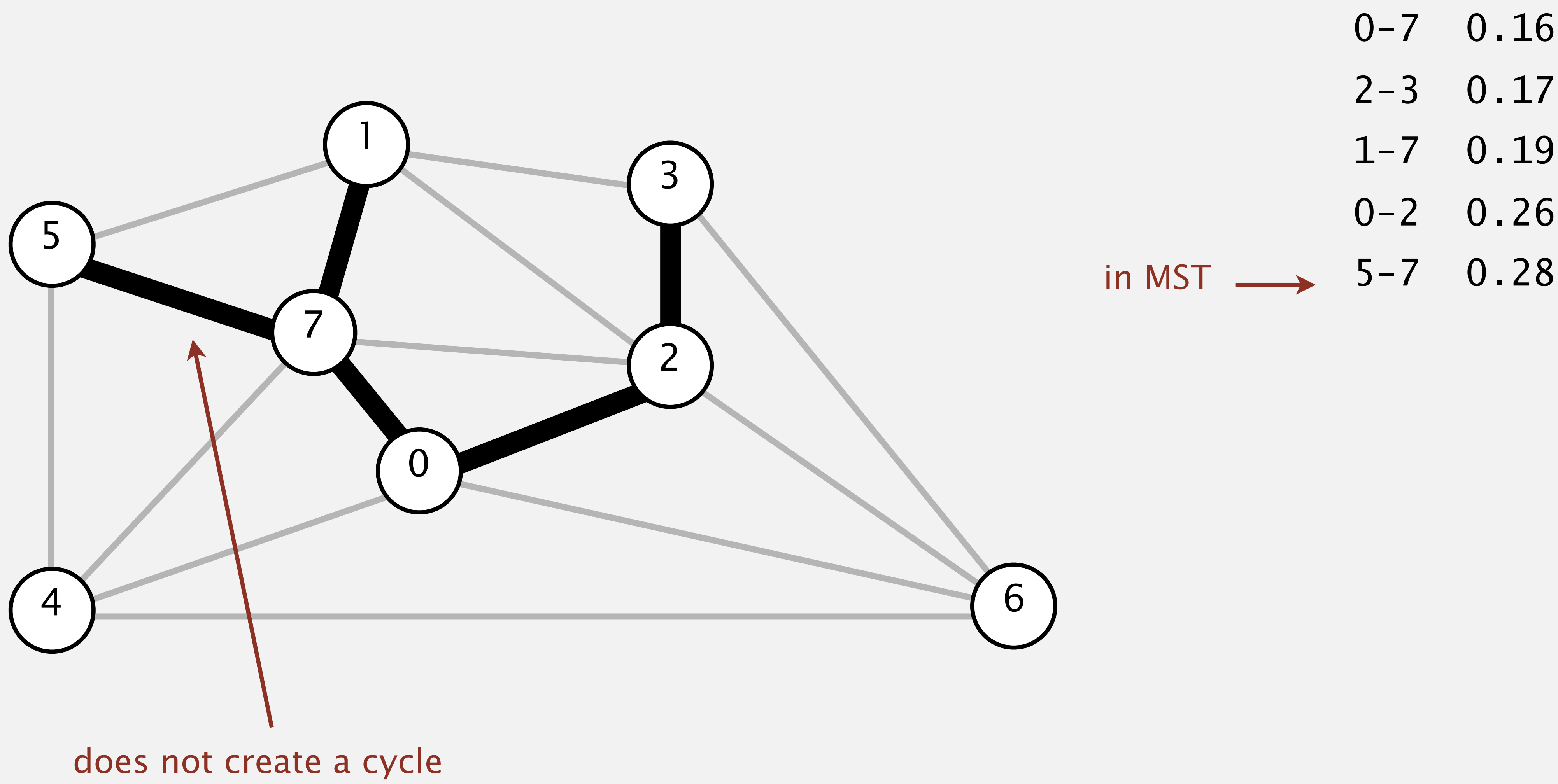


	0-7	0.16
	2-3	0.17
	1-7	0.19
in MST →	0-2	0.26

Kruskal's algorithm demo

Consider edges in ascending order of weight.

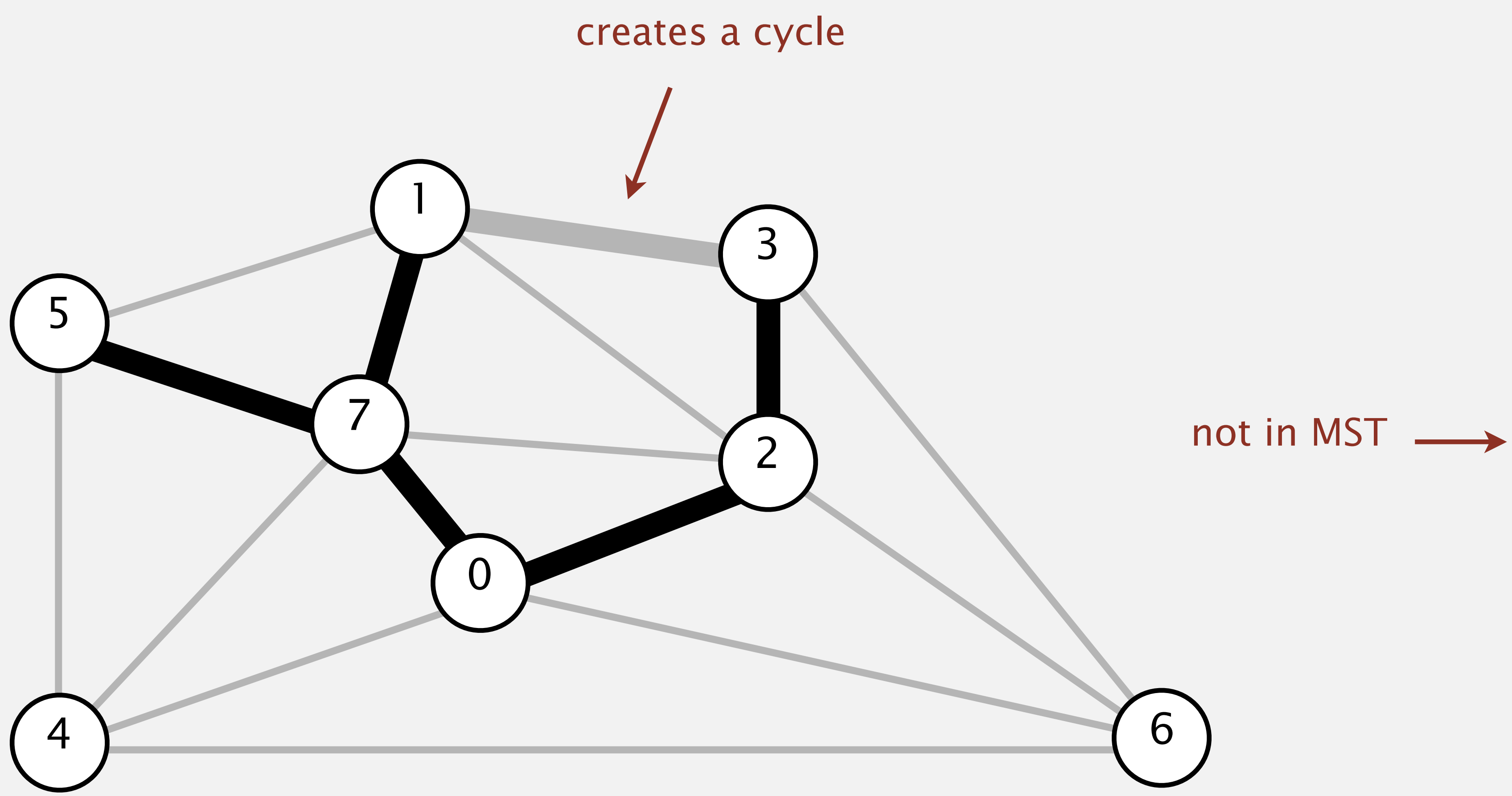
- Add next edge to tree T unless doing so would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight.

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

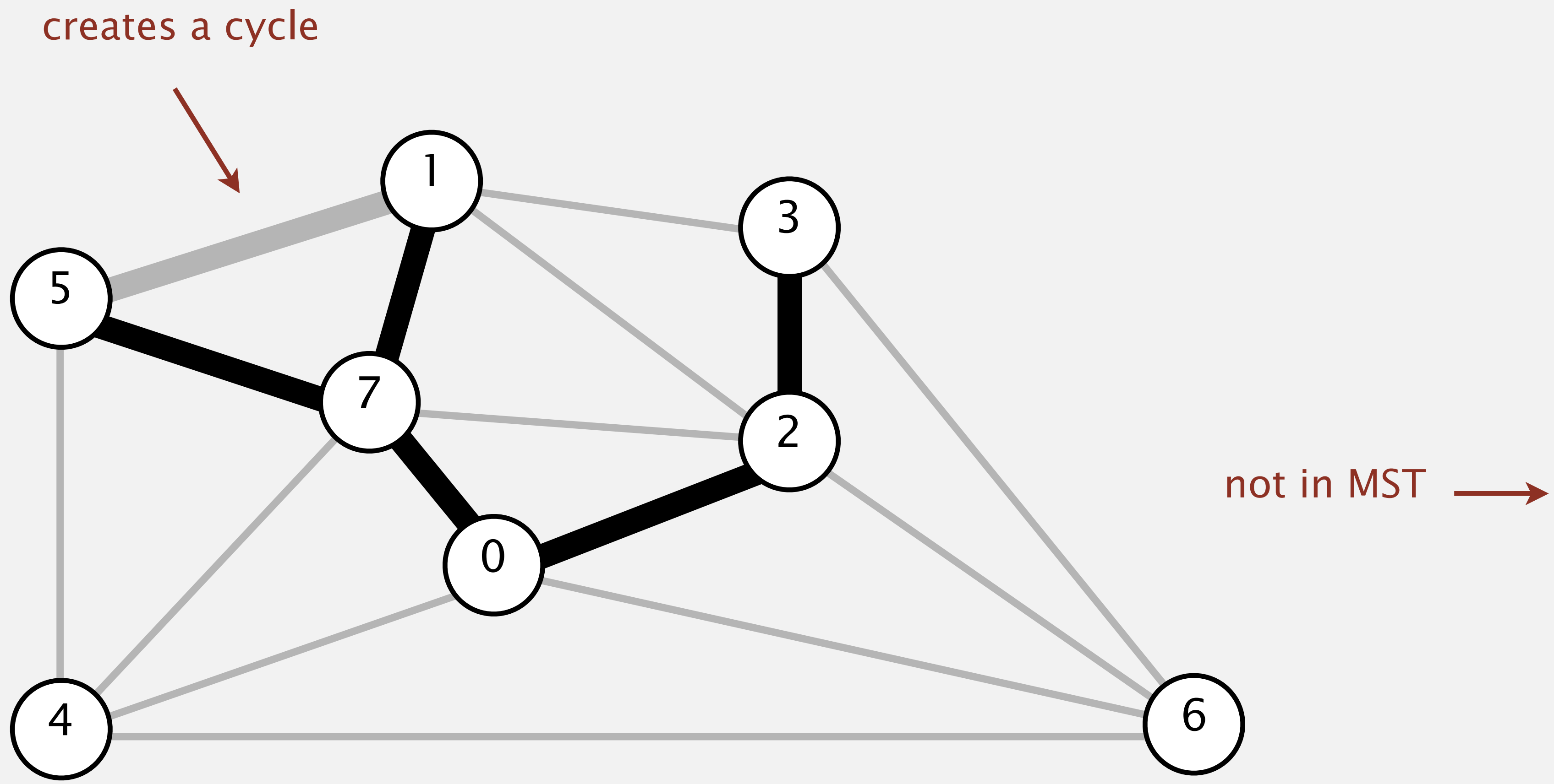


0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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

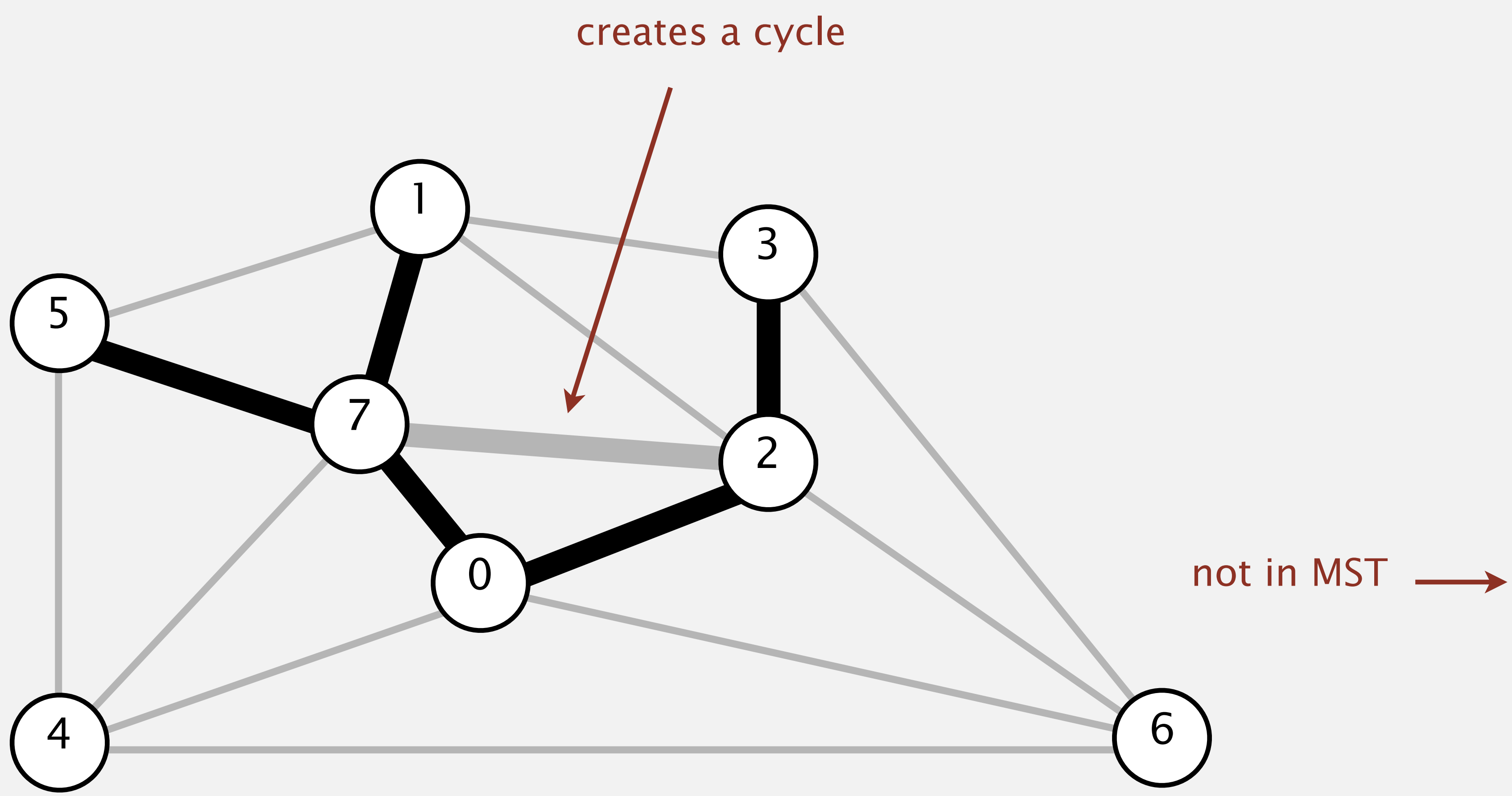


0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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

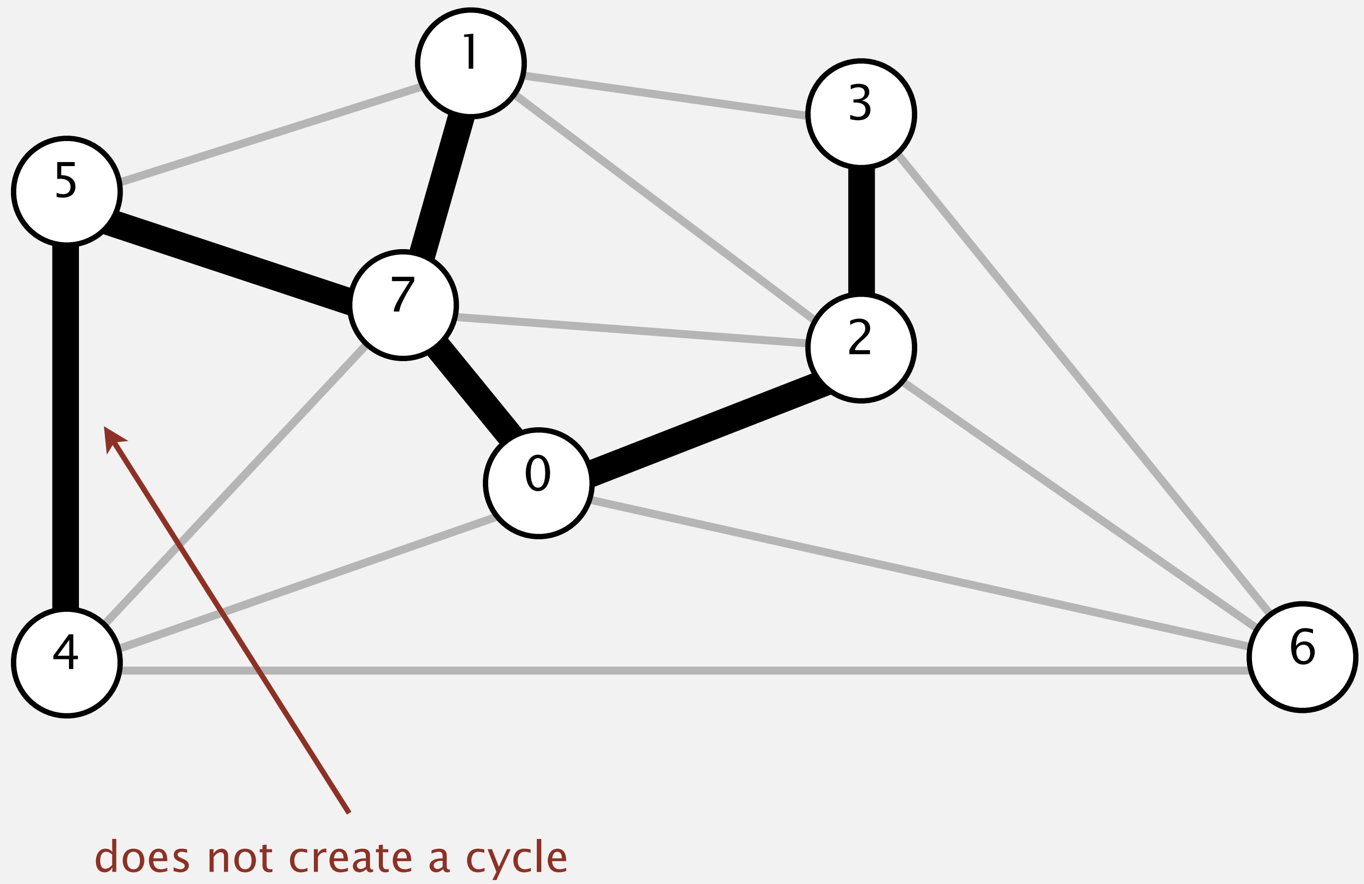


0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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



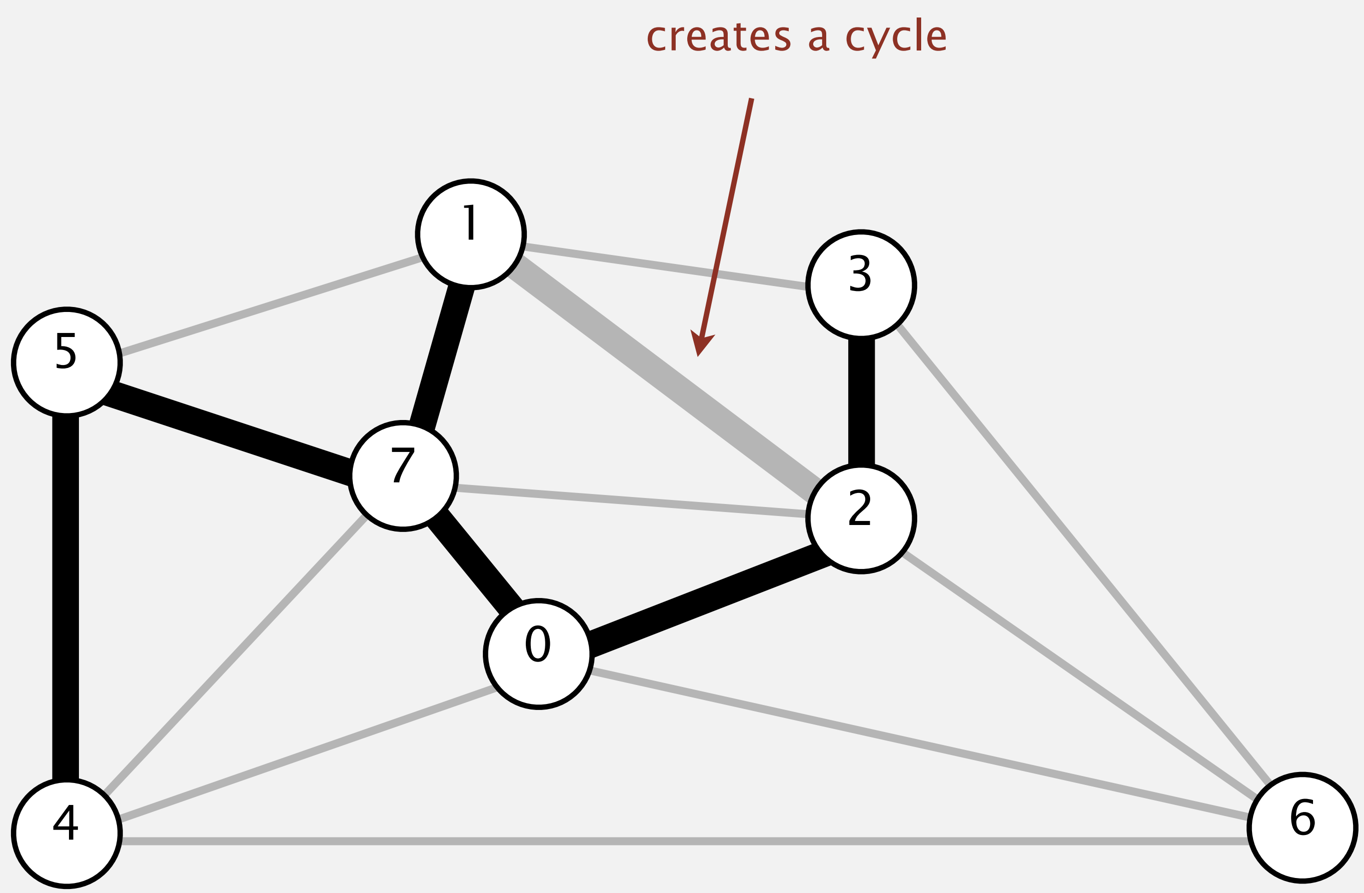
in MST →

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

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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



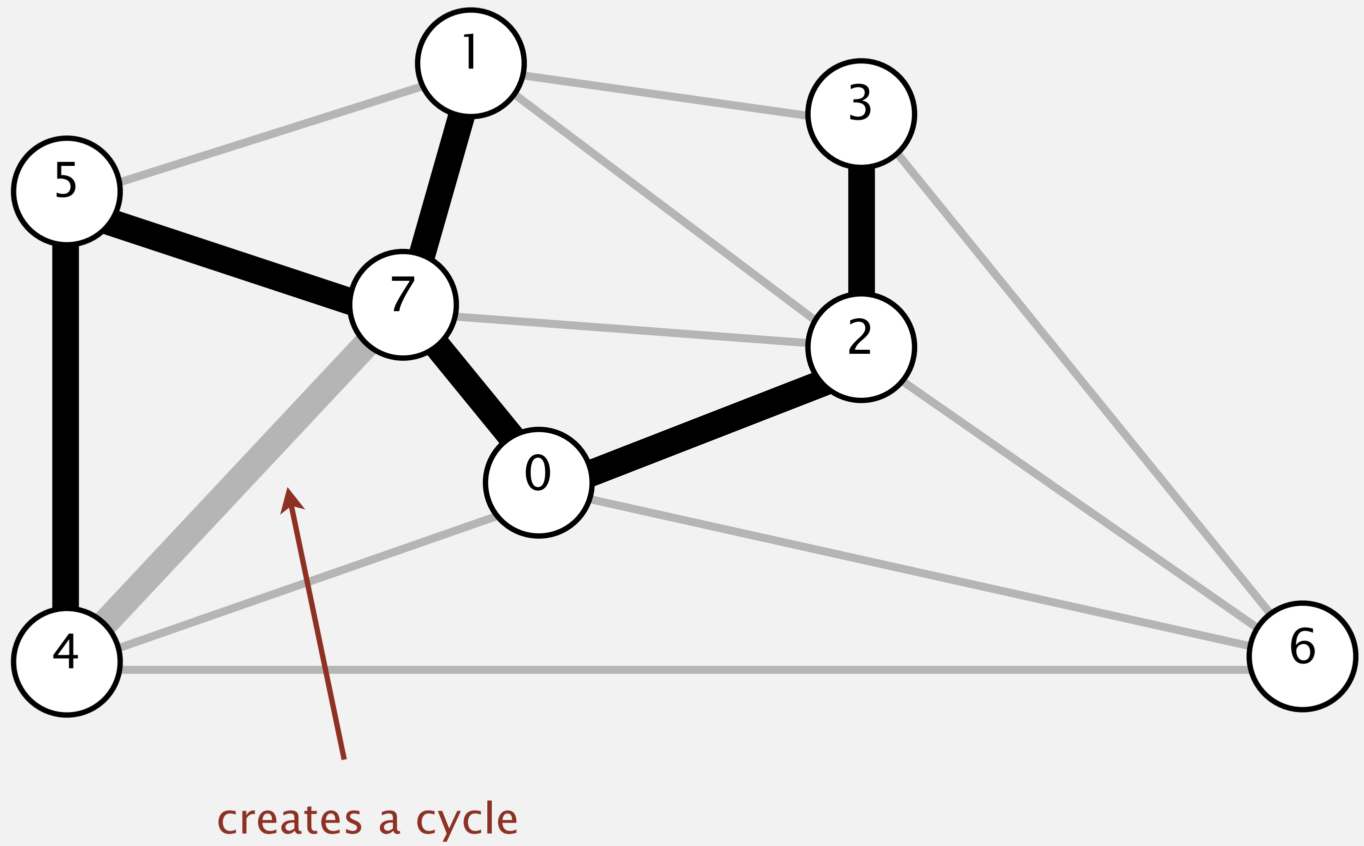
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36

not in MST

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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



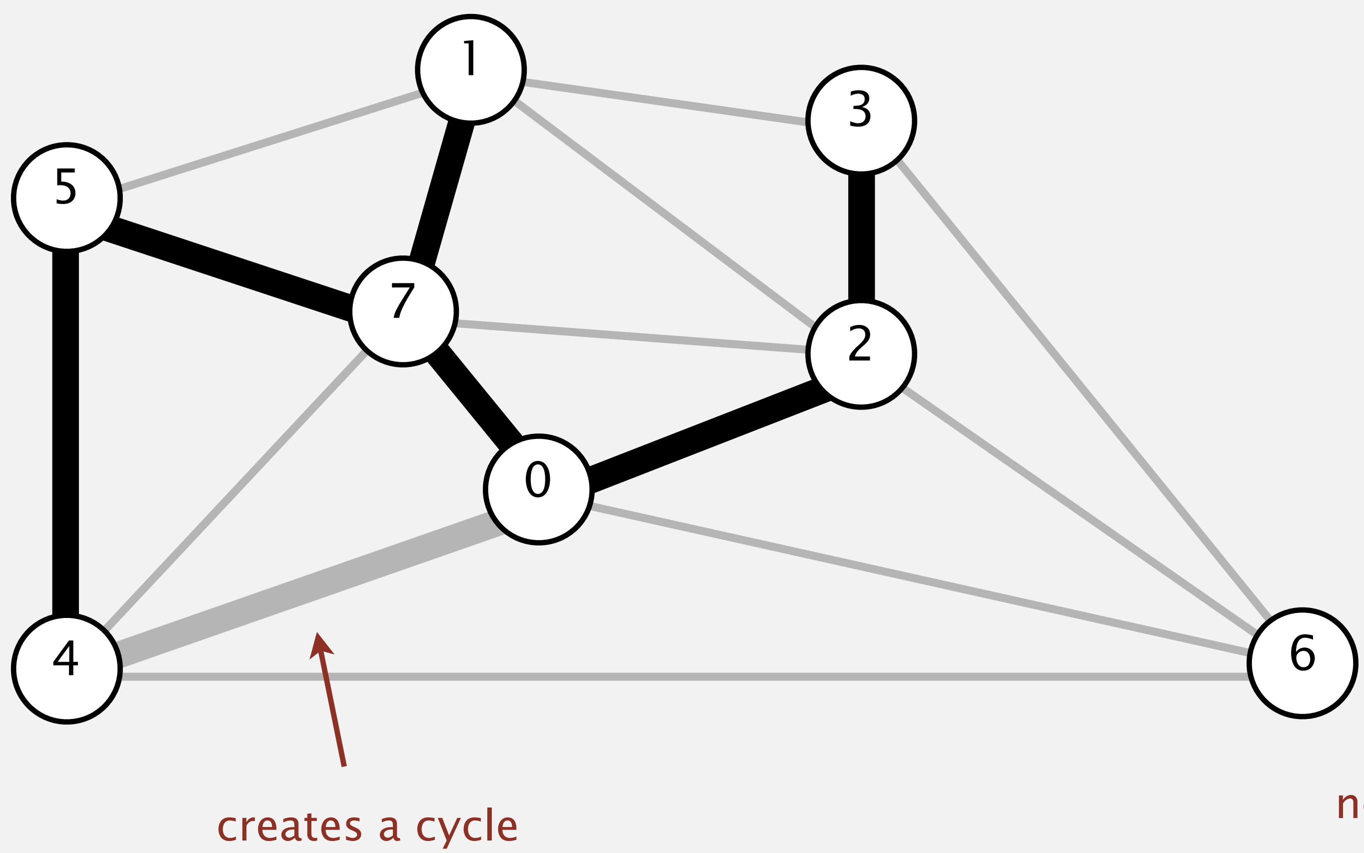
not in MST

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

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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

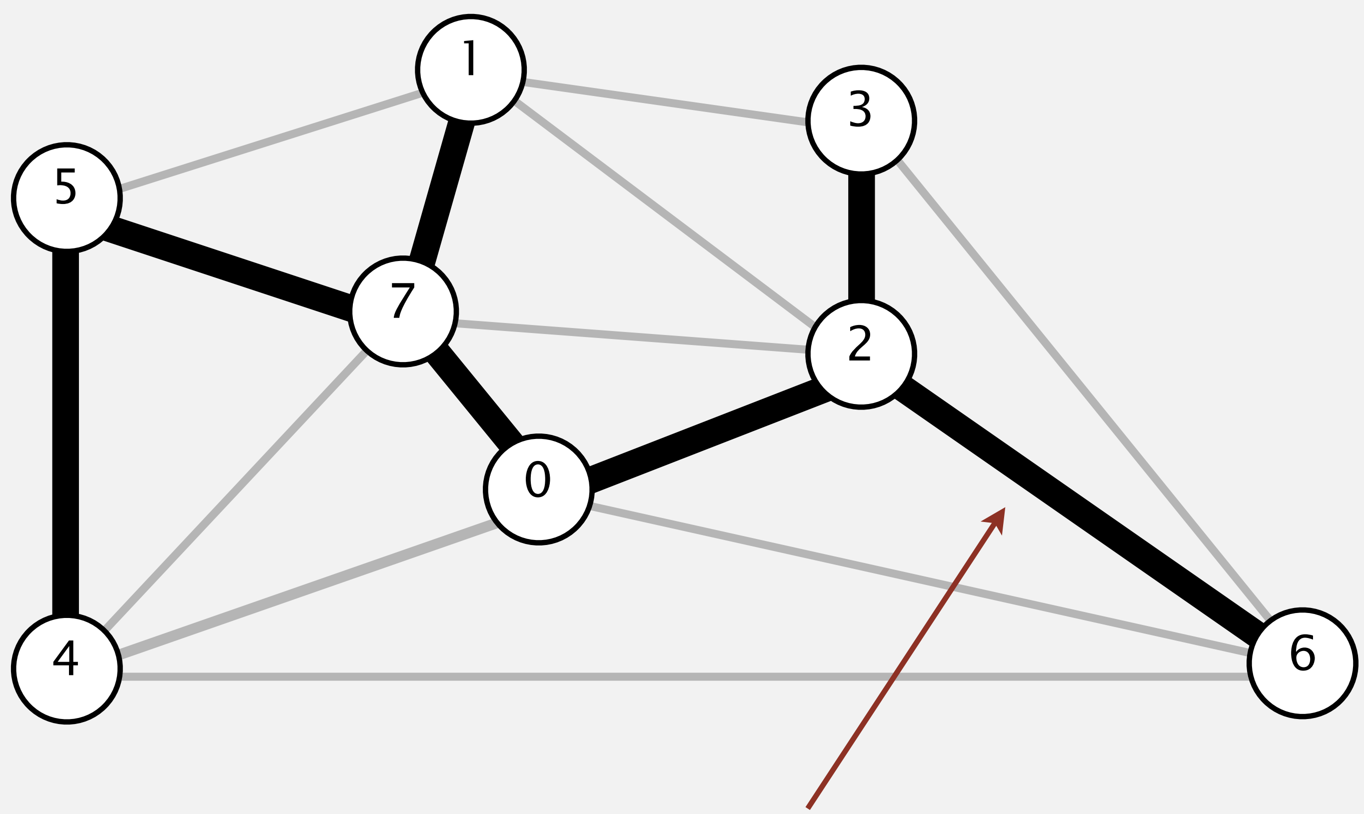


0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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



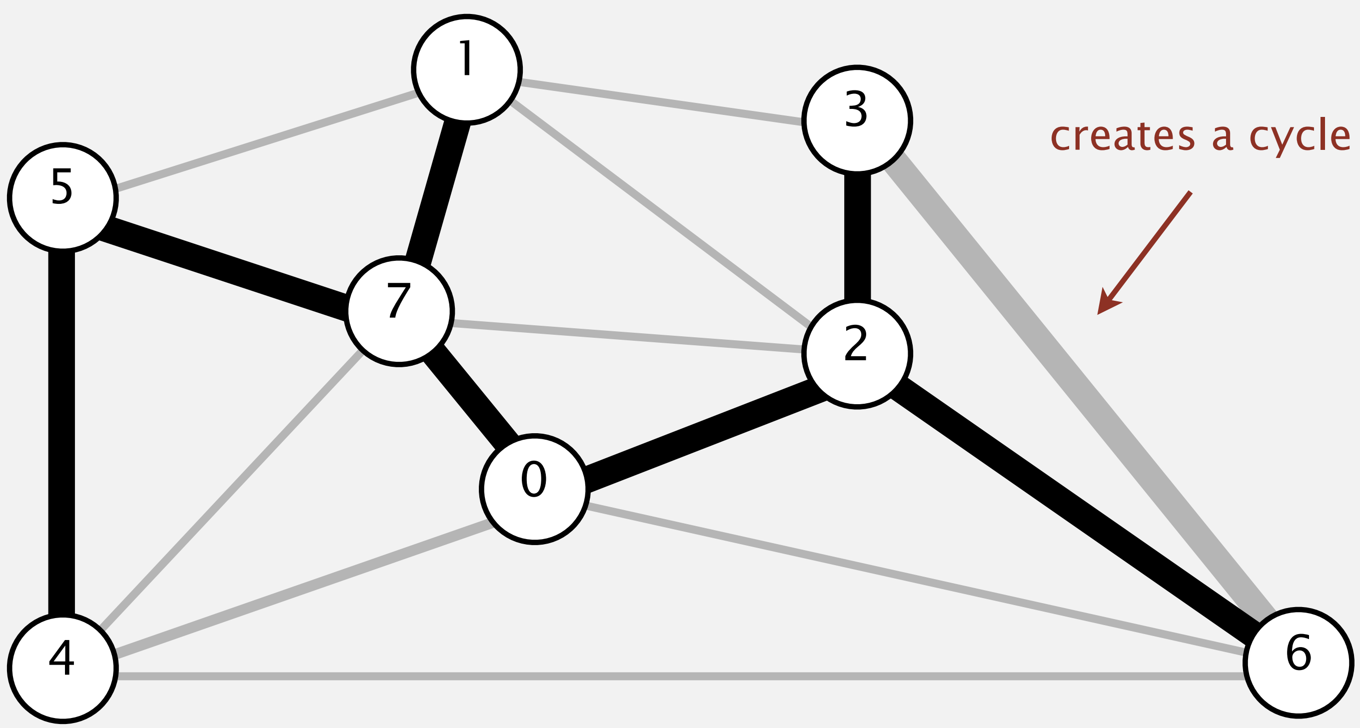
in MST →

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

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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



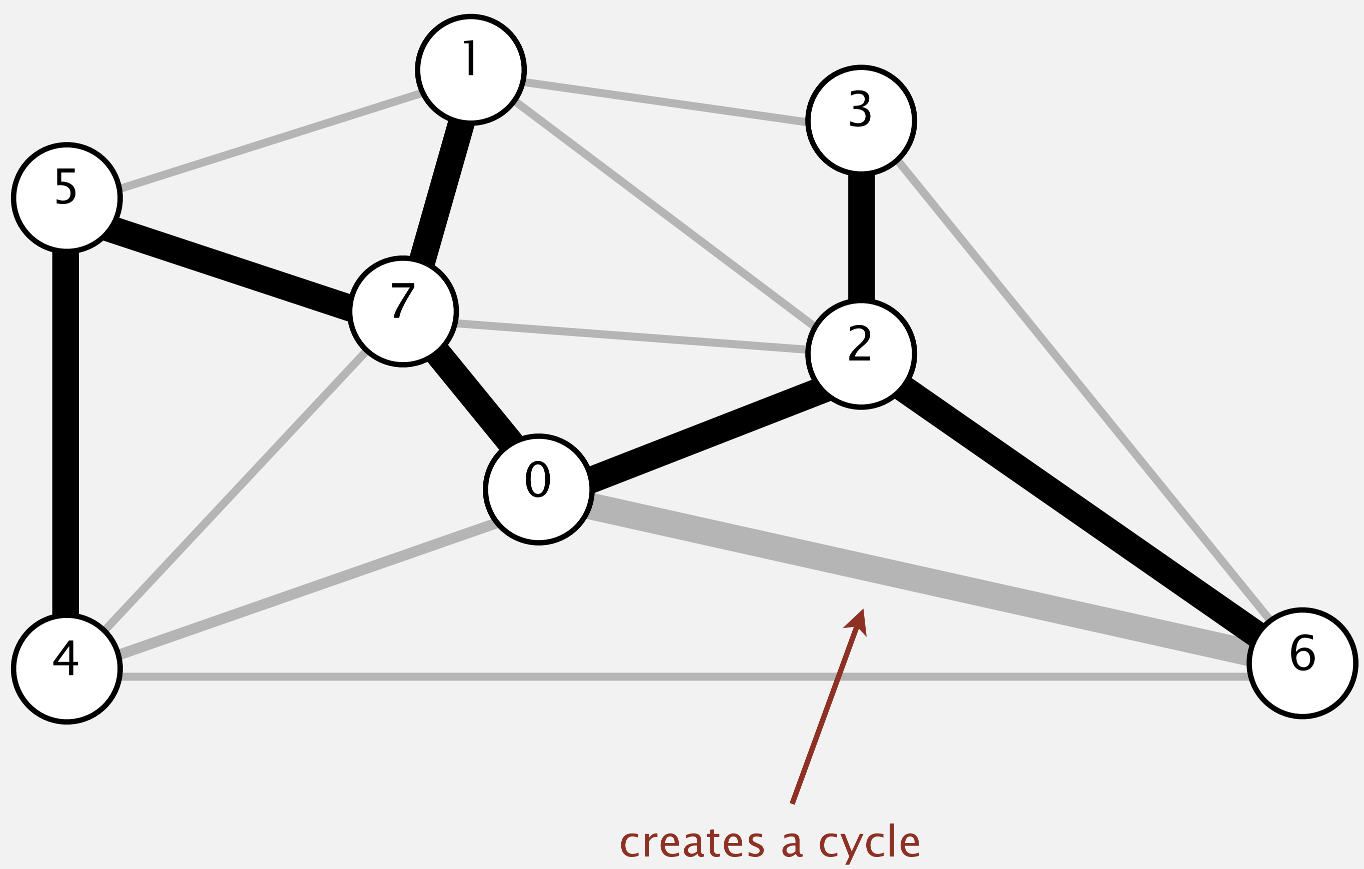
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52

not in MST

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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



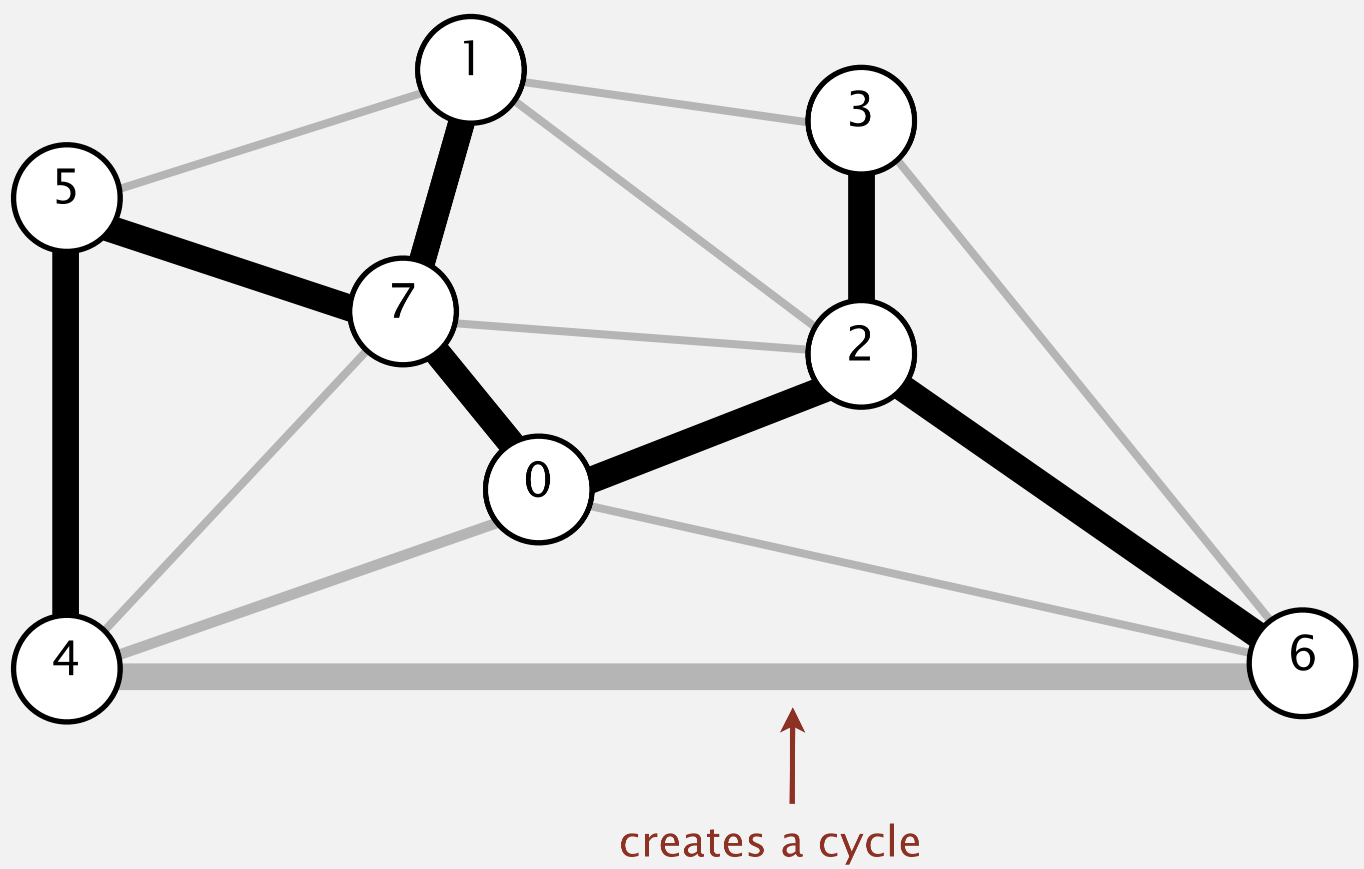
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

not in MST →

Kruskal's algorithm demo

Consider edges in ascending order of weight.

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



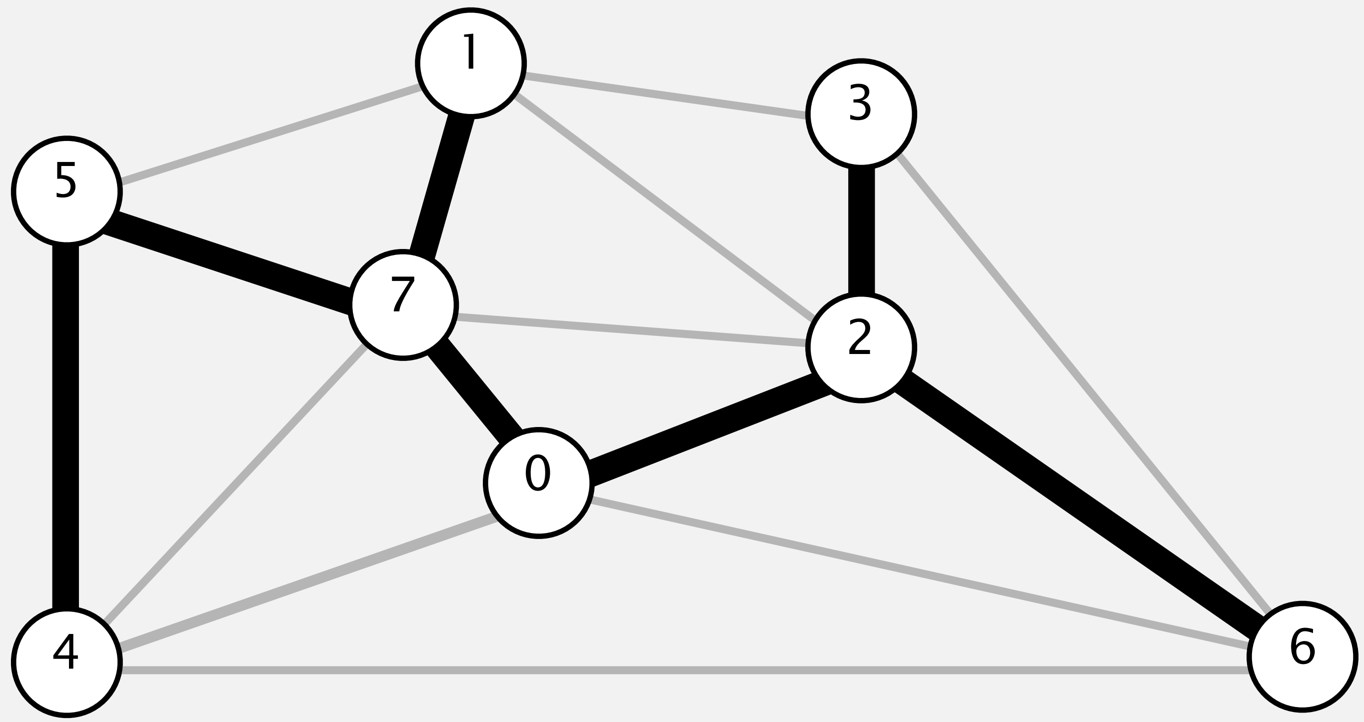
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

not in MST →

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

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

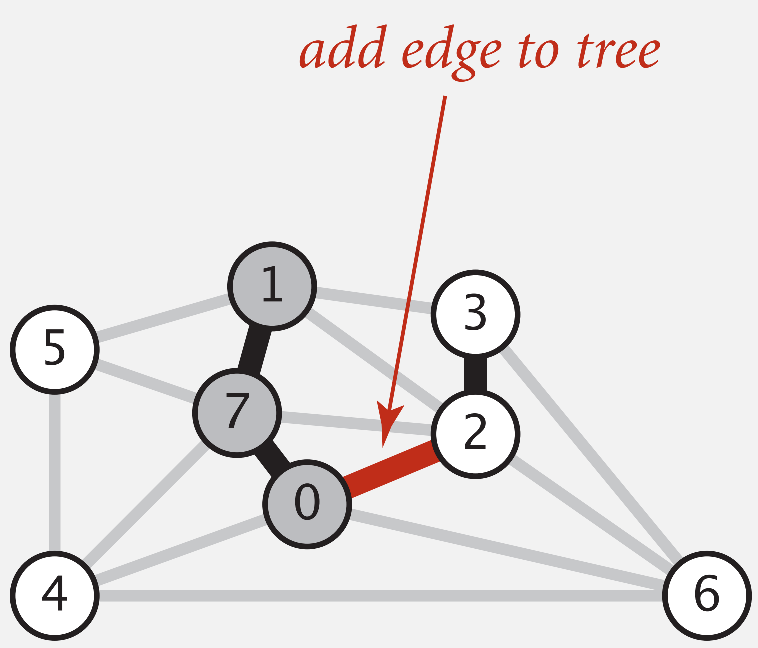
Kruskal's algorithm: visualization

Kruskal's algorithm: correctness proof

Proposition. [Kruskal 1956] Kruskal's algorithm computes the MST.

Pf. Kruskal's algorithm is a special case of the greedy MST algorithm.

- Suppose Kruskal's algorithm colors the edge $e = v–w$ black.
- Cut = set of vertices connected to v in tree T.
- No crossing edge is black.
- No crossing edge has lower weight. Why?

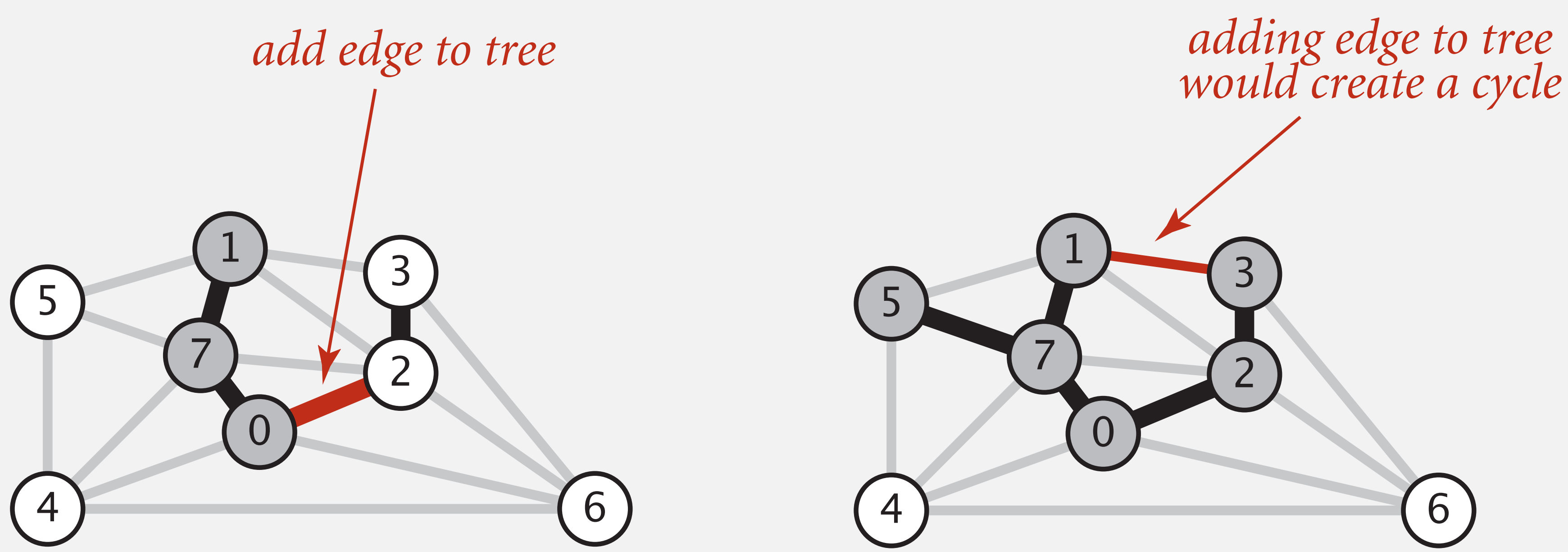


Kruskal's algorithm: implementation challenge

Challenge. Would adding edge $v\text{–}w$ to tree T create a cycle? If not, add it.

How difficult?

- $E + V$
- V ← run DFS from v, check if w is reachable (T has at most V – 1 edges)
- $\log V$
- $\log^* V$ ← use the union-find data structure !
- 1

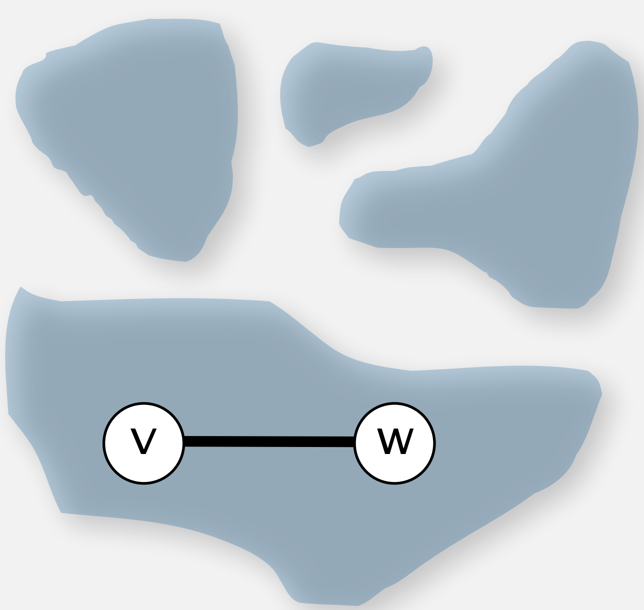


Kruskal's algorithm: implementation challenge

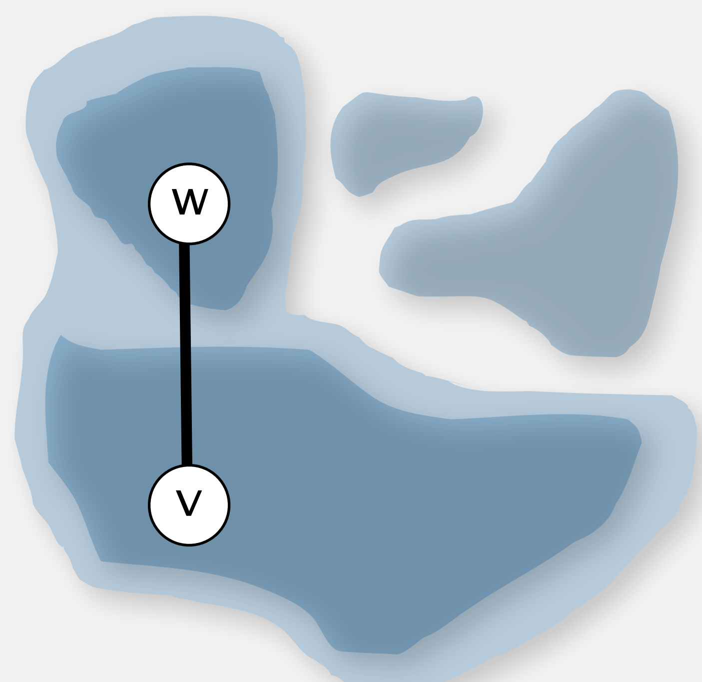
Challenge. Would adding edge $v–w$ to tree T create a cycle? If not, add it.

Efficient solution. Use the union-find data structure.

- Maintain a set for each connected component in T.
- If v and w are in same set, then adding $v–w$ would create a cycle.
- To add $v–w$ to T, merge sets containing v and w.



Case 1: adding v–w creates a cycle



Case 2: add v–w to T and merge sets containing v and w

Kruskal's algorithm: Java implementation

```
public class KruskalMST
{
   private Queue<Edge> mst = new Queue<Edge>();

   public KruskalMST(EdgeWeightedGraph G)
   {
      MinPQ<Edge> pq = new MinPQ<Edge>(G.edges());

      UF uf = new UF(G.V());
      while (!pq.isEmpty() && mst.size() < G.V()-1)
      {
         Edge e = pq.delMin();
         int v = e.either(), w = e.other(v);
         if (!uf.connected(v, w))
         {
            uf.union(v, w);
            mst.enqueue(e);
         }
      }
   }

   public Iterable<Edge> edges()
   {  return mst;  }
}
```

build priority queue (or sort)

greedily add edges to MST

edge v–w does not create cycle

merge sets

add edge to MST

Kruskal's algorithm: running time

Proposition. Kruskal's algorithm computes MST in time proportional to $E \log E$ (in the worst case).

Pf.

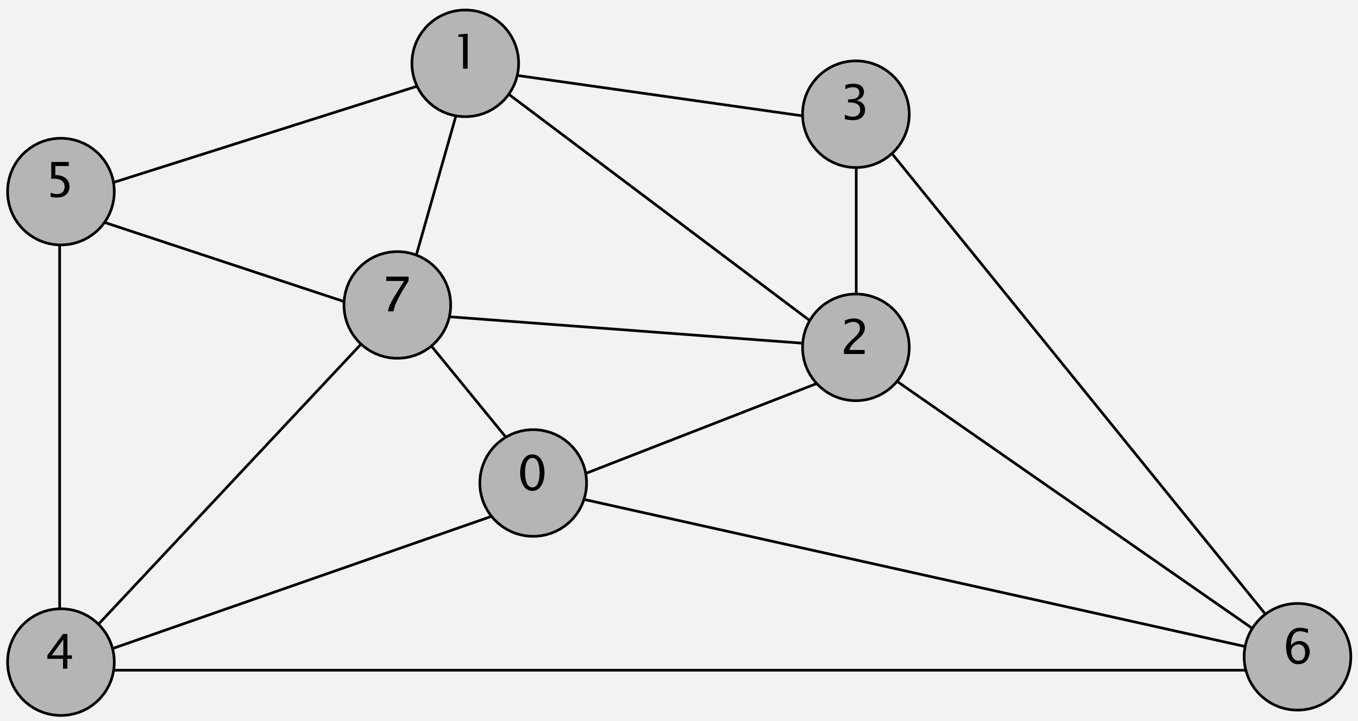
operation	frequency	time per op
build pq	1	E
delete-min	E	$\log E$
union	V	$\log^* V$ †
connected	E	$\log^* V$ †

† amortized bound using weighted quick union with path compression

PRIMS ALGORITHM

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.

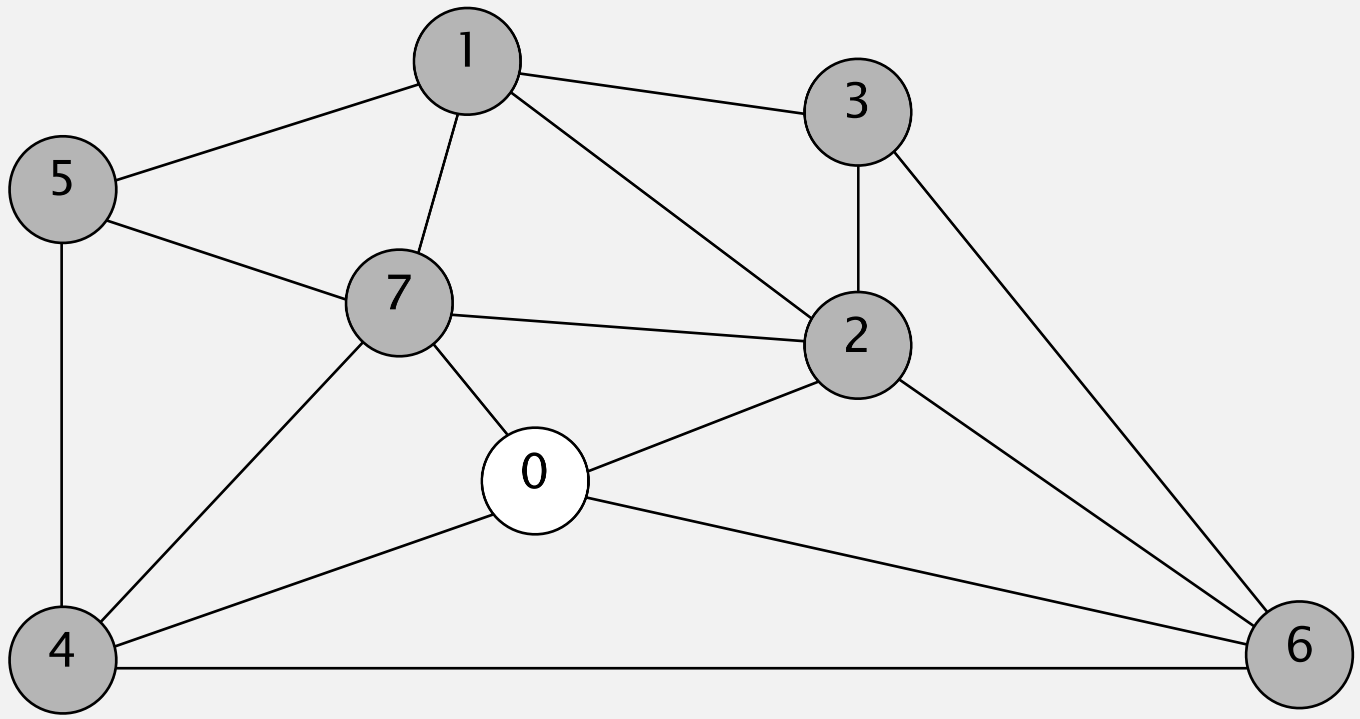


an edge-weighted graph

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

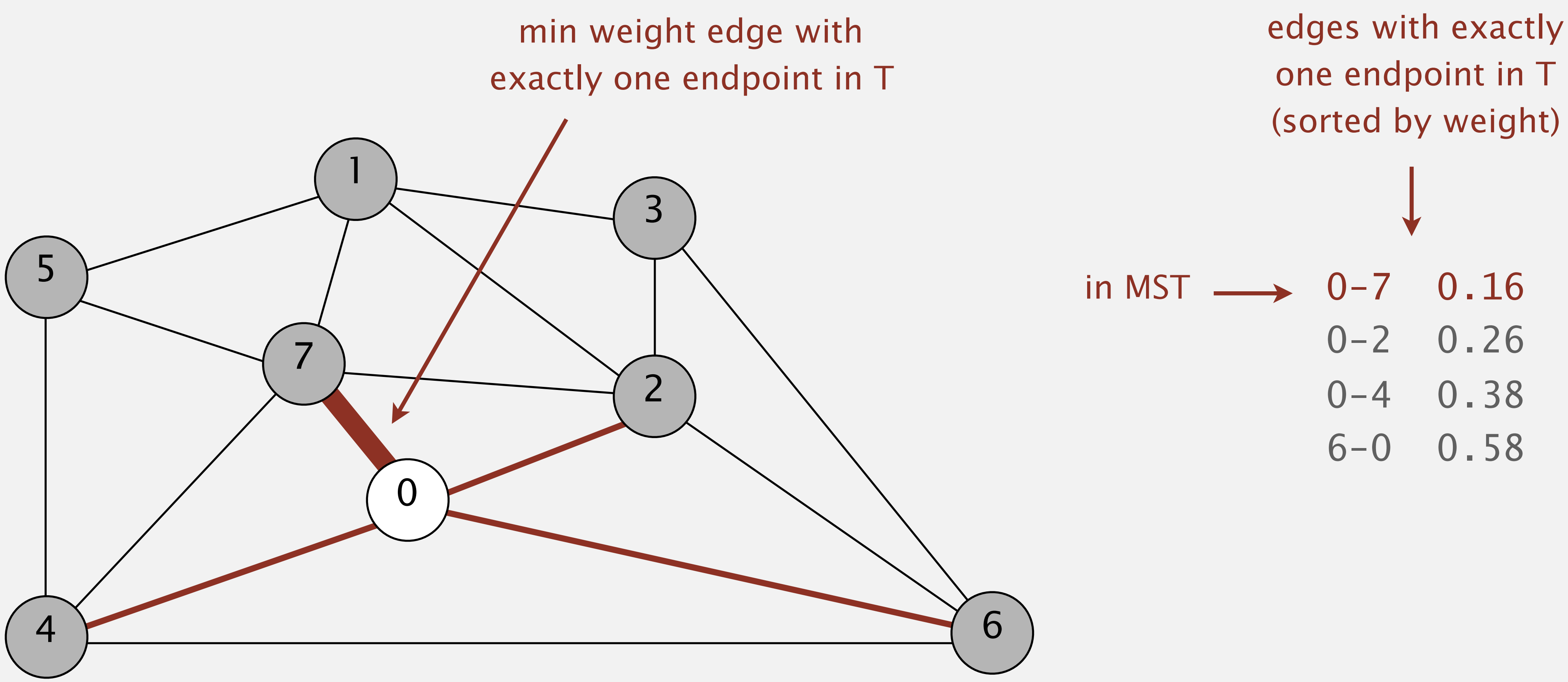
Prim's algorithm demo

- 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.



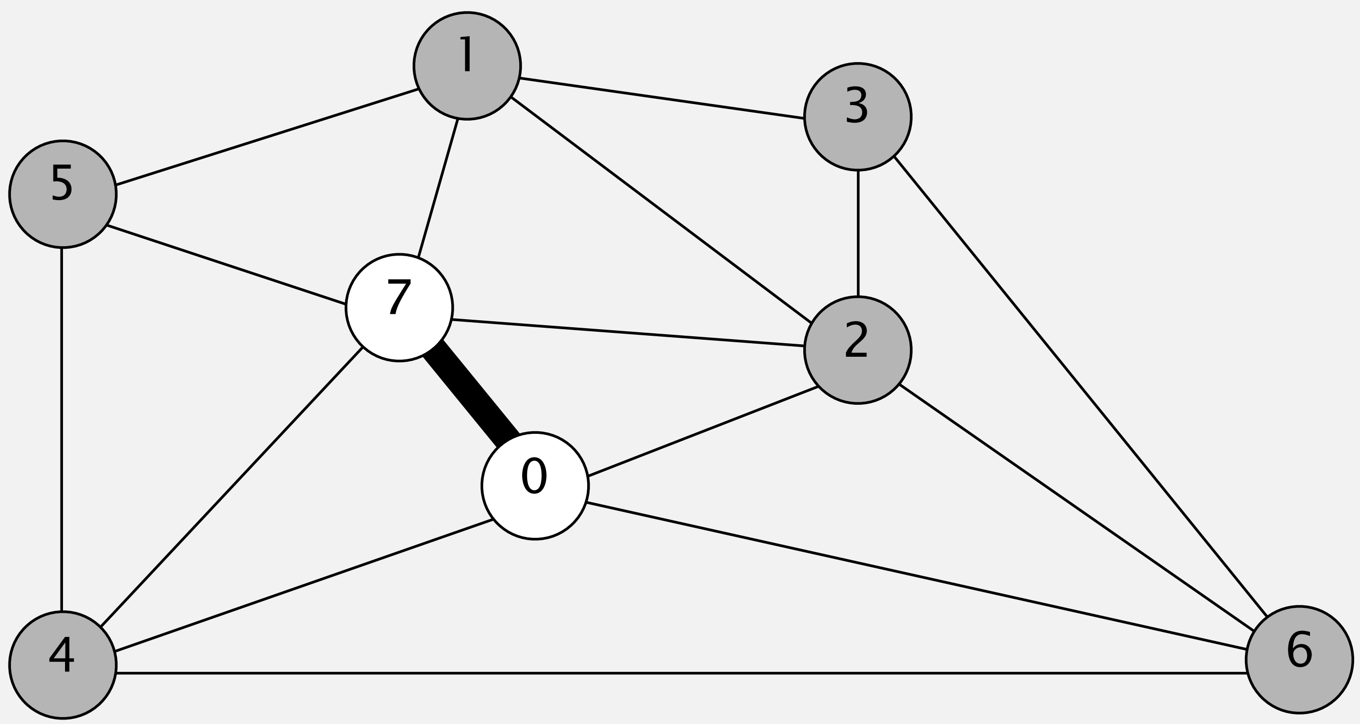
Prim's algorithm demo

- 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.



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.



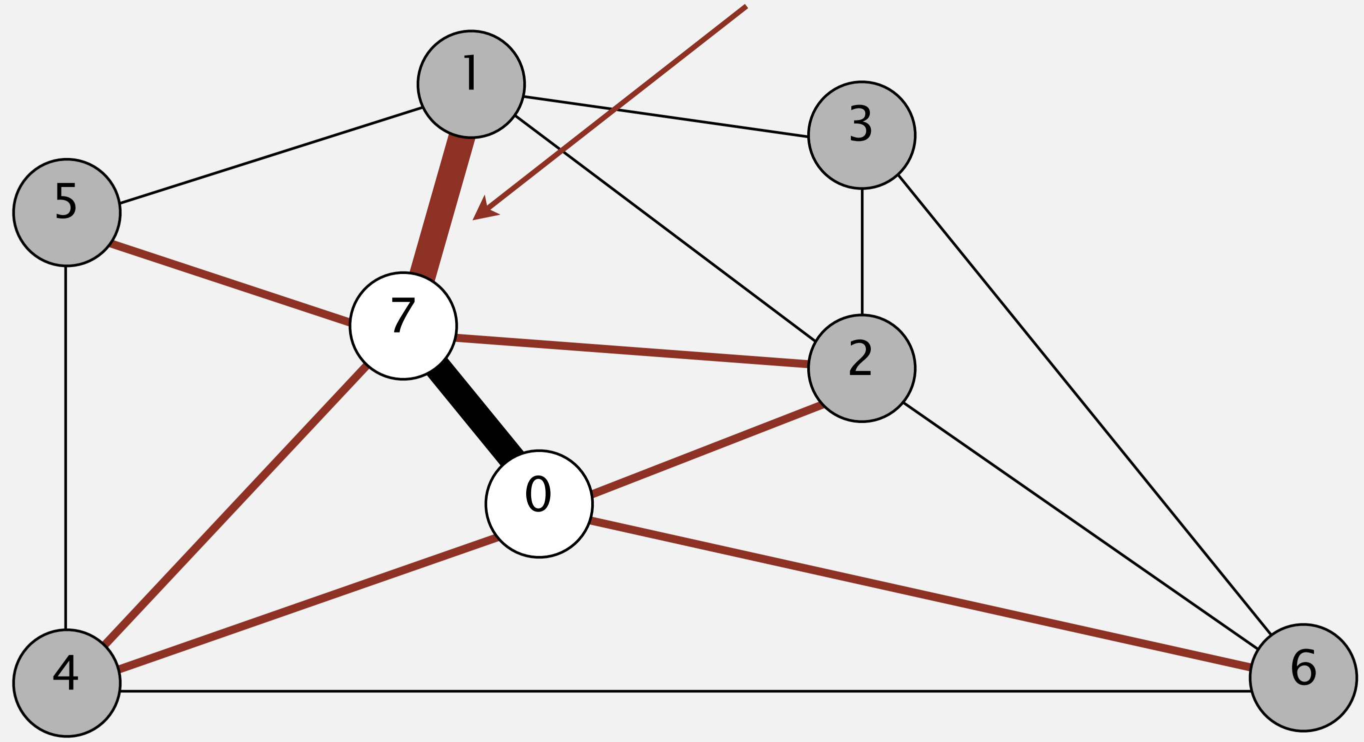
MST edges

0–7

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 →

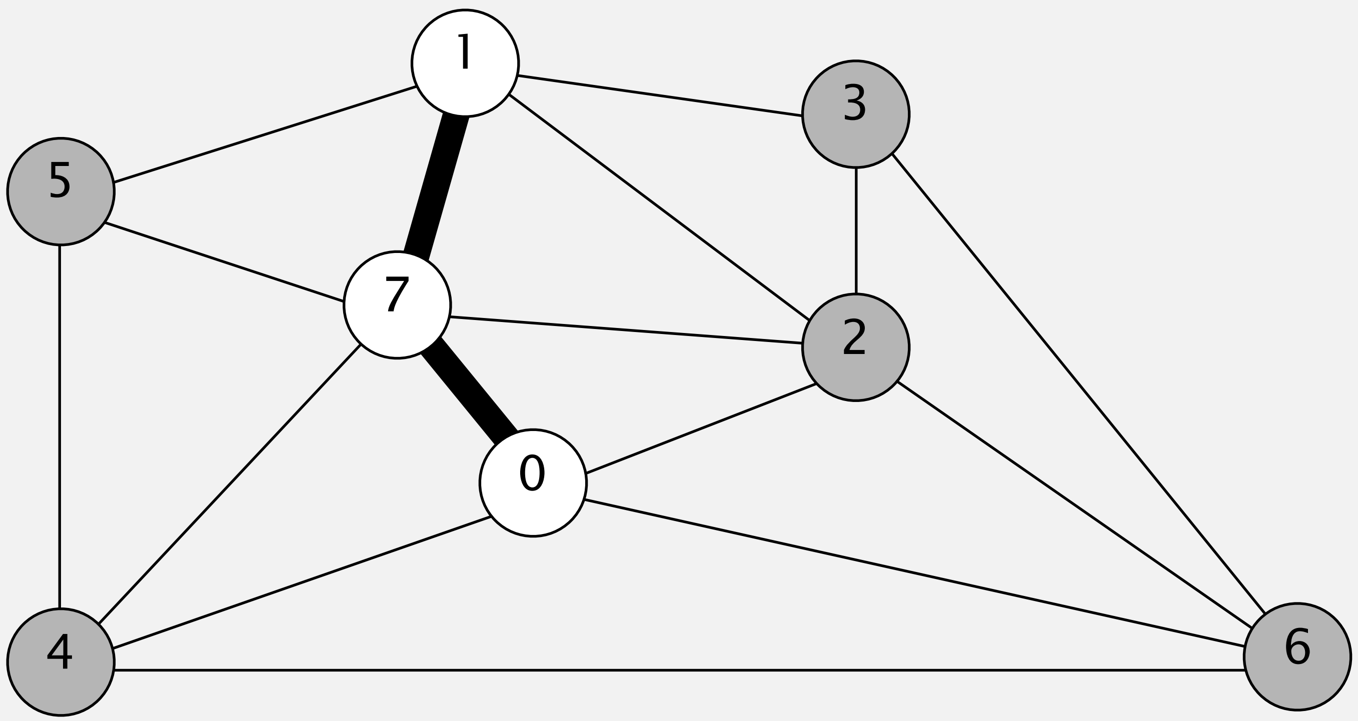
1-7	0.19
0-2	0.26
5-7	0.28
2-7	0.34
4-7	0.37
0-4	0.38
6-0	0.58

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.

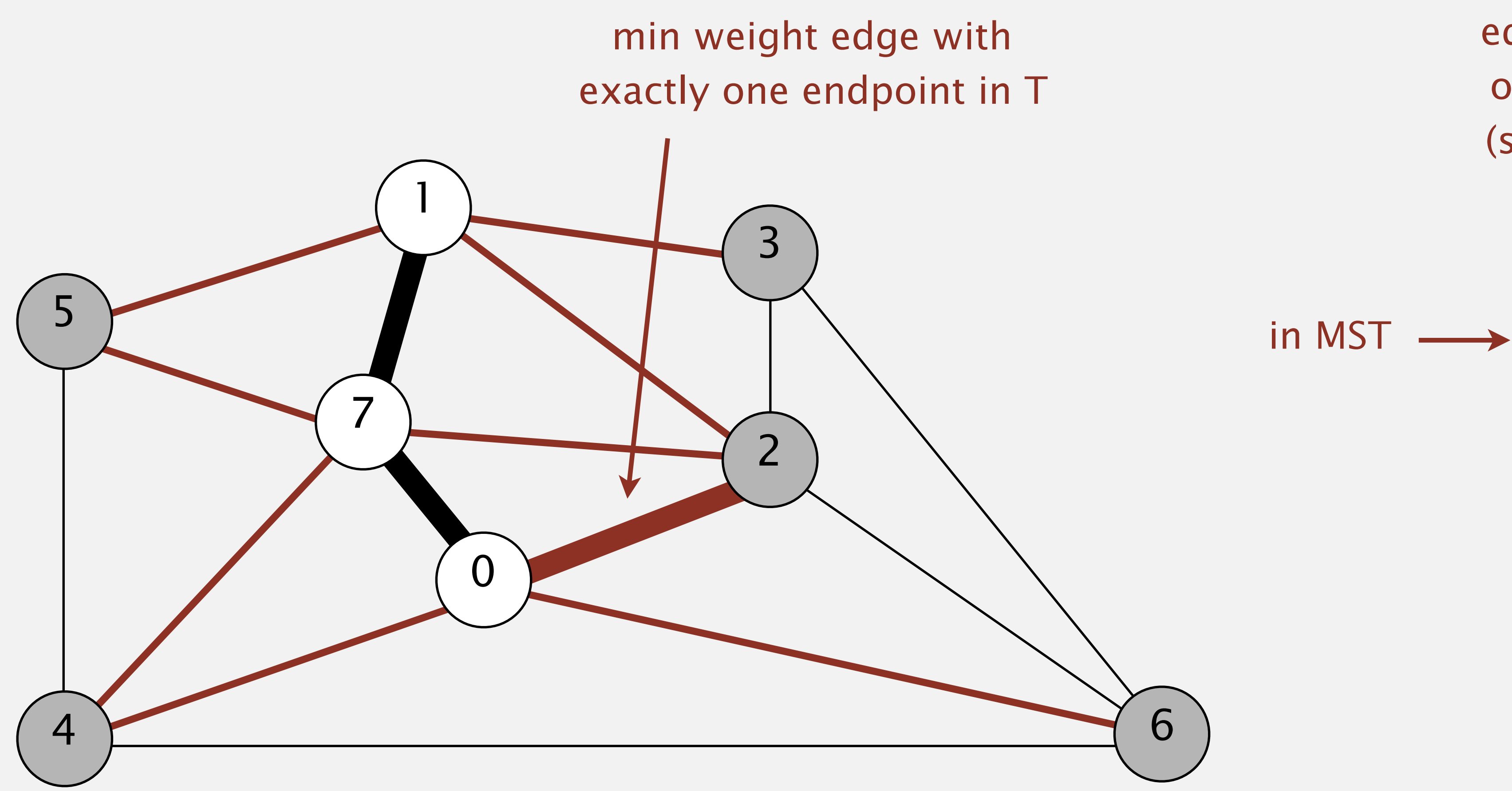


MST edges

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.



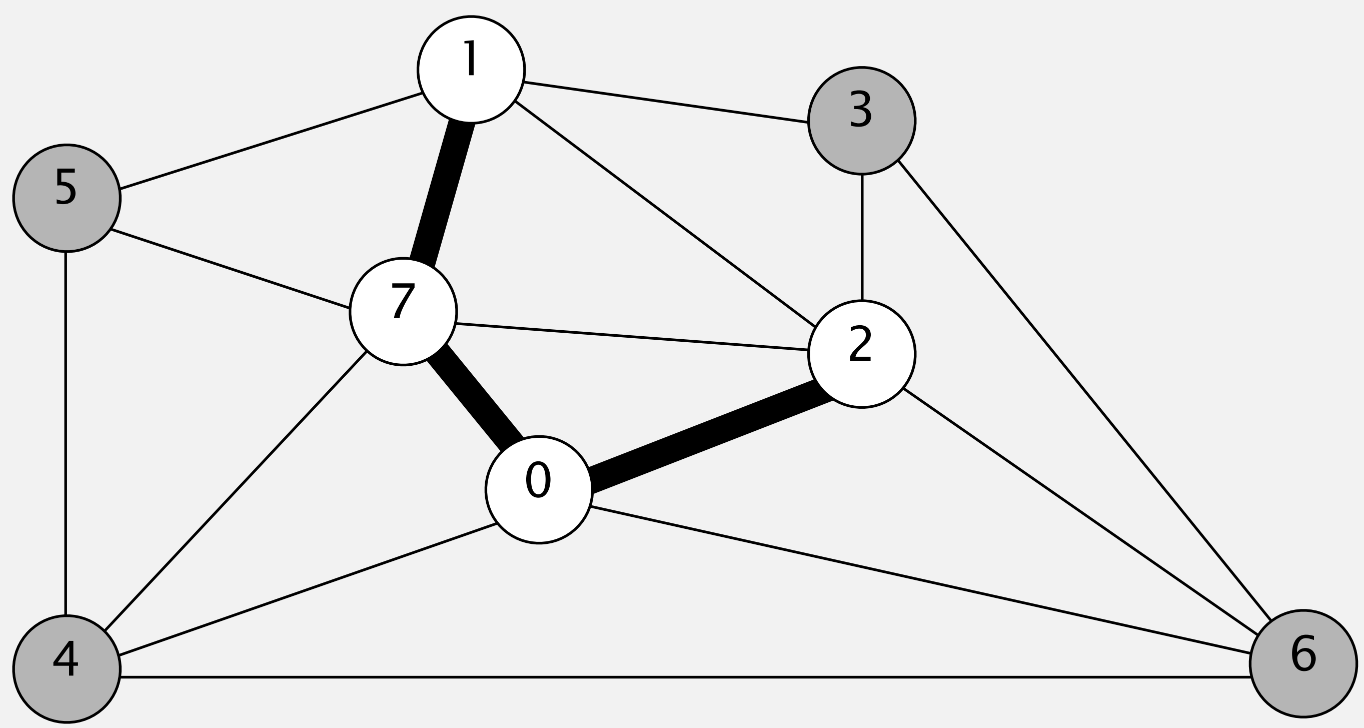
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-0	0.58

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.

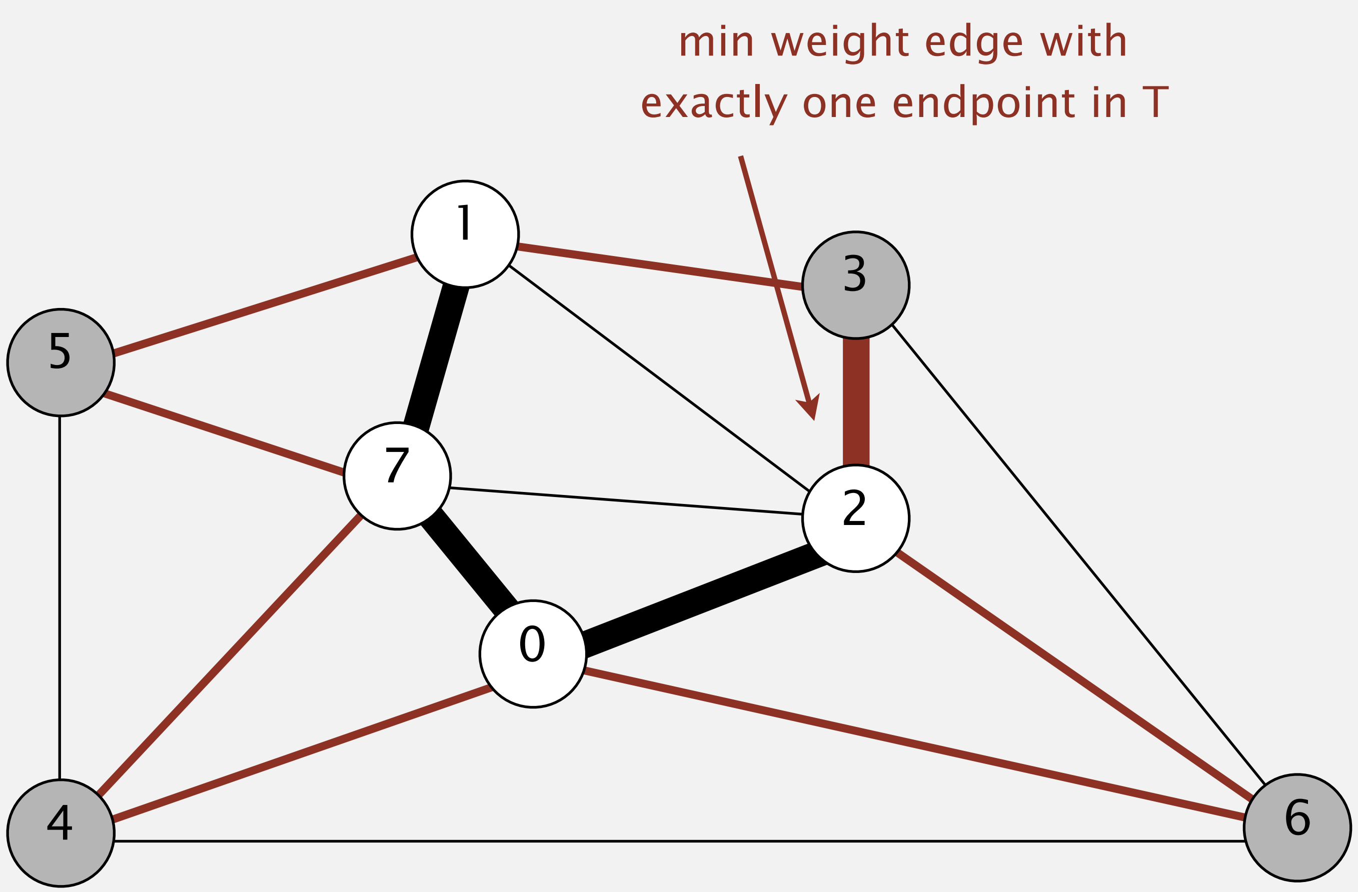


MST edges

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.



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

in MST →

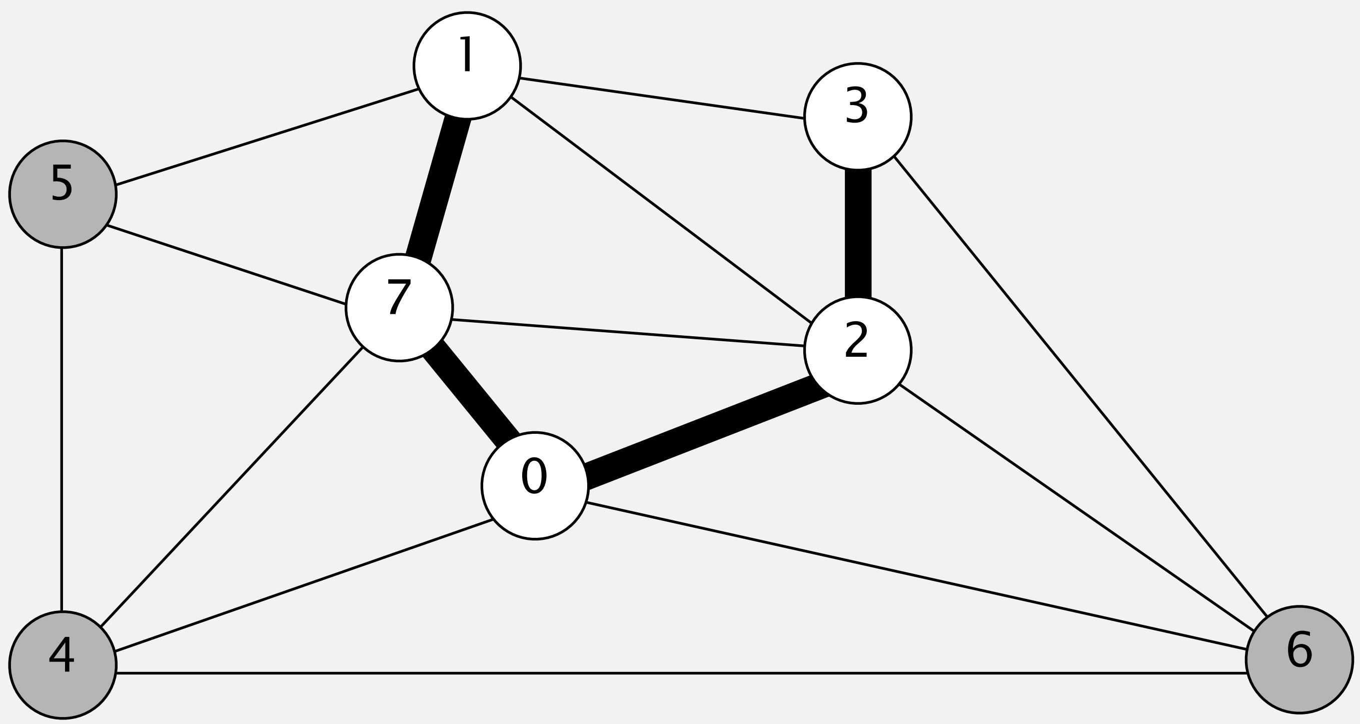
2-3	0.17
5-7	0.28
1-3	0.29
1-5	0.32
4-7	0.37
0-4	0.38
6-2	0.40
6-0	0.58

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.

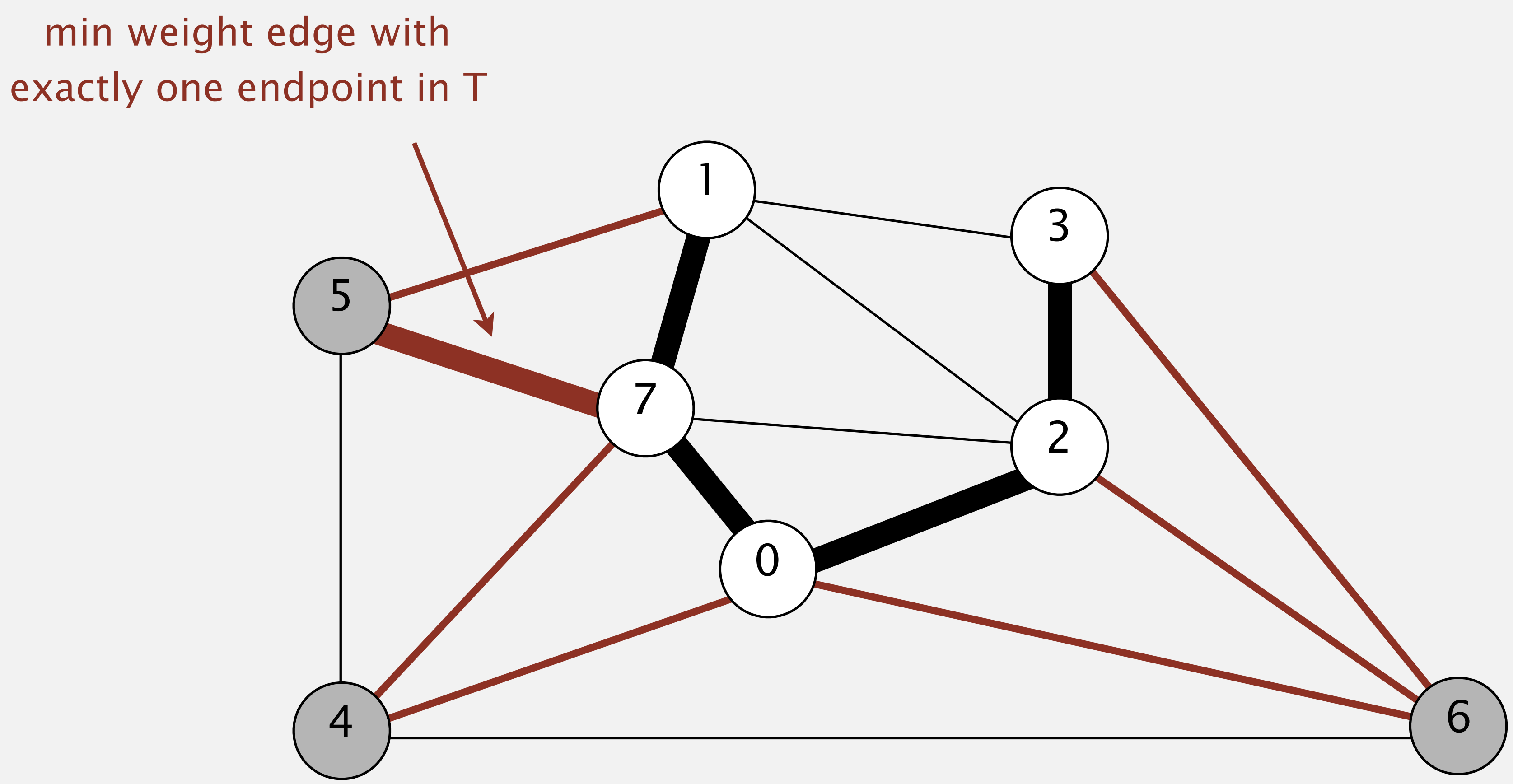


MST edges

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.



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

in MST →

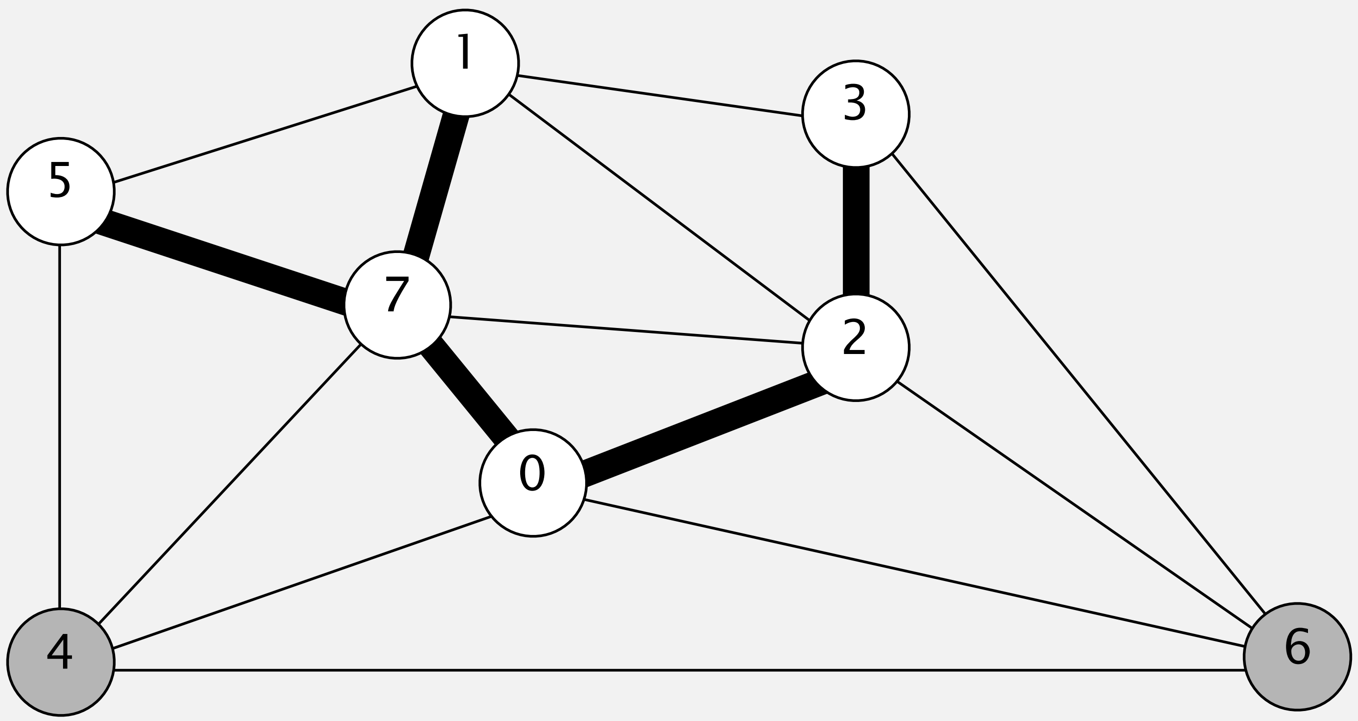
5-7	0.28
1-5	0.32
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

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.



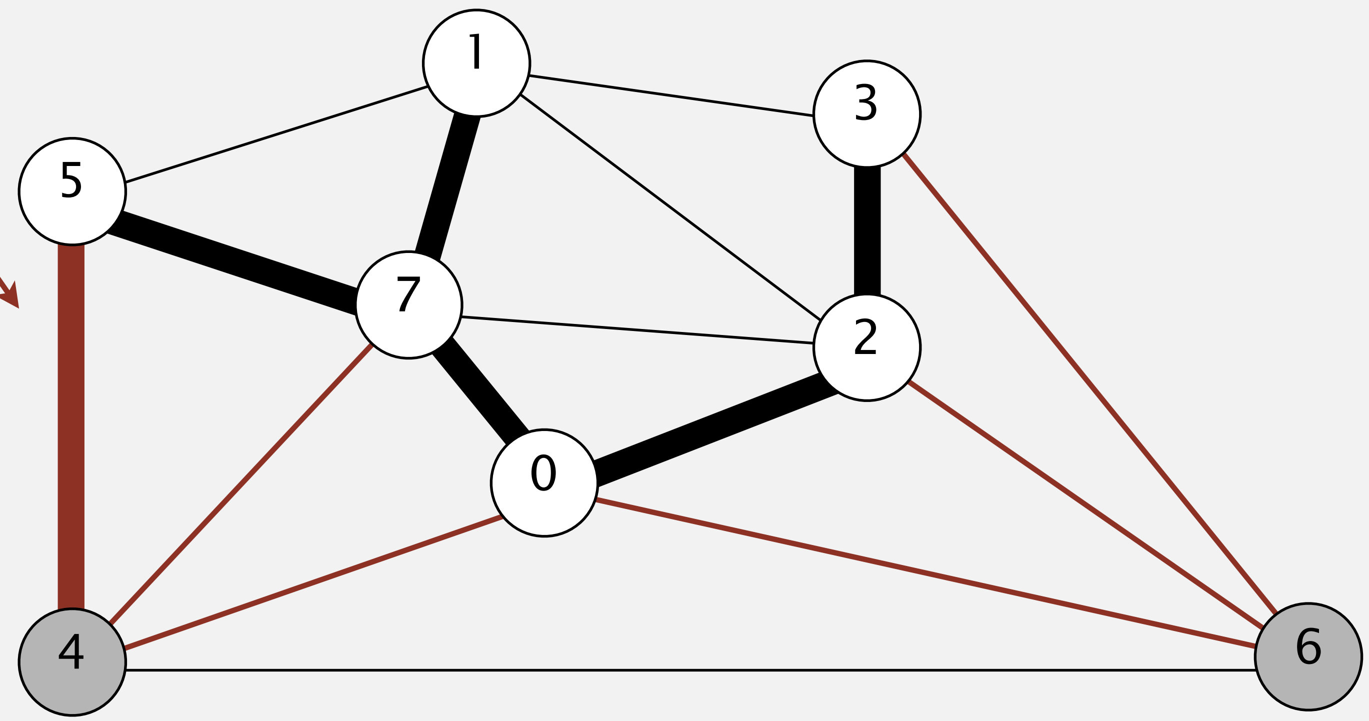
MST edges

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 →

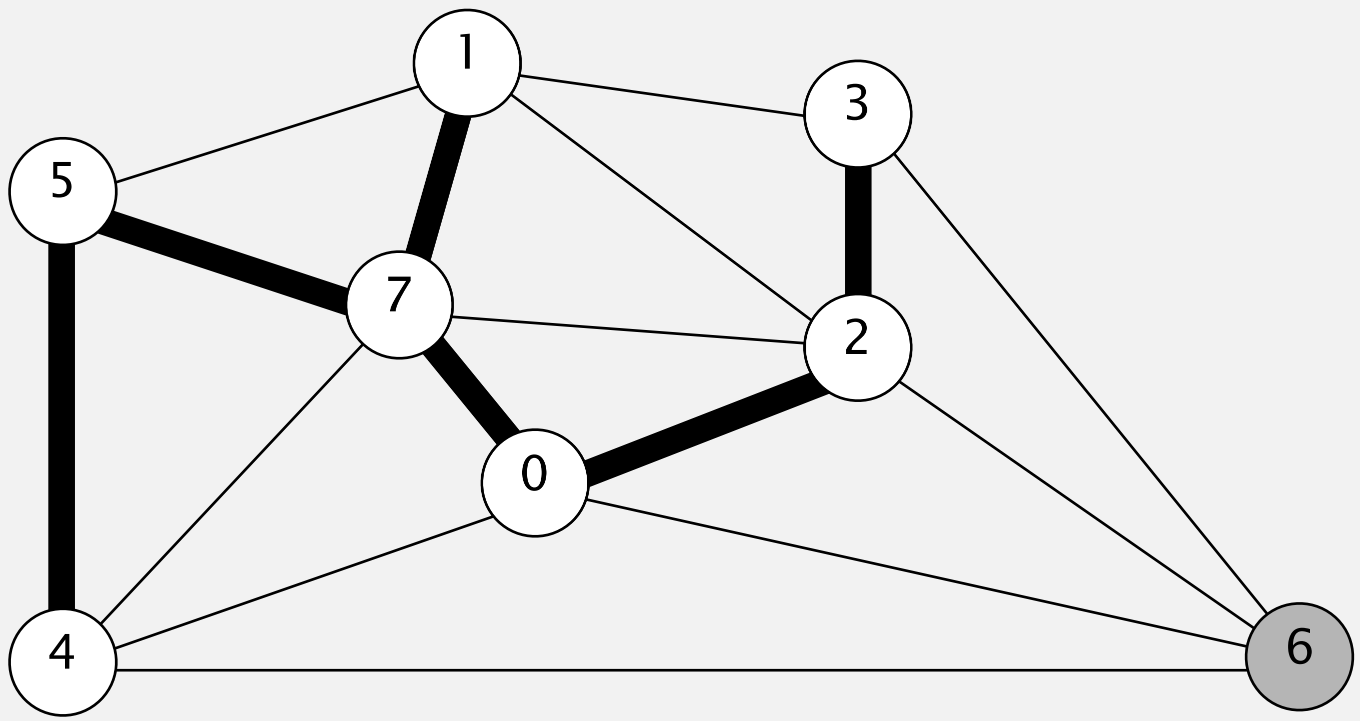
4-5	0.35
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

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.

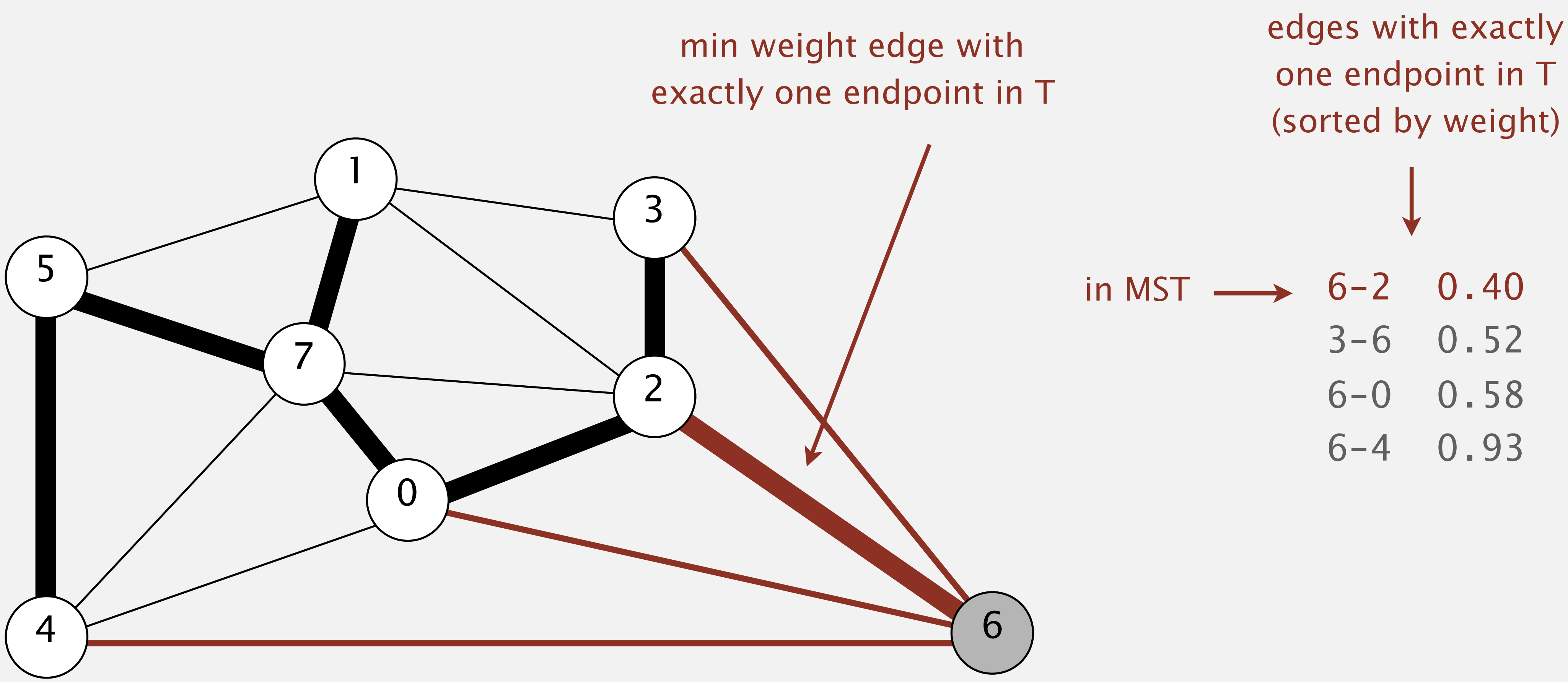


MST edges

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.

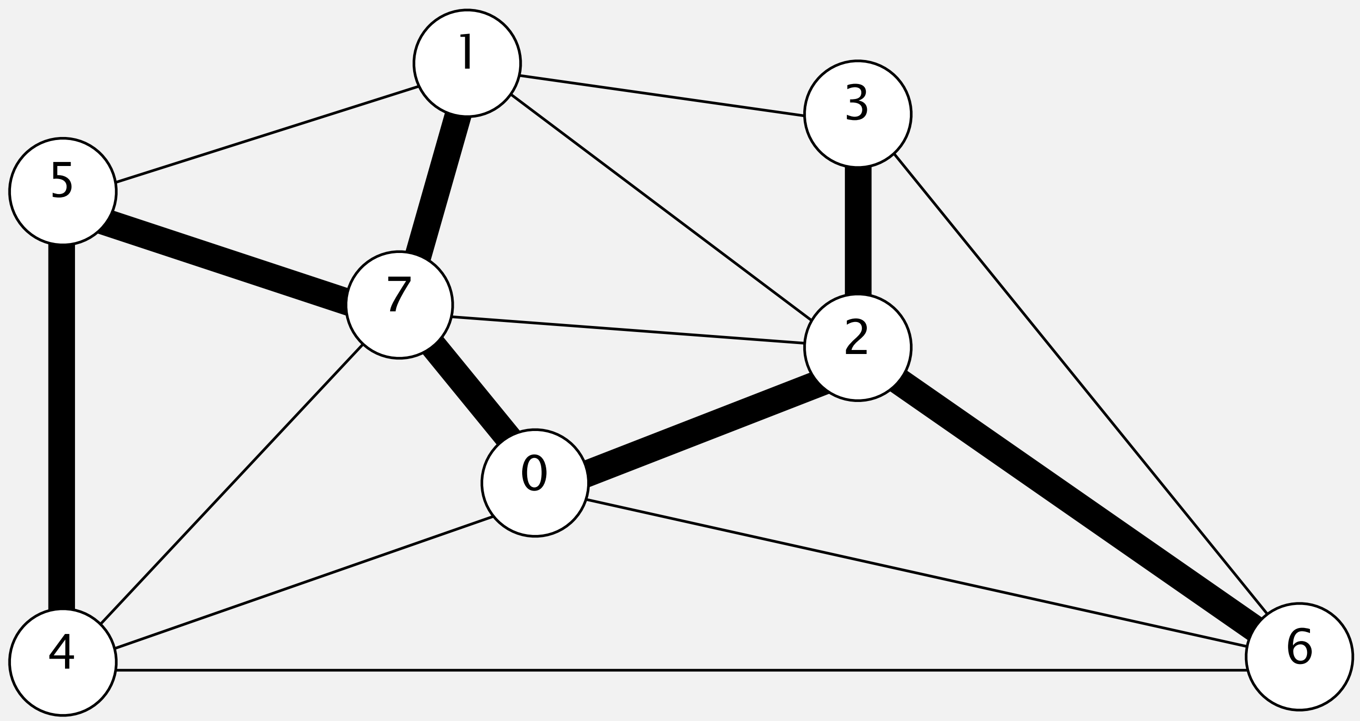


MST edges

0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm demo

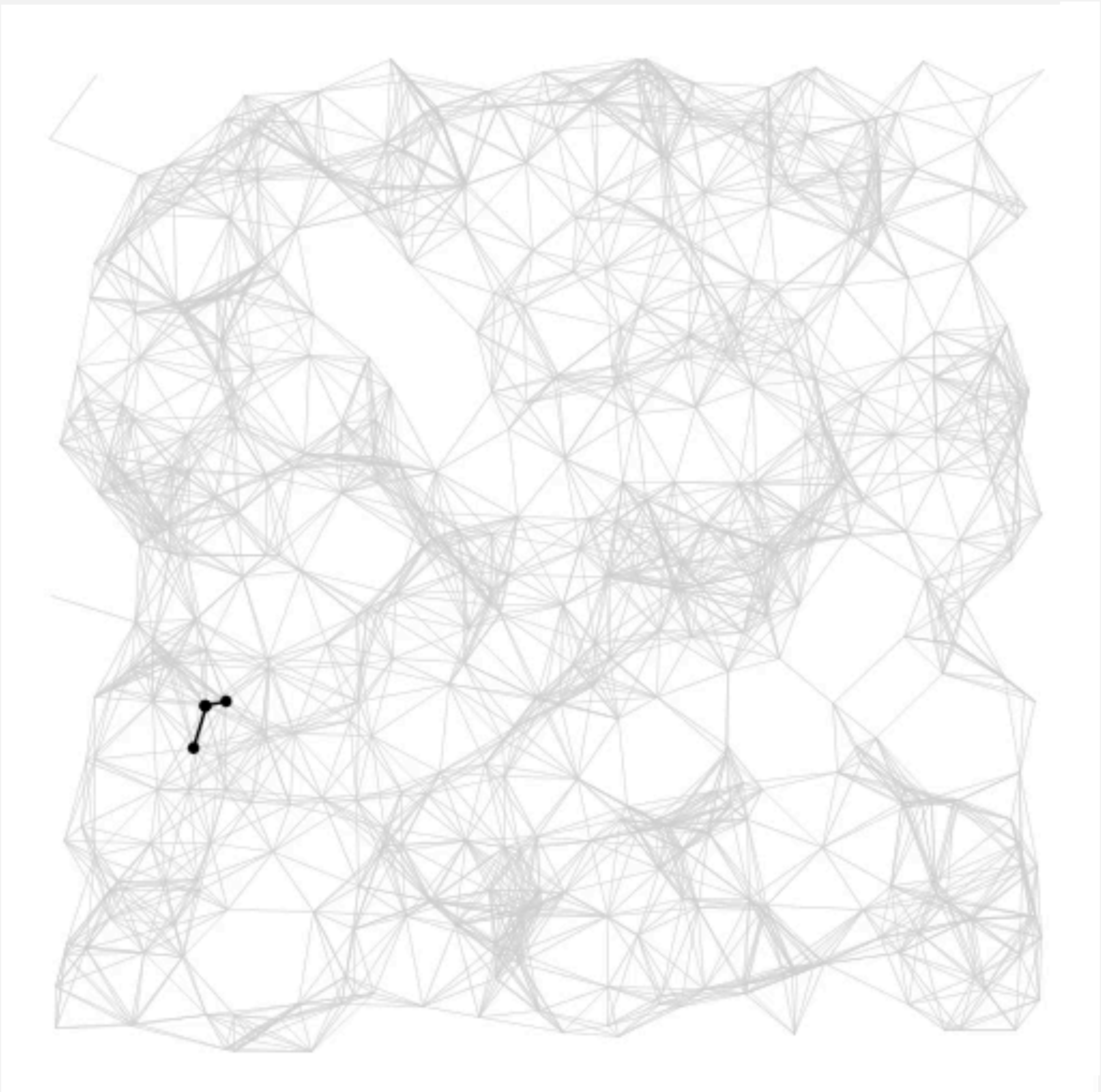
- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



MST edges

0-7 1-7 0-2 2-3 5-7 4-5 6-2

Prim’s algorithm: visualization



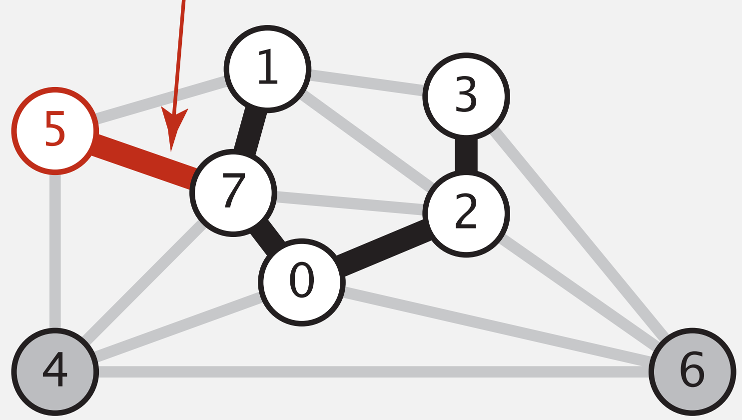
Prim's algorithm: proof of correctness

Proposition. [Jarník 1930, Dijkstra 1957, Prim 1959]
Prim's algorithm computes the MST.

Pf. Prim's algorithm is a special case of the greedy MST algorithm.

- Suppose edge e = min weight edge connecting a vertex on the tree to a vertex not on the tree.
- Cut = set of vertices connected on tree.
- No crossing edge is black.
- No crossing edge has lower weight.

edge e = 7-5 added to tree



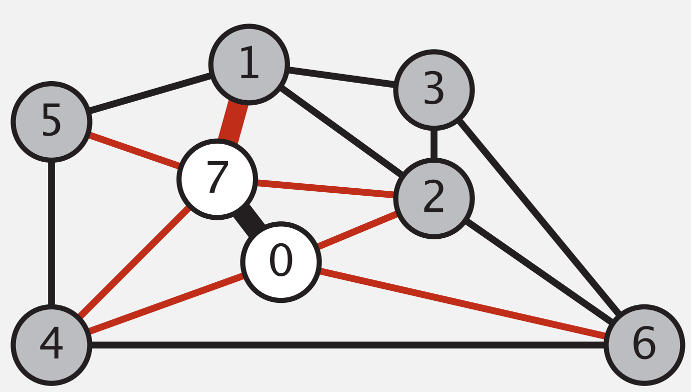
Prim's algorithm: implementation challenge

Challenge. Find the min weight edge with exactly one endpoint in T.

How difficult?

- E ← try all edges
- V
- $\log E$ ← use a priority queue!
- $\log^* E$
- 1

1-7 is min weight edge with exactly one endpoint in T



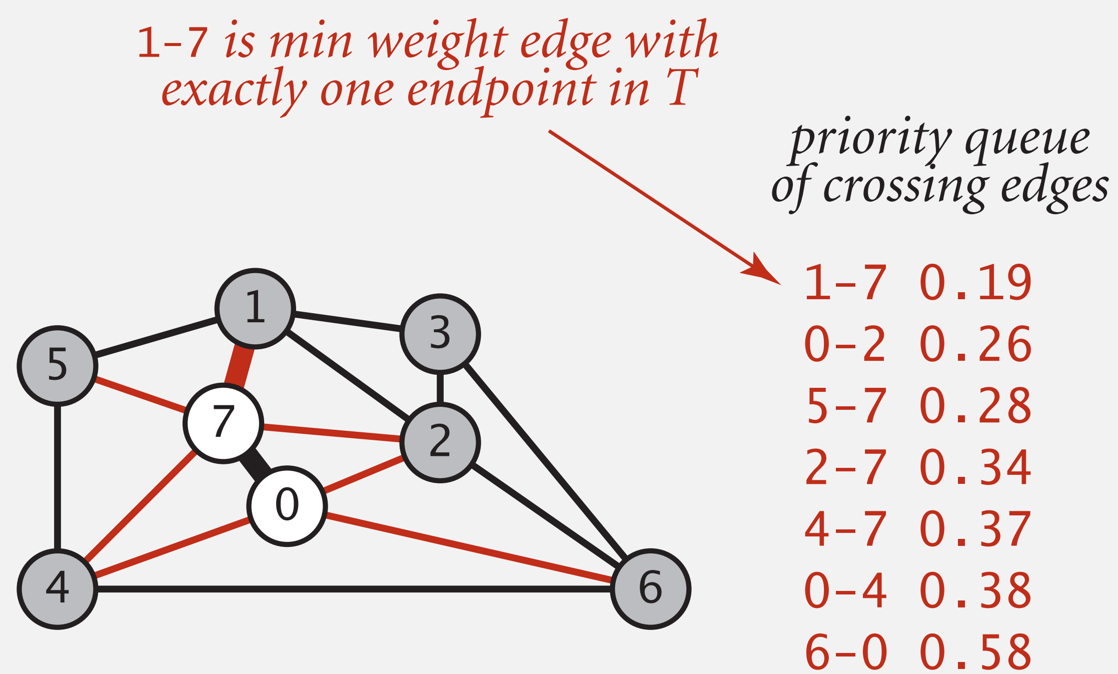

```
1-7 0.19
0-2 0.26
5-7 0.28
2-7 0.34
4-7 0.37
0-4 0.38
6-0 0.58
```

Prim's algorithm: lazy implementation

Challenge. Find the min weight edge with exactly one endpoint in T.

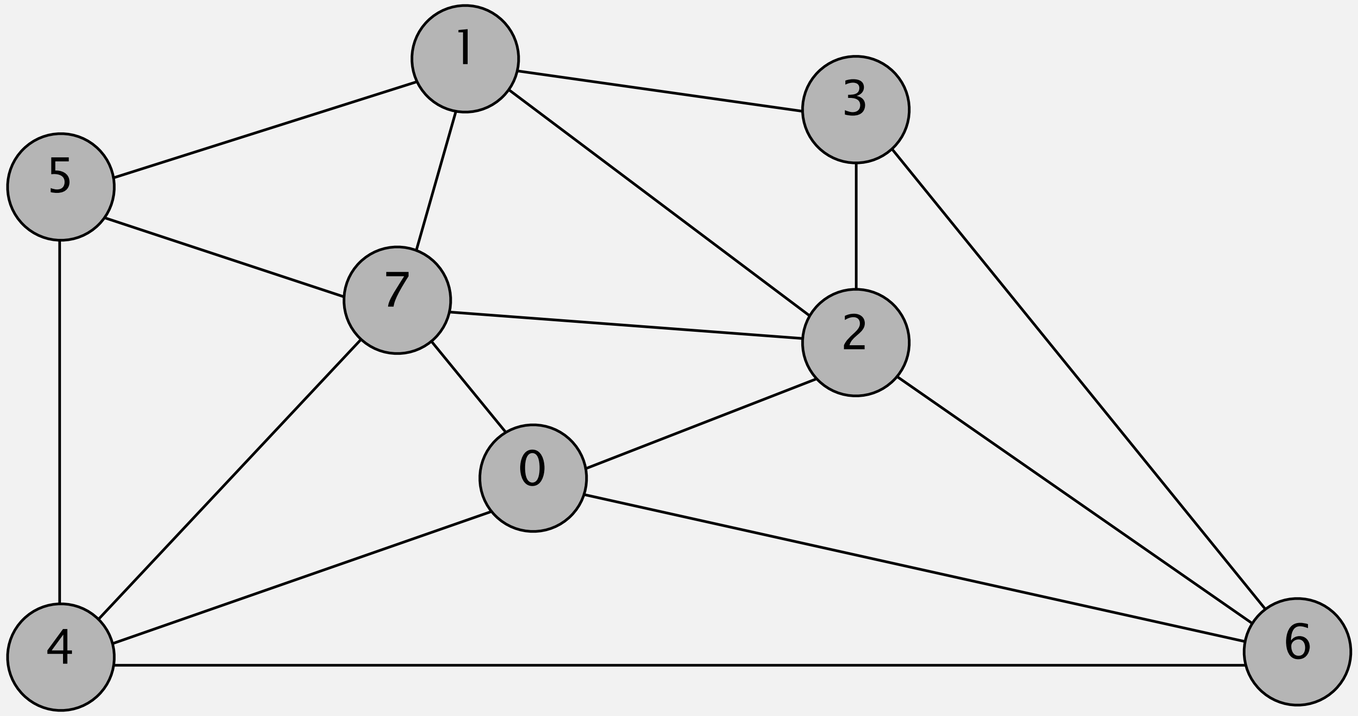
Lazy solution. Maintain a PQ of edges with (at least) one endpoint in T.

- Key = edge; priority = weight of edge.
- Delete-min to determine next edge $e = v\text{–}w$ to add to T.
- Disregard if both endpoints v and w are marked (both in T).
- Otherwise, let w be the unmarked vertex (not in T):
  - add to PQ any edge incident to w (assuming other endpoint not in T)
  - add e to T and mark w



Prim's algorithm: lazy implementation 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.

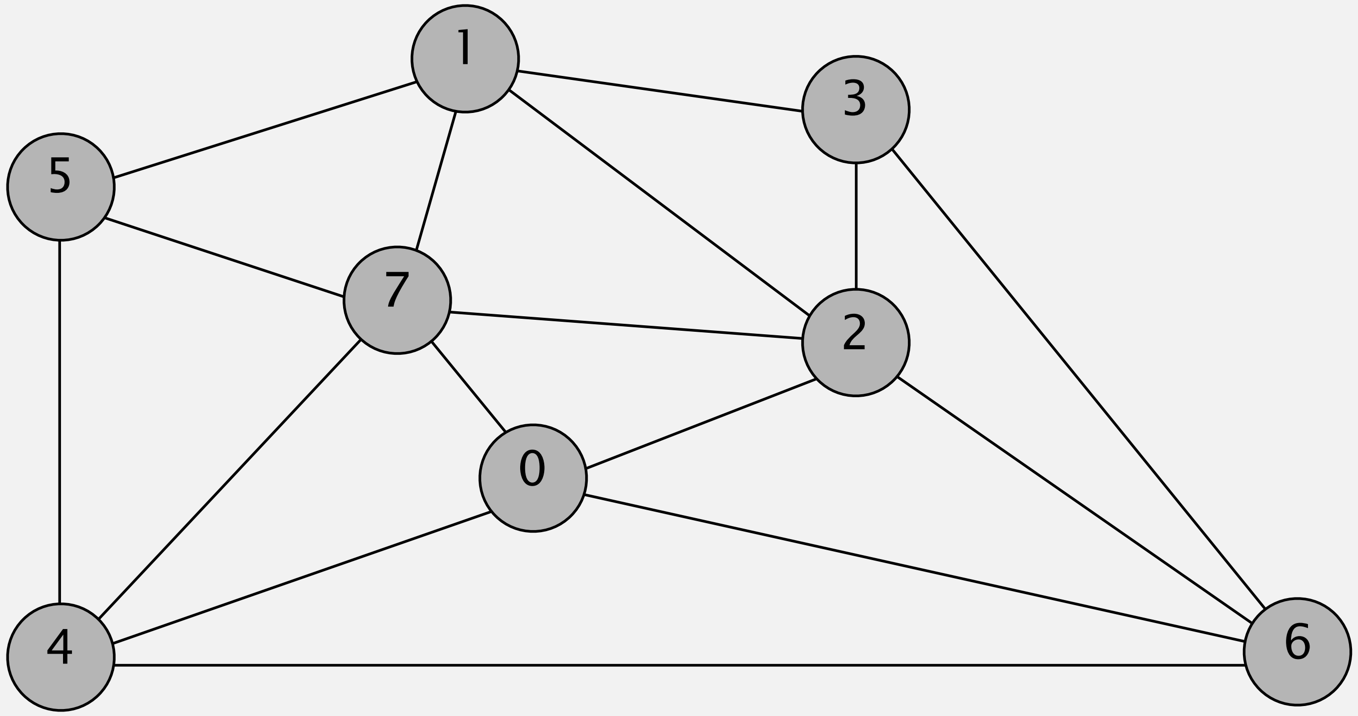


an edge-weighted graph

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: lazy implementation 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.

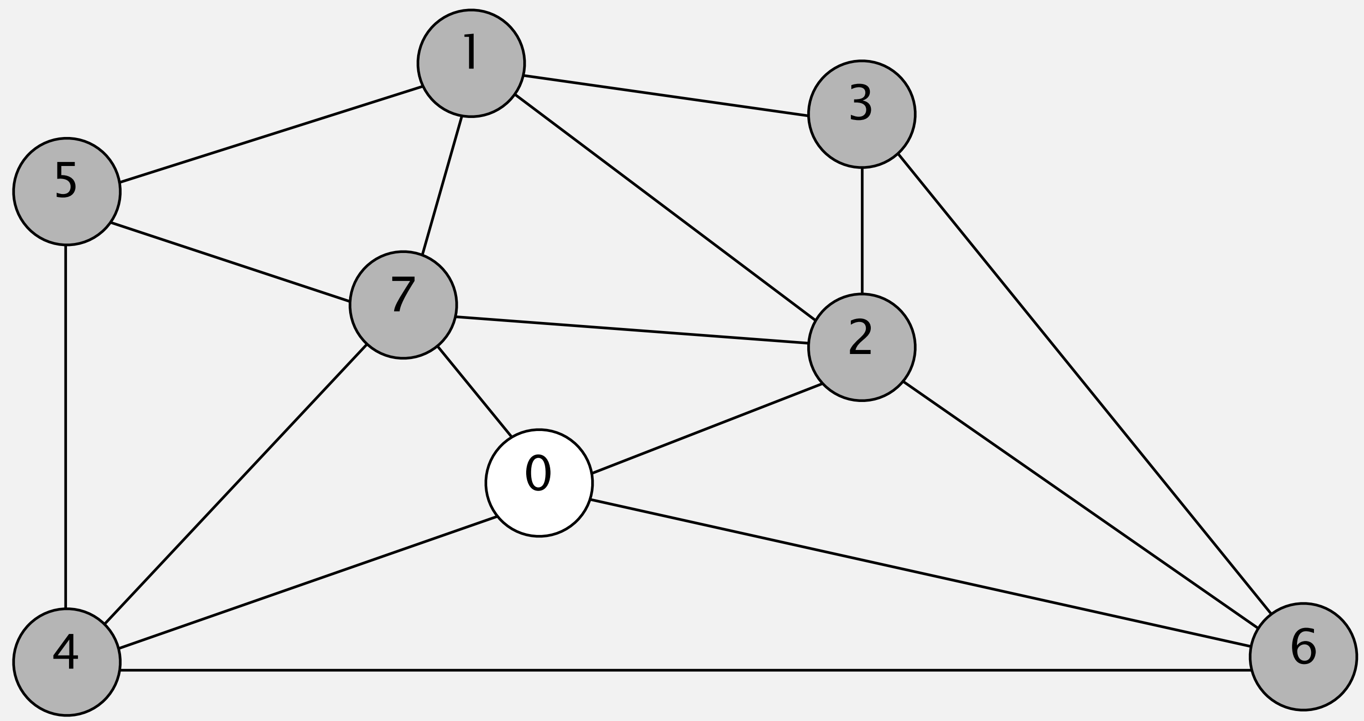


an edge-weighted graph

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: lazy implementation demo

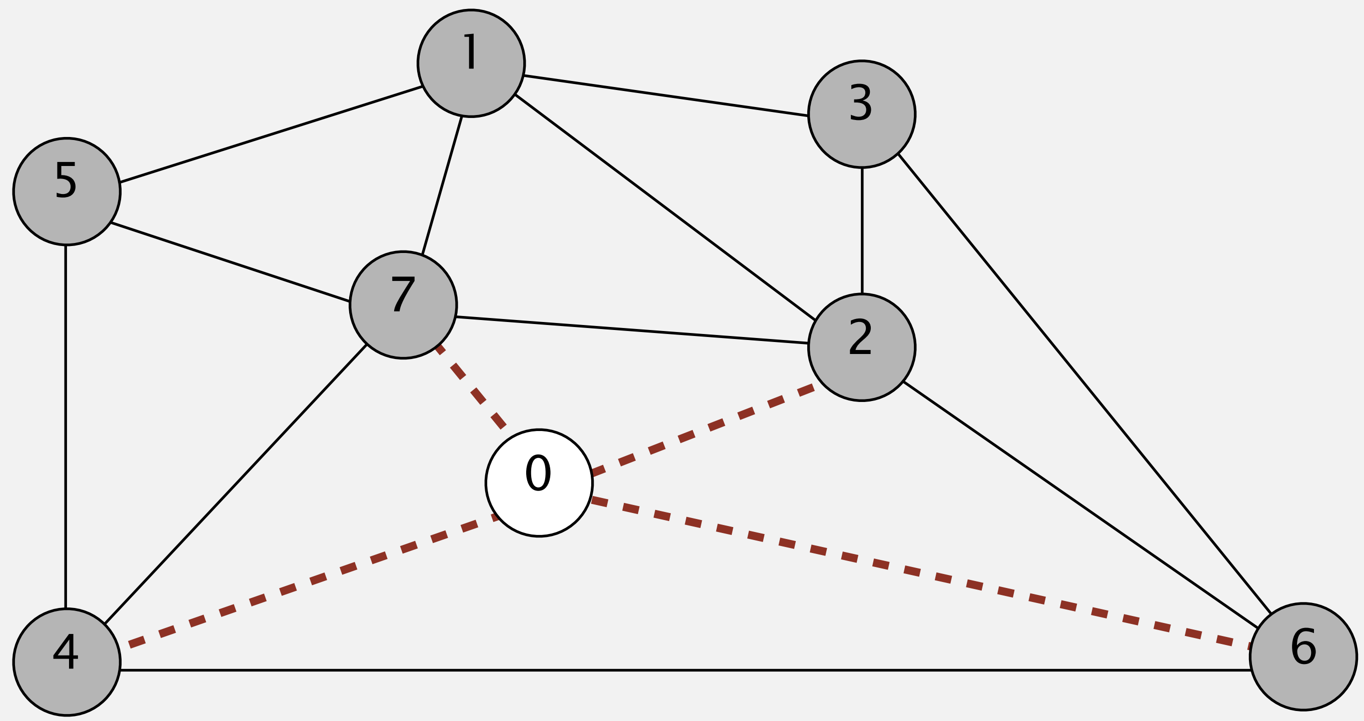
- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



Prim's algorithm: lazy implementation 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.

add to PQ all edges incident to 0



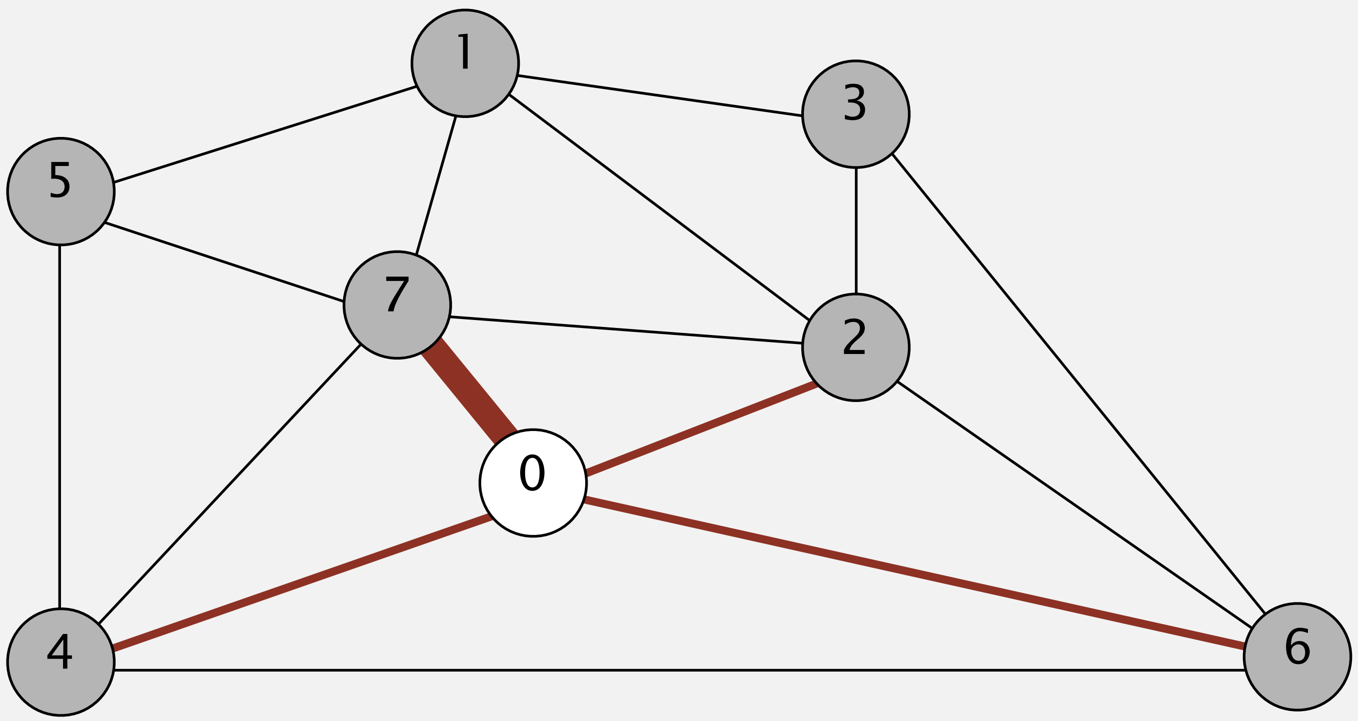
edges on PQ
(sorted by weight)

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

Prim's algorithm: lazy implementation 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.

delete 0-7 and add to MST

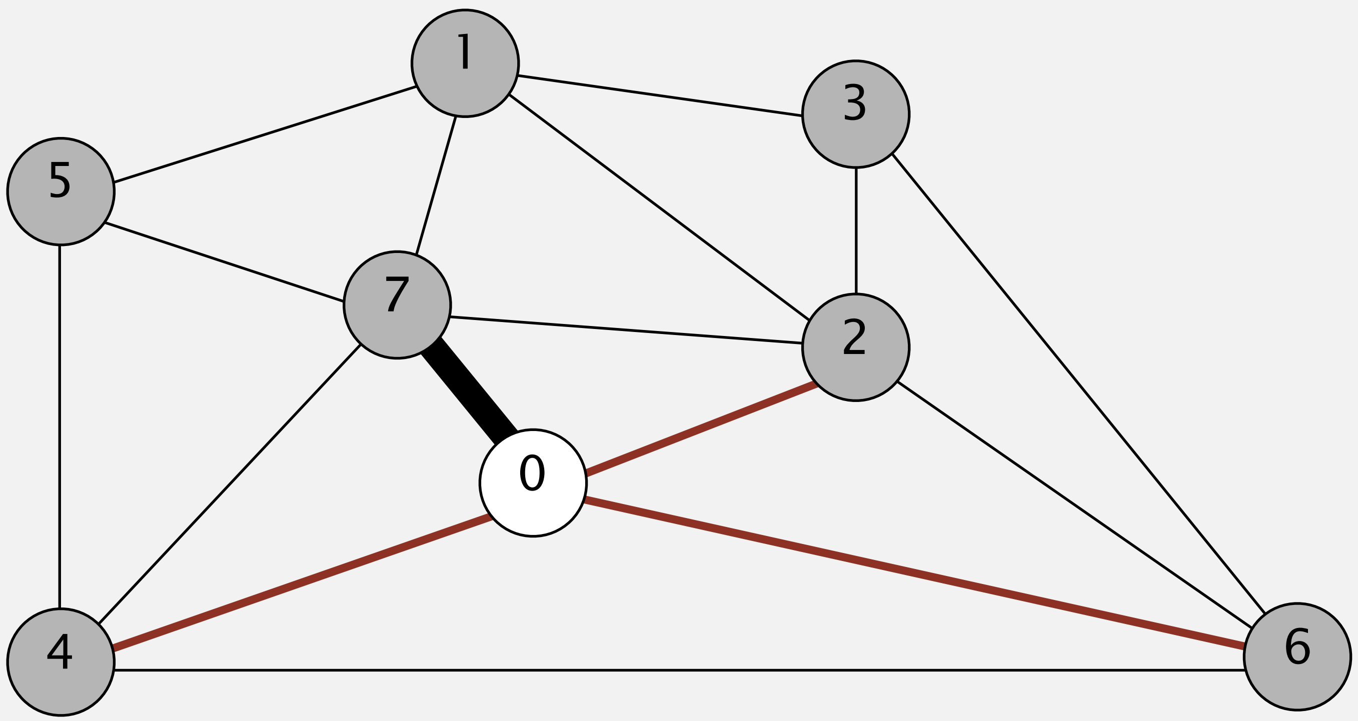


edges on PQ
(sorted by weight)

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

Prim's algorithm: lazy implementation 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 on PQ
(sorted by weight)

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

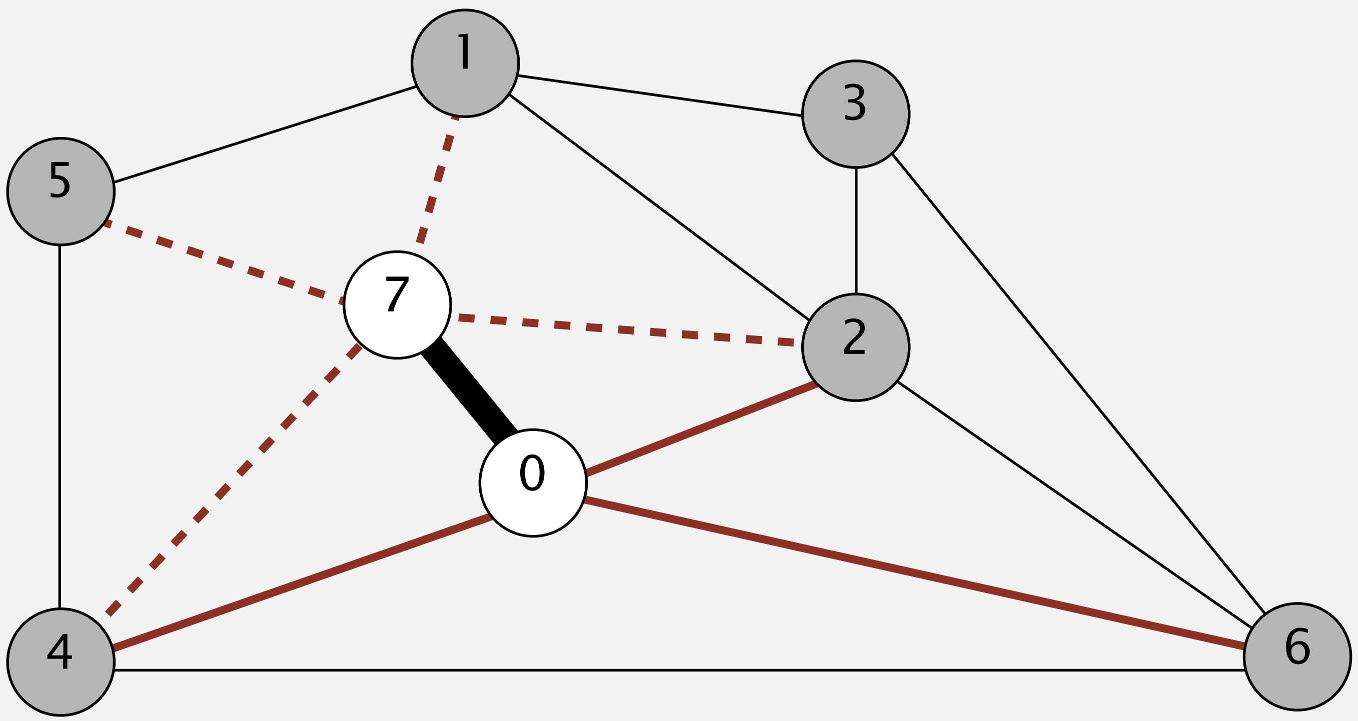
MST edges

0-7

Prim's algorithm: lazy implementation 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.

add to PQ all edges incident to 7



edges on PQ
(sorted by weight)

*	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

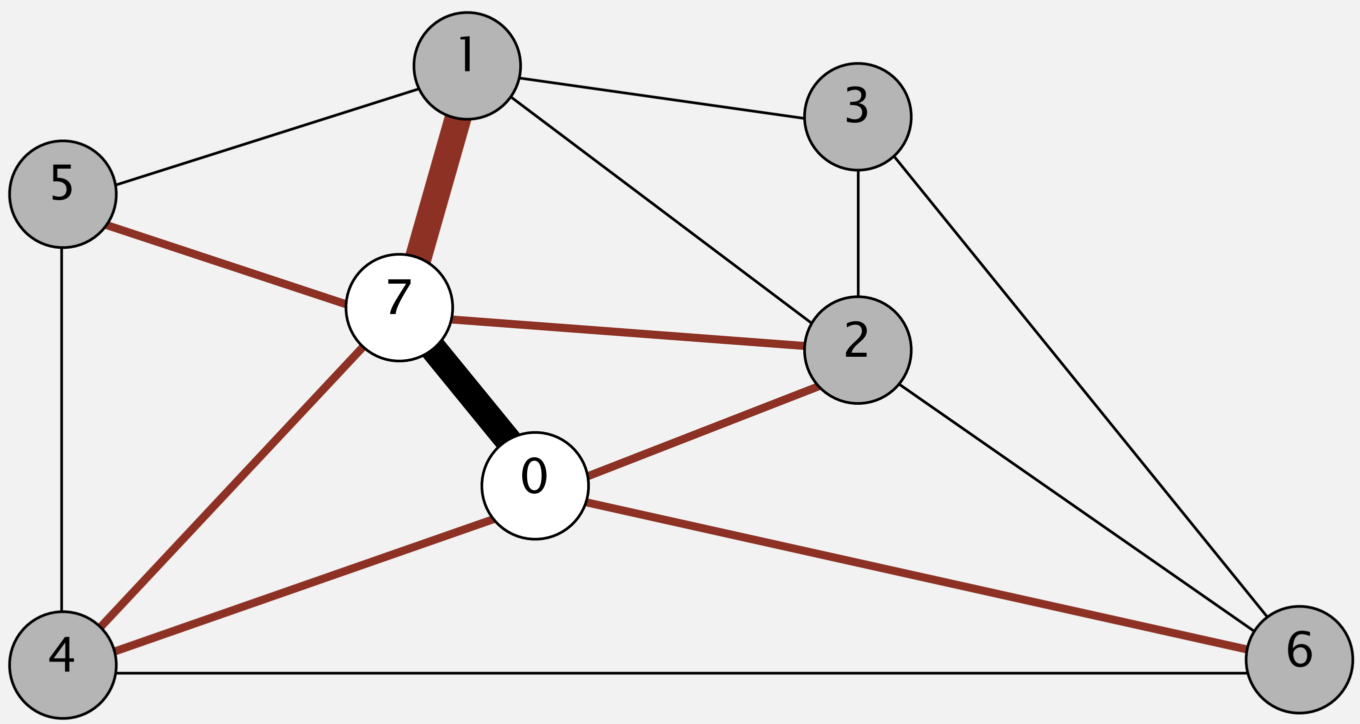
MST edges

0-7

Prim's algorithm: lazy implementation 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.

delete 1–7 and add to MST



edges on PQ
(sorted by weight)

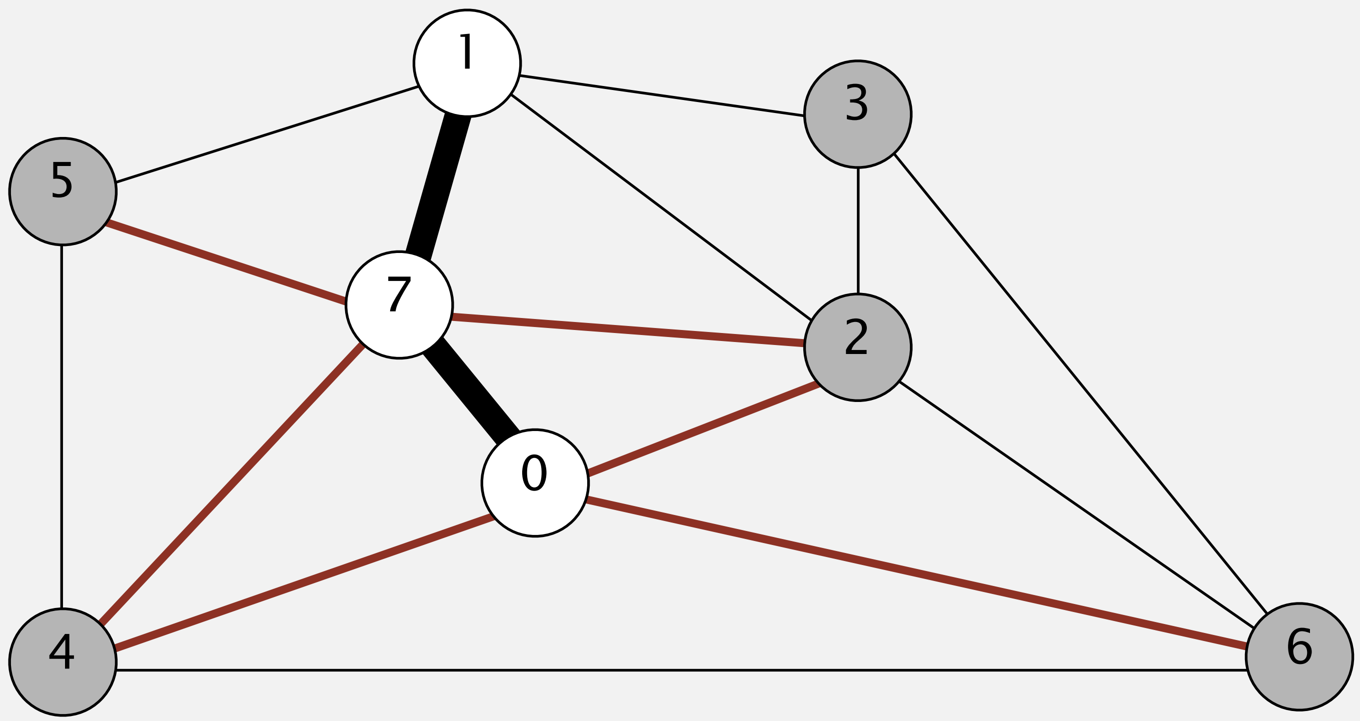
1-7	0.19
0-2	0.26
5-7	0.28
2-7	0.34
4-7	0.37
0-4	0.38
6-0	0.58

MST edges

0–7

Prim's algorithm: lazy implementation 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 on PQ
(sorted by weight)

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

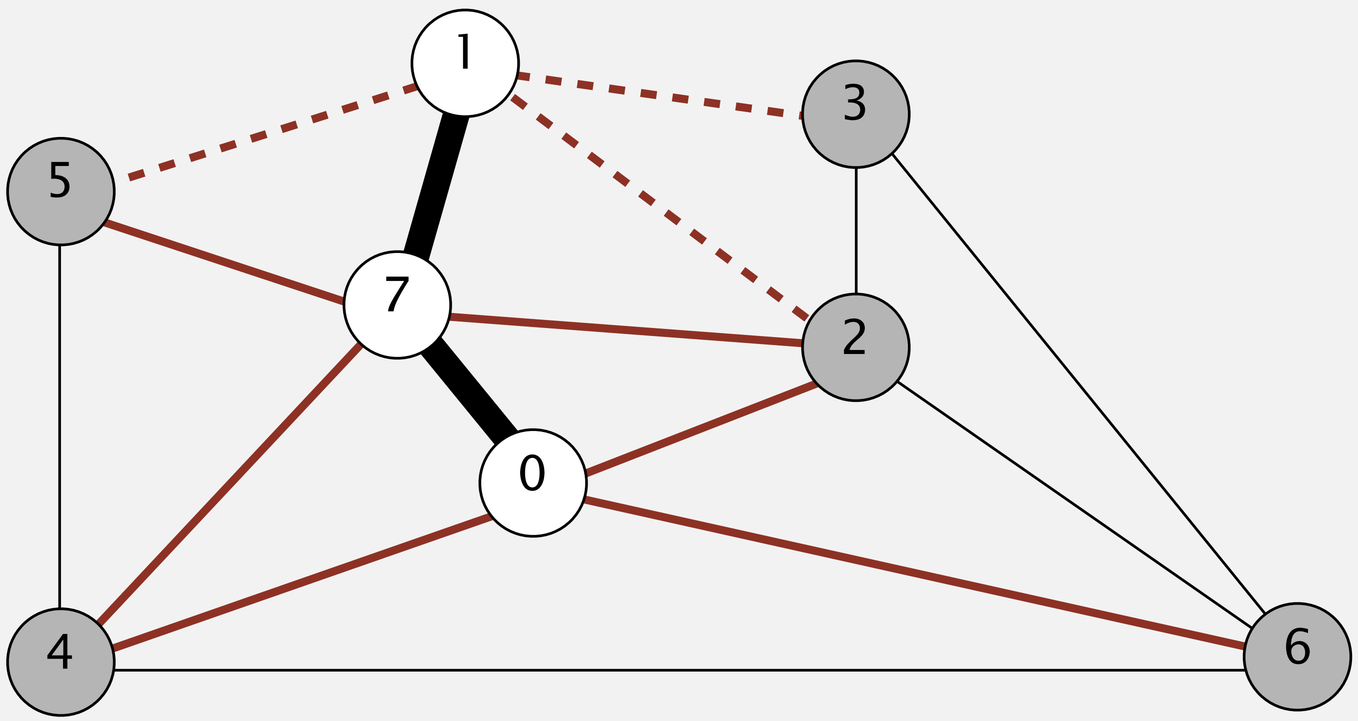
MST edges

0-7 1-7

Prim's algorithm: lazy implementation 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.

add to PQ all edges incident to 1



edges on PQ
(sorted by weight)

	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

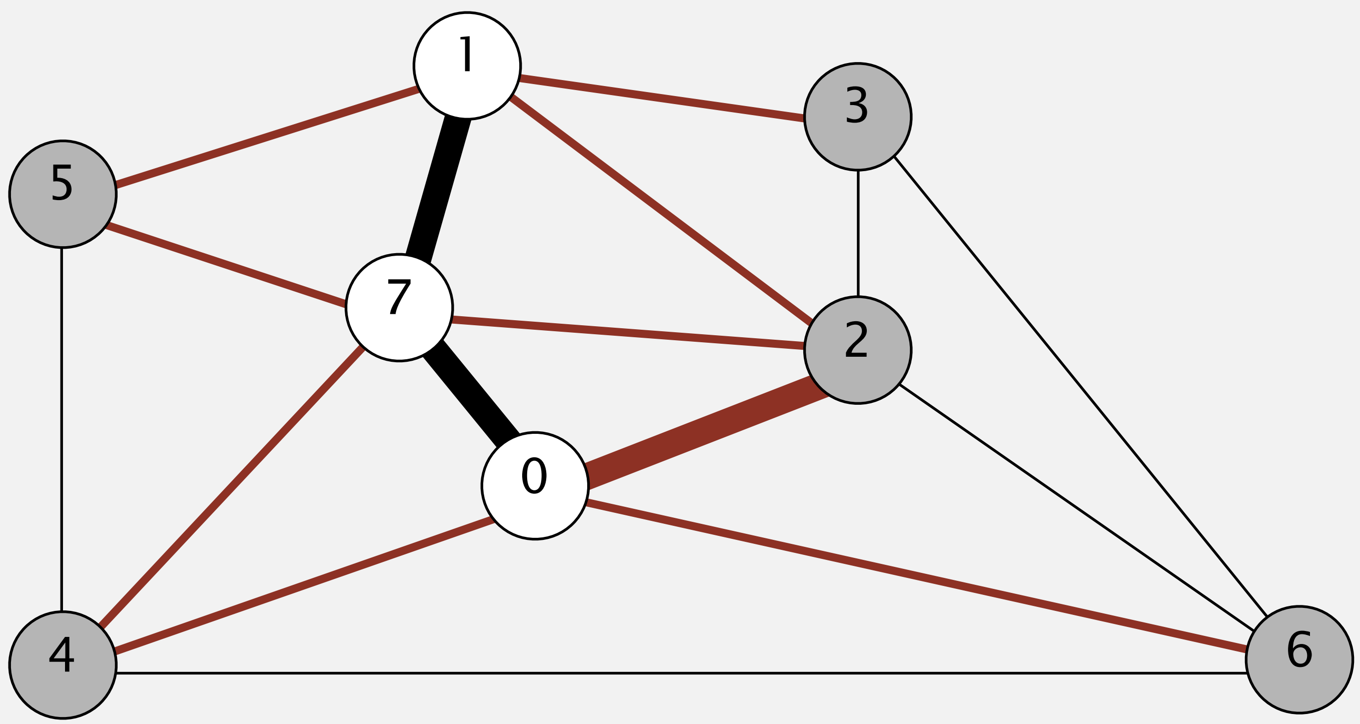
MST edges

0-7 1-7

Prim's algorithm: lazy implementation 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.

delete edge 0-2 and add to MST



edges on PQ
(sorted by weight)

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

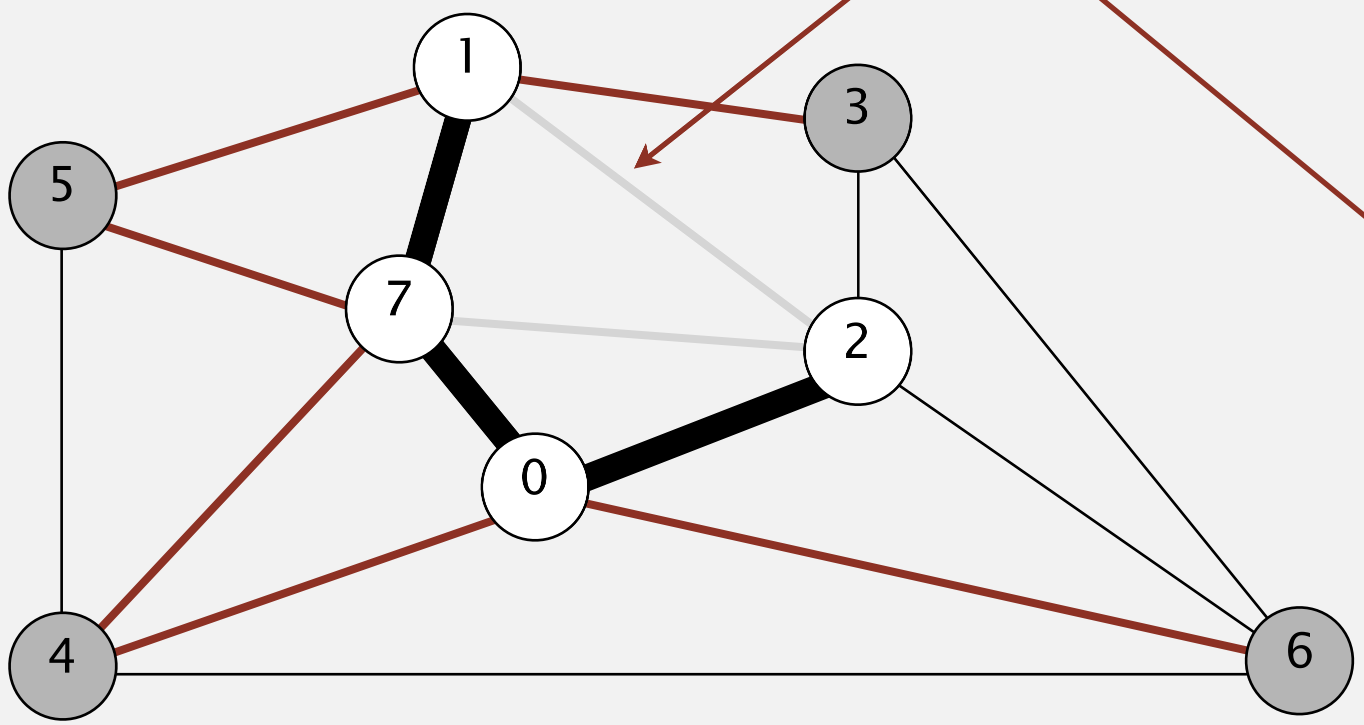
MST edges

0-7 1-7

Prim's algorithm: lazy implementation 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.

edge becomes obsolete
(lazy implementation leaves on PQ)



edges on PQ
(sorted by weight)

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

MST edges

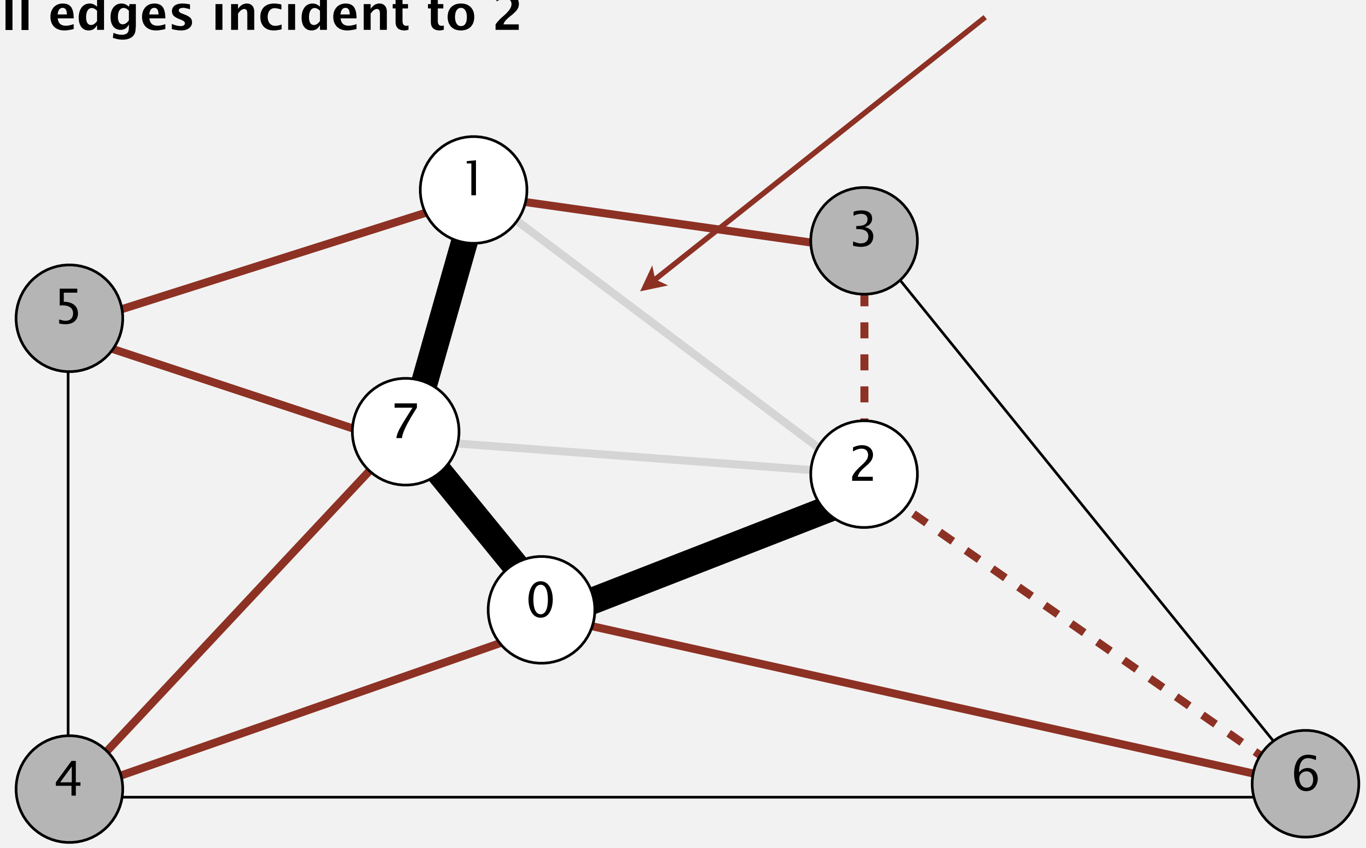
0-7 1-7 0-2

Prim's algorithm: lazy implementation 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.

no need to add edge 1-2 or 2-7
because it's already obsolete

add to PQ all edges incident to 2



edges on PQ
(sorted by weight)

*	2-3	0.17
	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
*	6-2	0.40
	6-0	0.58

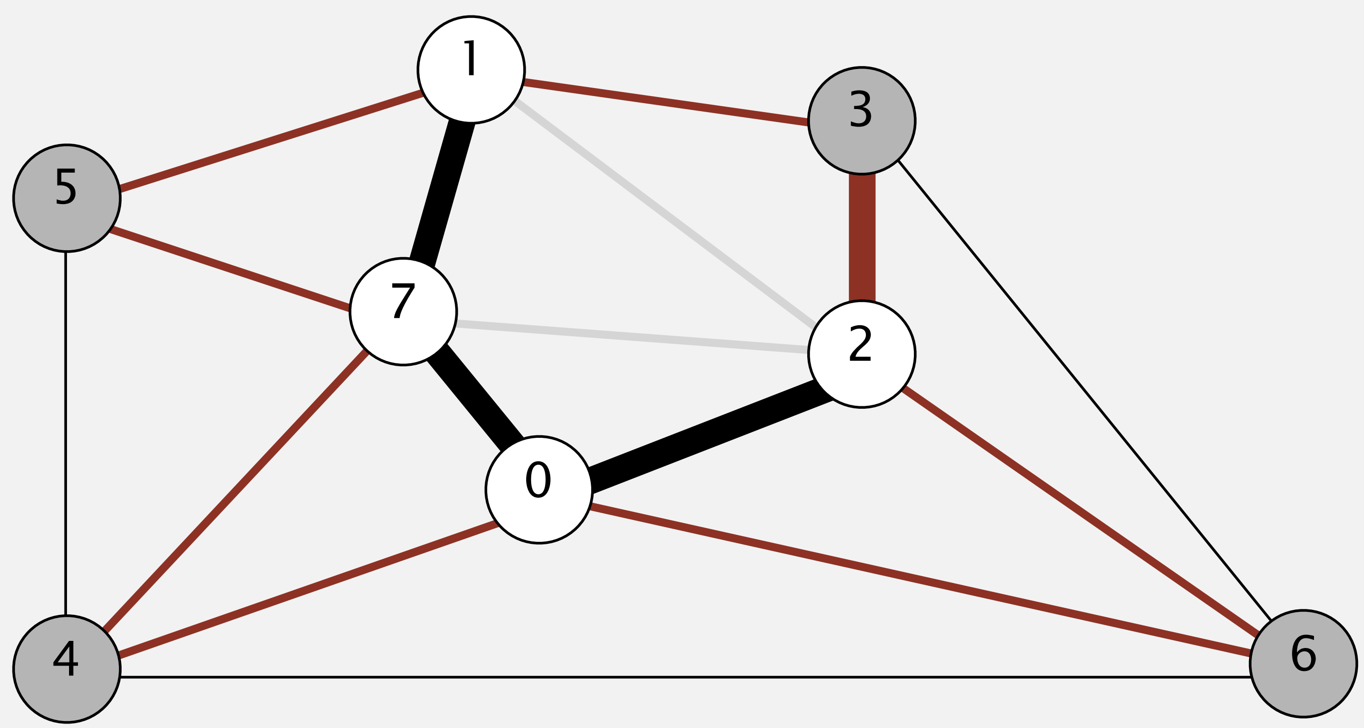
MST edges

0-7 1-7 0-2

Prim's algorithm: lazy implementation 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.

delete 2-3 and add to MST



edges on PQ
(sorted by weight)

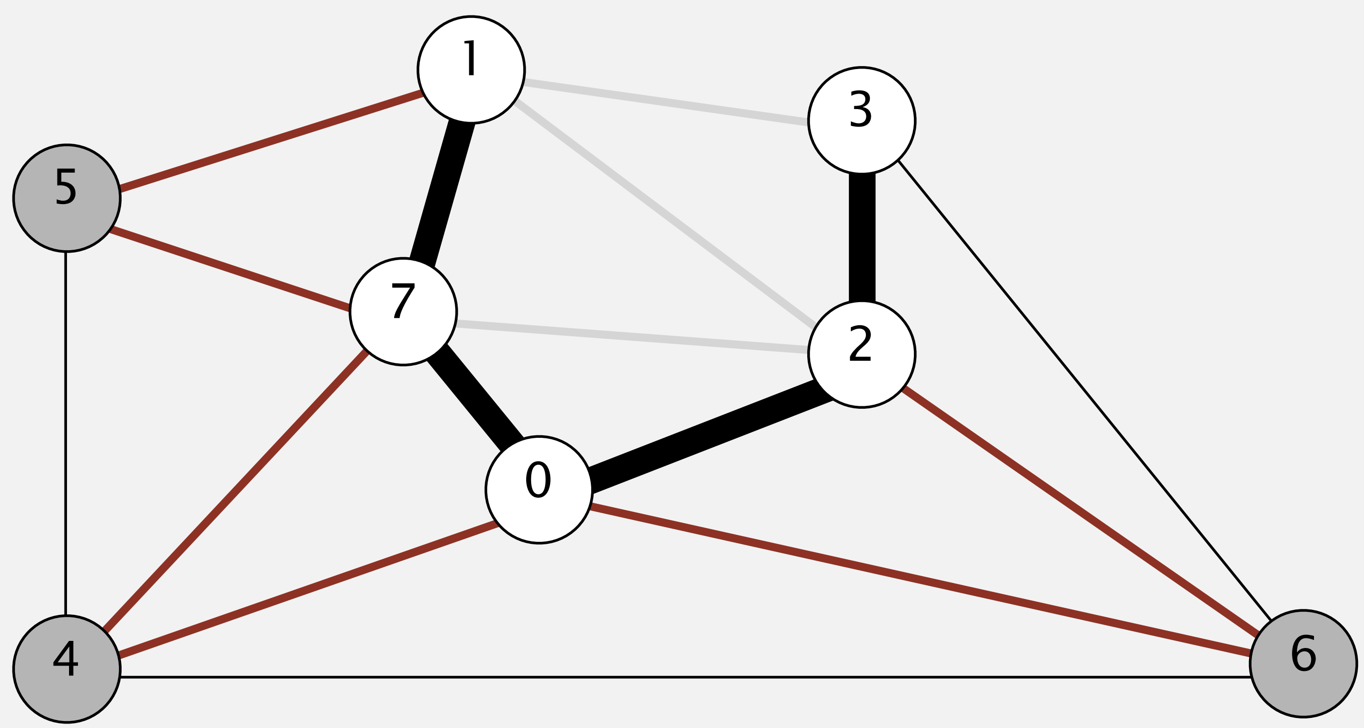
*	2-3	0.17
	5-7	0.28
	1-3	0.29
	1-5	0.32
	2-7	0.34
	1-2	0.36
	4-7	0.37
	0-4	0.38
*	6-2	0.40
	6-0	0.58

MST edges

0-7 1-7 0-2

Prim's algorithm: lazy implementation 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 on PQ
(sorted by weight)

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
6-2	0.40
6-0	0.58

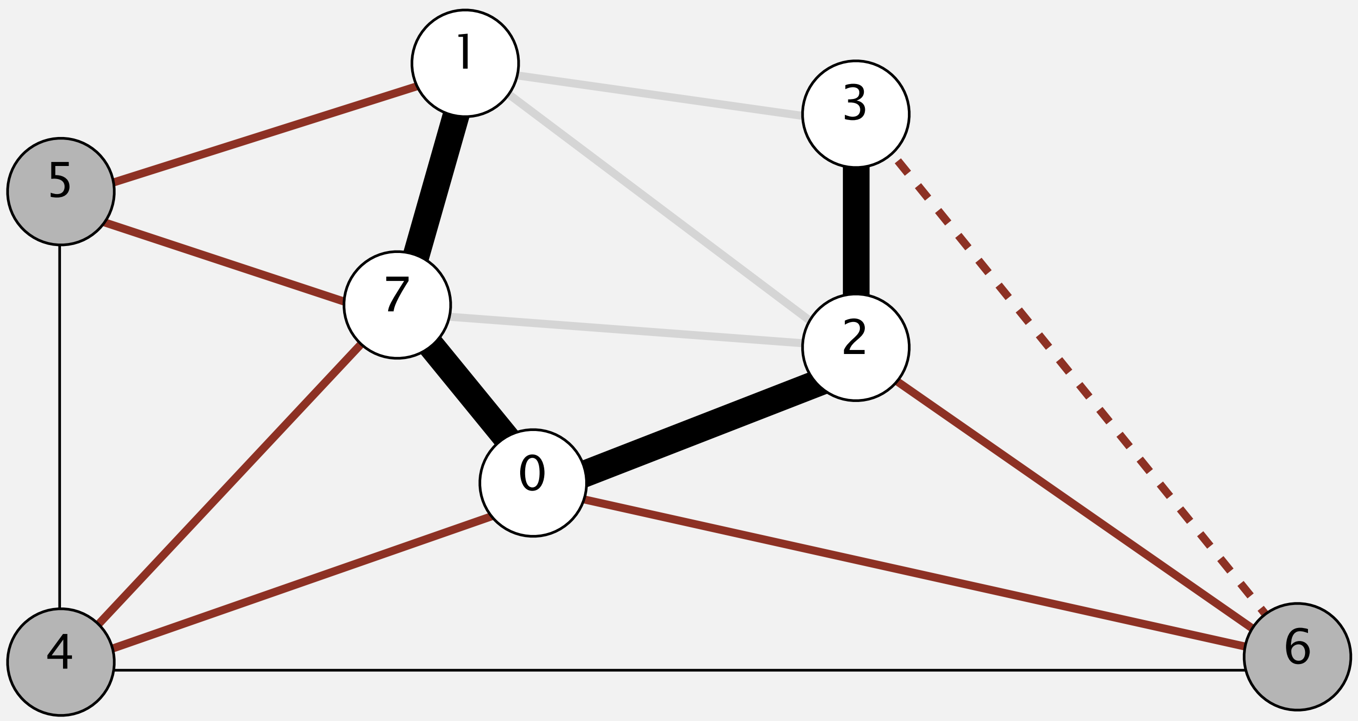
MST edges

0-7 1-7 0-2 2-3

Prim's algorithm: lazy implementation 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.

add to PQ all edges incident to 3



edges on PQ
(sorted by weight)

	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
	6-2	0.40
*	3-6	0.52
	6-0	0.58

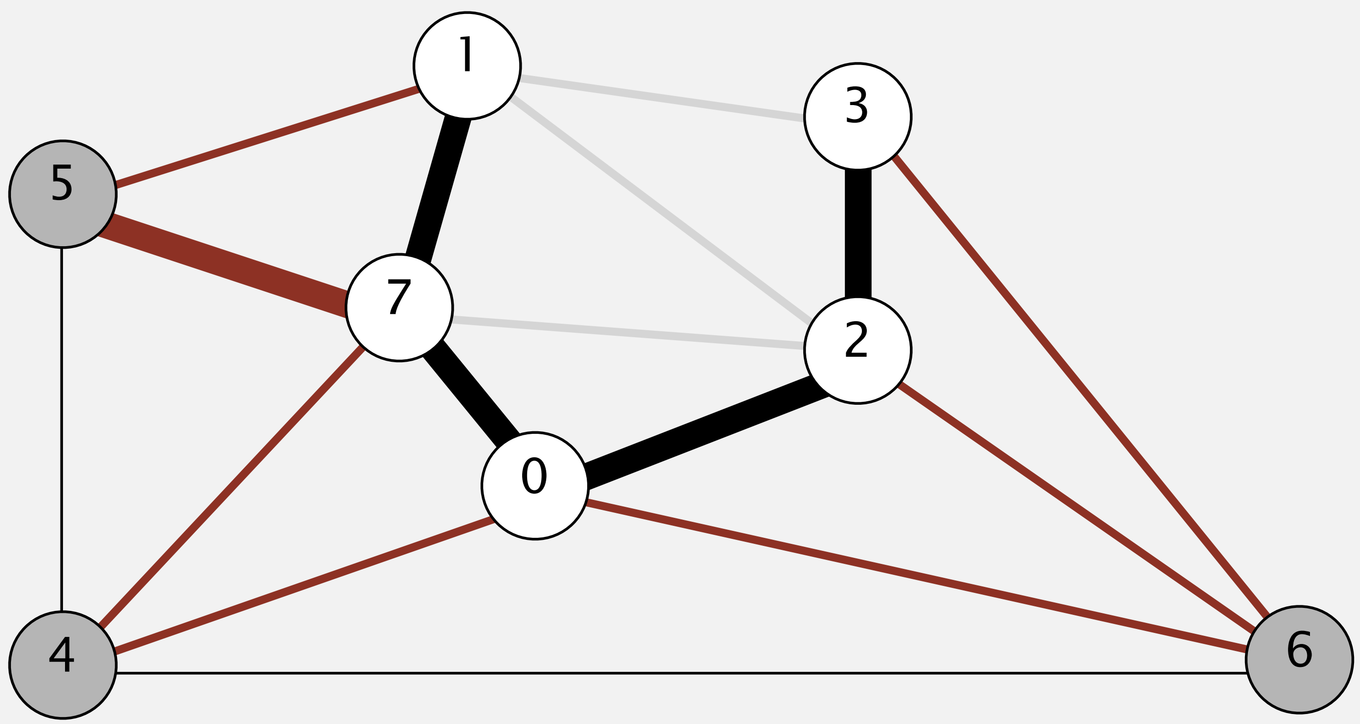
MST edges

0-7 1-7 0-2 2-3

Prim's algorithm: lazy implementation 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.

delete 5-7 and add to MST



edges on PQ
(sorted by weight)

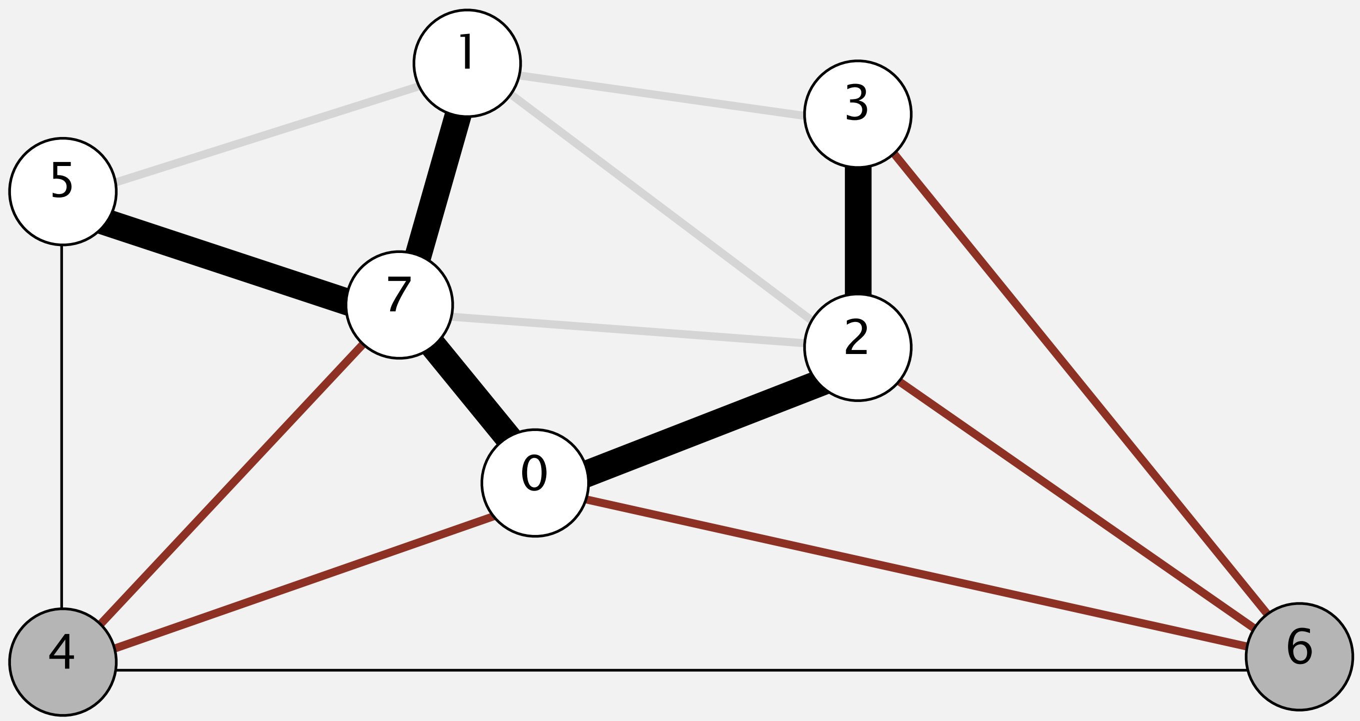
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

MST edges

0-7 1-7 0-2 2-3

Prim's algorithm: lazy implementation 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 on PQ
(sorted by weight)

1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

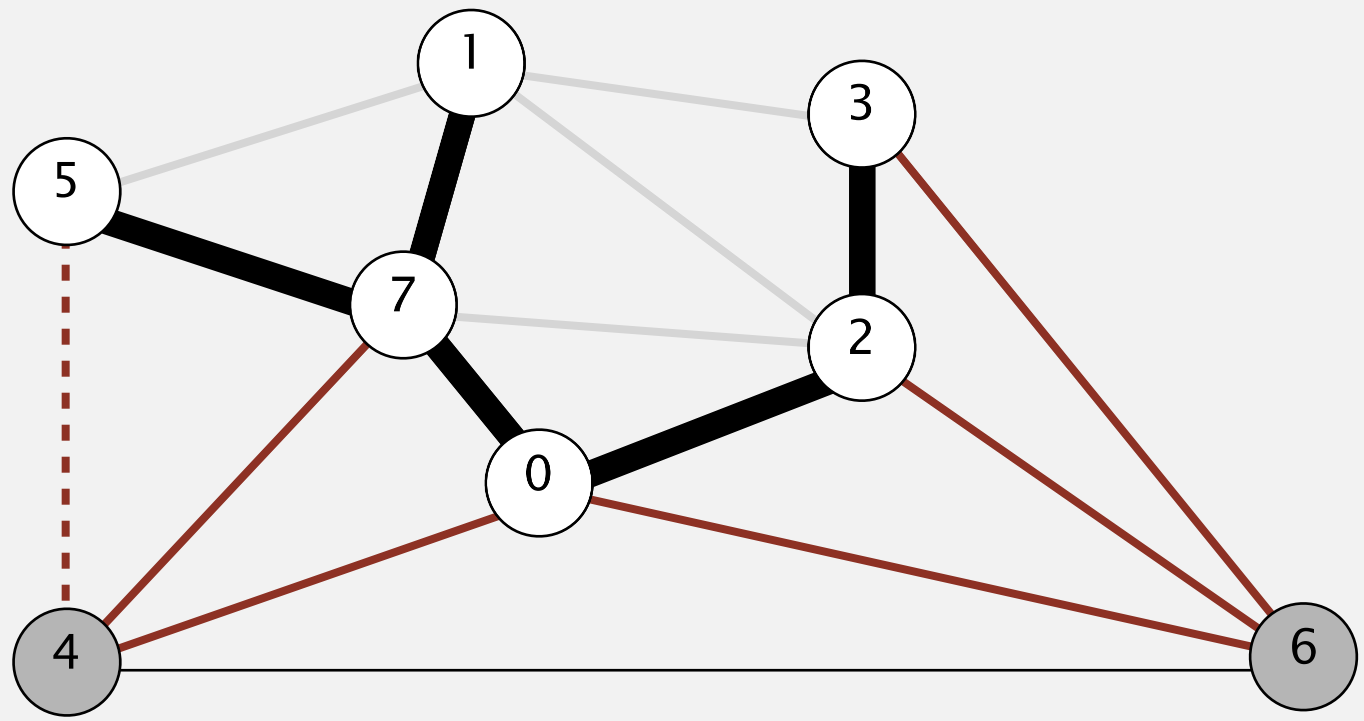
MST edges

0-7 1-7 0-2 2-3 5-7

Prim's algorithm: lazy implementation 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.

add to PQ all edges incident to 5



edges on PQ
(sorted by weight)

	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

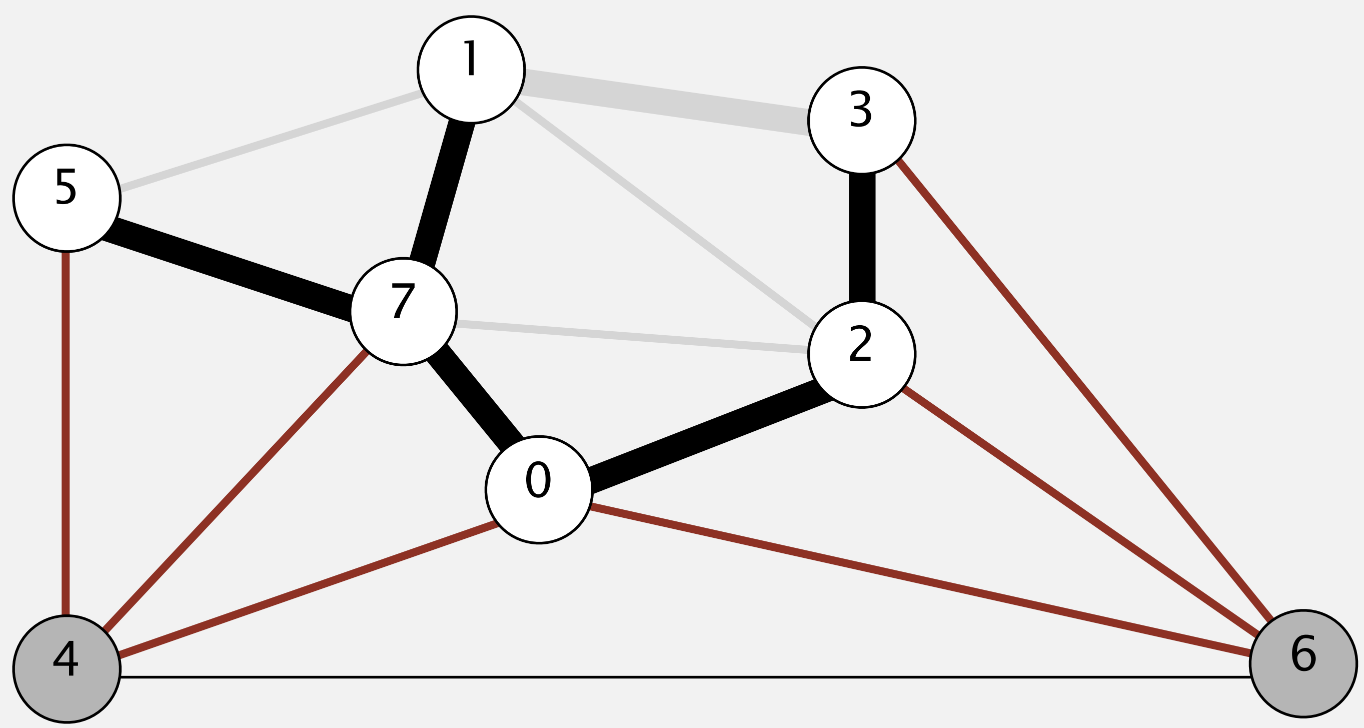
MST edges

0-7 1-7 0-2 2-3 5-7

Prim's algorithm: lazy implementation 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.

delete 1–3 and discard obsolete edge



edges on PQ
(sorted by weight)

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

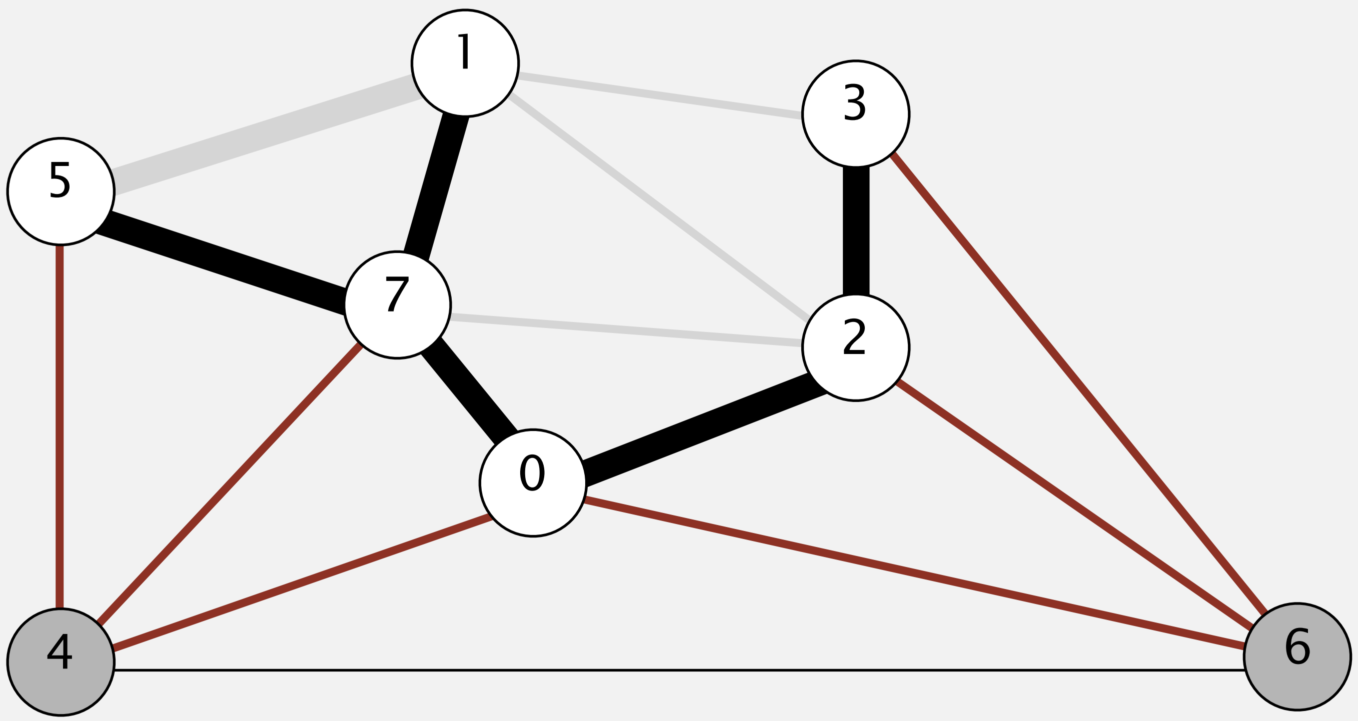
MST edges

0-7 1-7 0-2 2-3 5-7

Prim's algorithm: lazy implementation 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.

delete 1–5 and discard obsolete edge



edges on PQ
(sorted by weight)

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

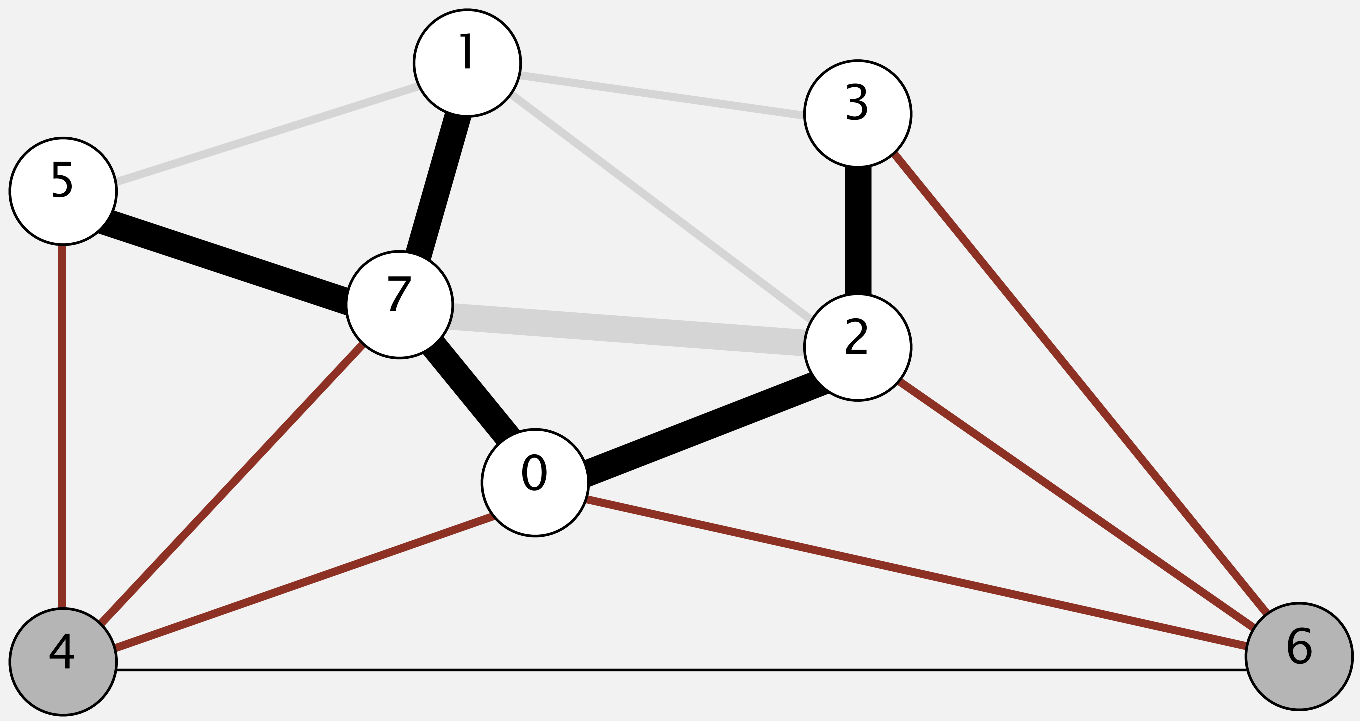
MST edges

0–7 1–7 0–2 2–3 5–7

Prim's algorithm: lazy implementation 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.

delete 2-7 and discard obsolete edge



edges on PQ
(sorted by weight)

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

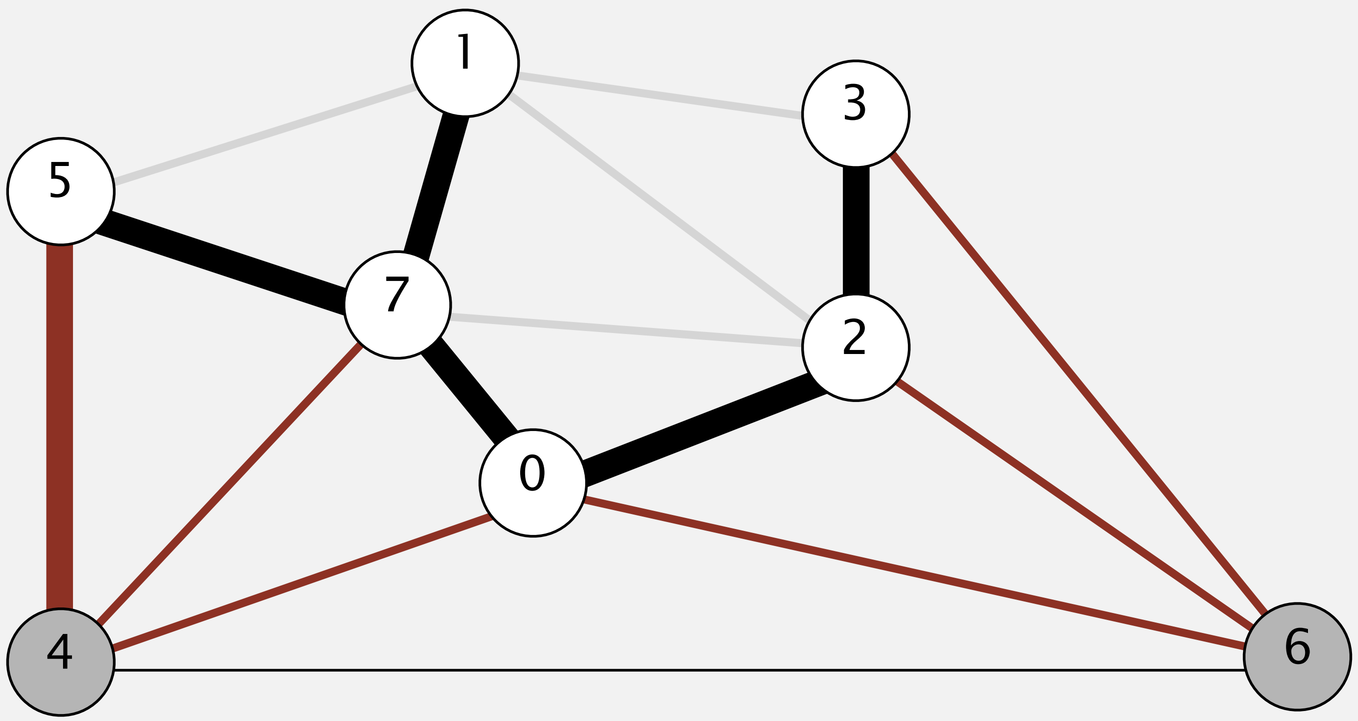
MST edges

0-7 1-7 0-2 2-3 5-7

Prim's algorithm: lazy implementation 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.

delete 4-5 and add to MST



edges on PQ
(sorted by weight)

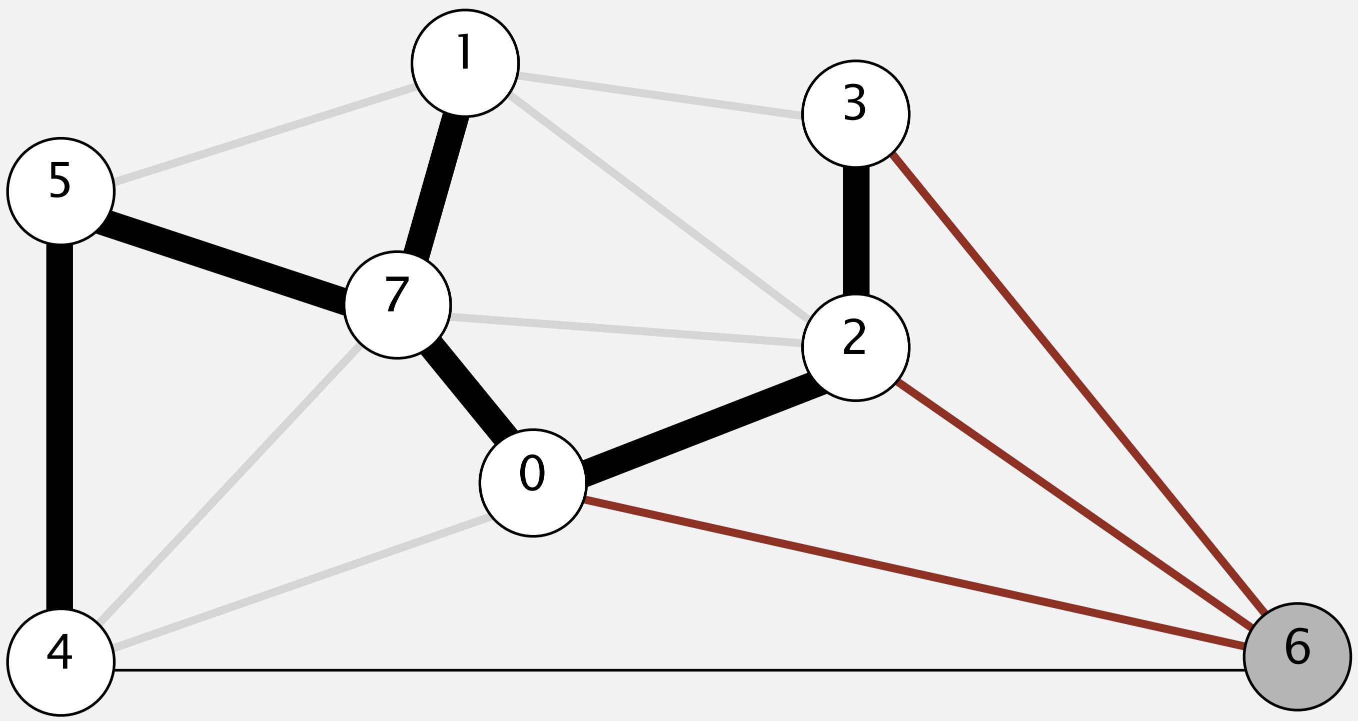
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

MST edges

0-7 1-7 0-2 2-3 5-7

Prim's algorithm: lazy implementation 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 on PQ
(sorted by weight)

1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

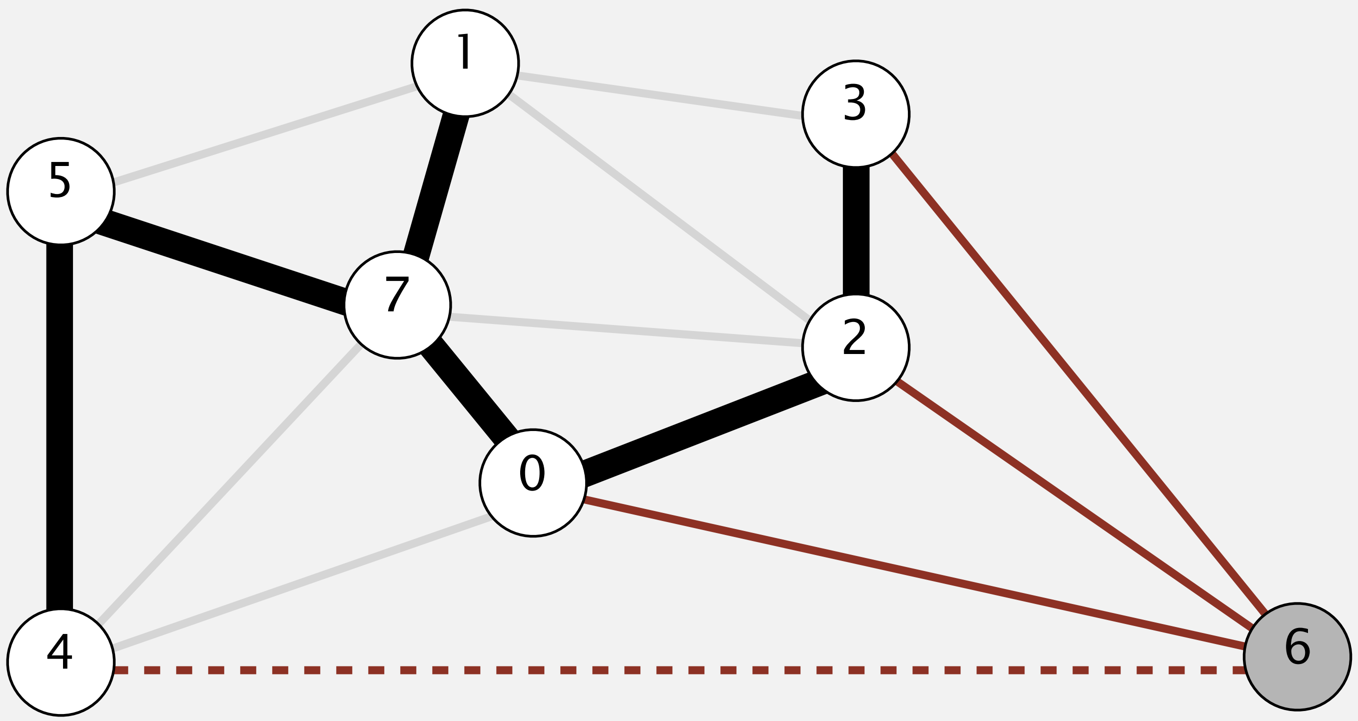
MST edges

0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm: lazy implementation 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.

add to PQ all edges incident to 4



edges on PQ
(sorted by weight)

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

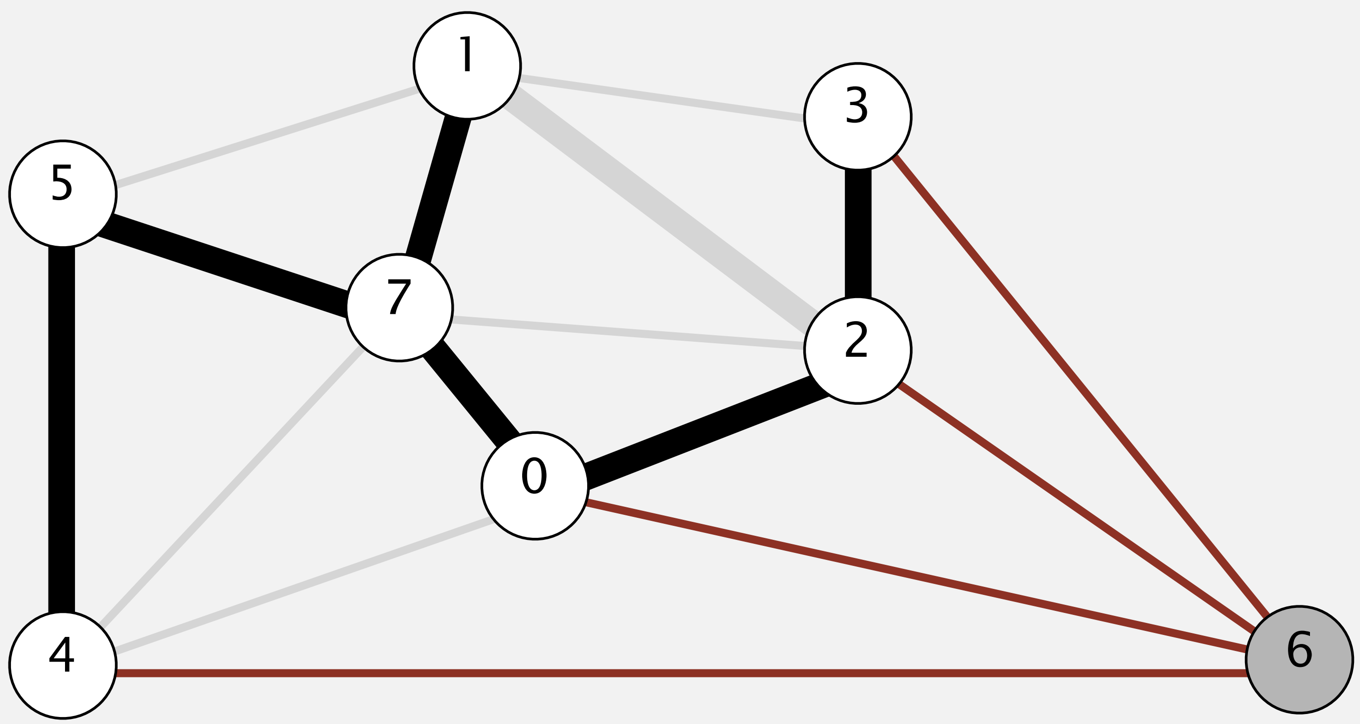
MST edges

0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm: lazy implementation 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.

delete 1–2 and discard obsolete edge



edges on PQ
(sorted by weight)

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

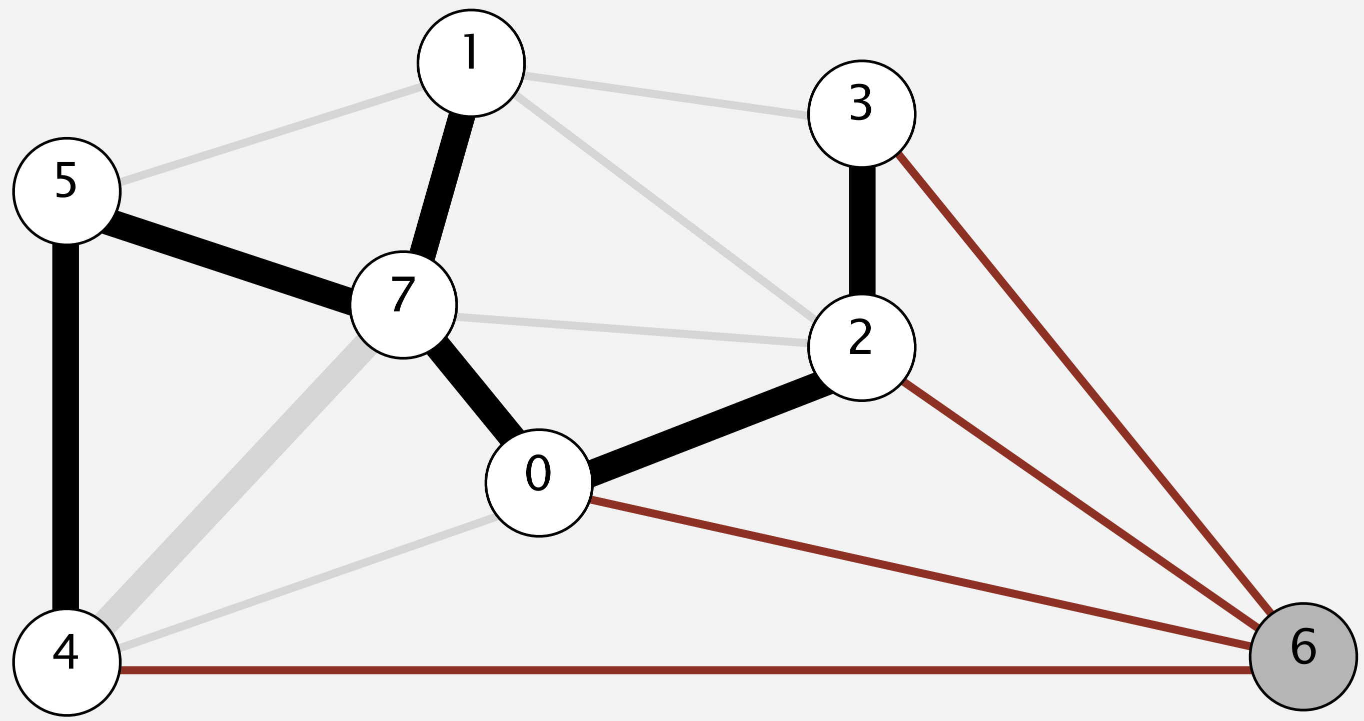
MST edges

0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm: lazy implementation 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.

delete 4–7 and discard obsolete edge



edges on PQ
(sorted by weight)

4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

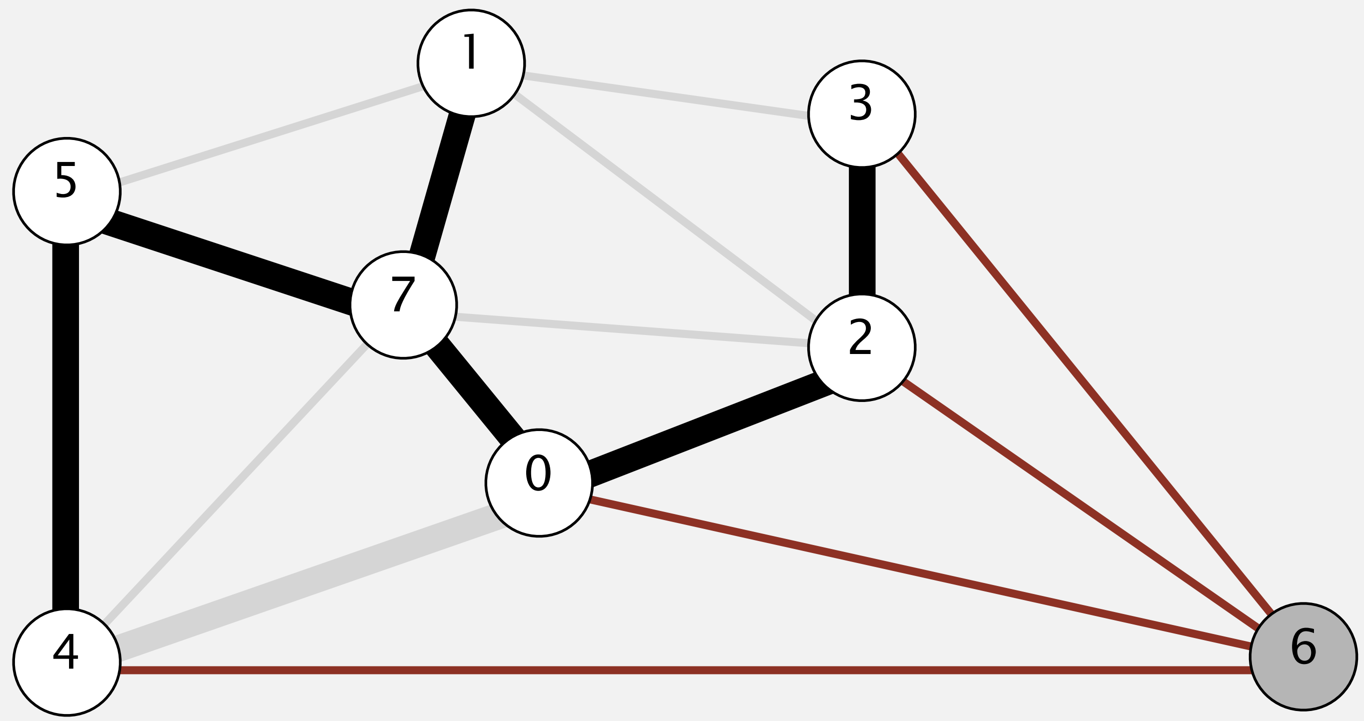
MST edges

0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm: lazy implementation 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.

delete 0–4 and discard obsolete edge



edges on PQ
(sorted by weight)

0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

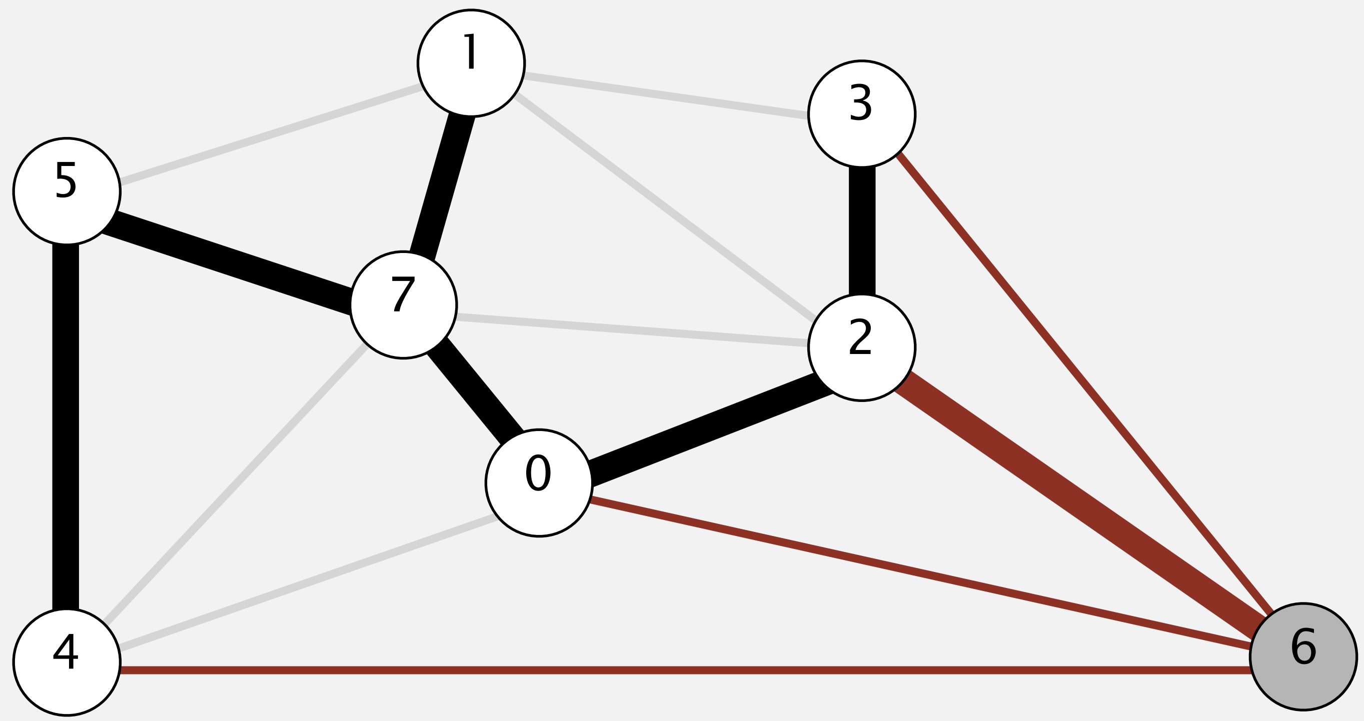
MST edges

0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm: lazy implementation 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.

delete 6-2 and add to MST



edges on PQ
(sorted by weight)

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

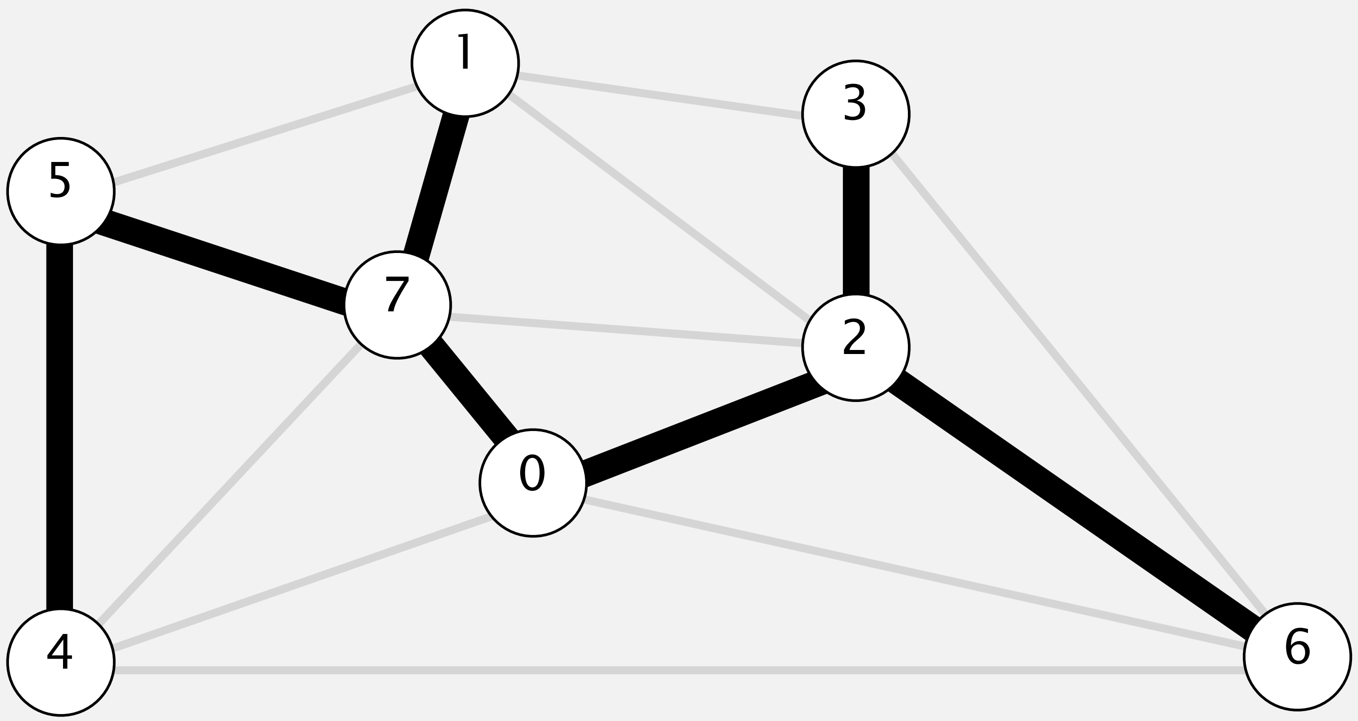
MST edges

0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm: lazy implementation 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.

delete 6–2 and add to MST



edges on PQ
(sorted by weight)

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

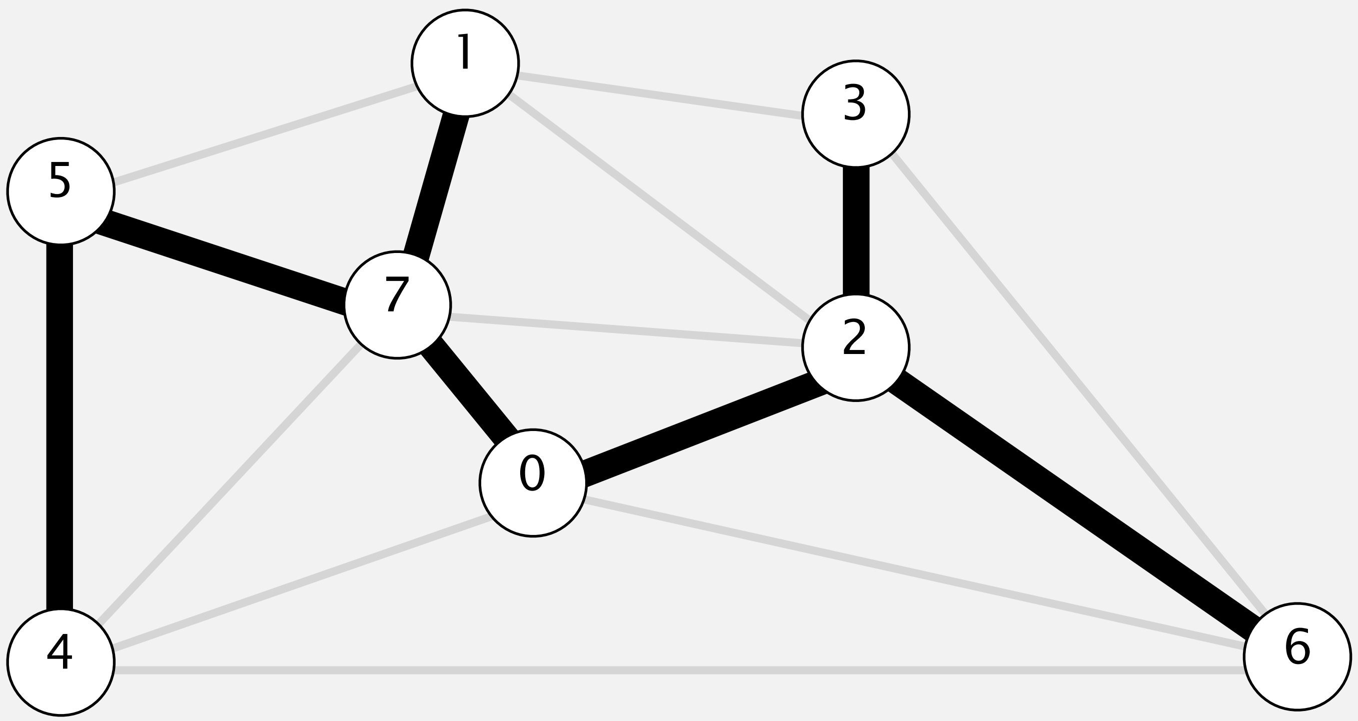
MST edges

0-7 1-7 0-2 2-3 5-7 4-5 6-2

Prim's algorithm: lazy implementation 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.

stop since V-1 edges



edges on PQ
(sorted by weight)

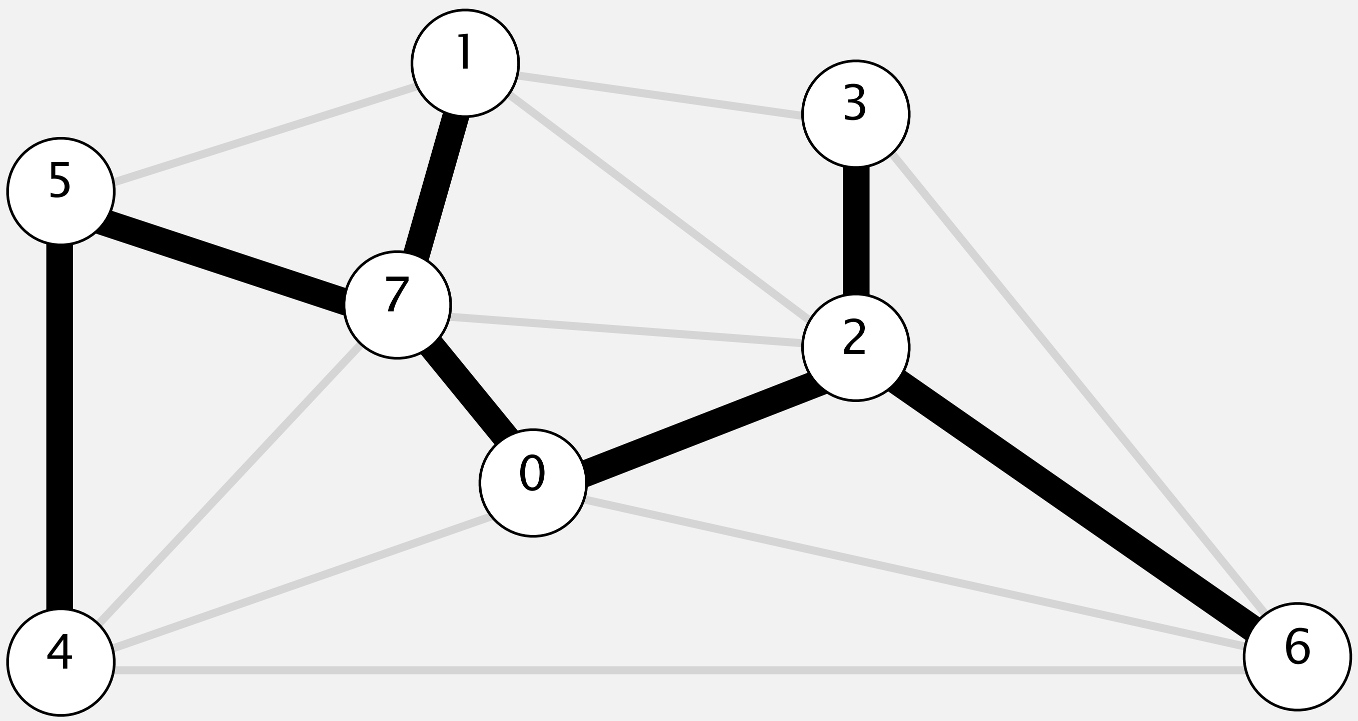
3-6	0.52
6-0	0.58
6-4	0.93

MST edges

0-7 1-7 0-2 2-3 5-7 4-5 6-2

Prim's algorithm: lazy implementation 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 6-2

Prim's algorithm: lazy implementation

```
public class LazyPrimMST
{
   private boolean[] marked;   // MST vertices
   private Queue<Edge> mst;    // MST edges
   private MinPQ<Edge> pq;     // PQ of edges

    public LazyPrimMST(WeightedGraph G)
    {
        pq = new MinPQ<Edge>();
        mst = new Queue<Edge>();
        marked = new boolean[G.V()];
        visit(G, 0);

        while (!pq.isEmpty() && mst.size() < G.V() - 1)
        {
           Edge e = pq.delMin();
           int v = e.either(), w = e.other(v);
           if (marked[v] && marked[w]) continue;
           mst.enqueue(e);
           if (!marked[v]) visit(G, v);
           if (!marked[w]) visit(G, w);
        }
   }
}
```

assume G is connected

repeatedly delete the min weight edge e = v–w from PQ

ignore if both endpoints in T

add edge e to tree

add v or w to tree

Prim's algorithm: lazy implementation

```
private void visit(WeightedGraph G, int v)
{
   marked[v] = true;
   for (Edge e : G.adj(v))
      if (!marked[e.other(v)])
         pq.insert(e);
}

public Iterable<Edge> mst()
{  return mst;  }
```

add v to T

for each edge e = v–w, add to PQ if w not already in T

Lazy Prim's algorithm: running time

Proposition. Lazy Prim's algorithm computes the MST in time proportional to $E \log E$ and extra space proportional to E (in the worst case).

Pf.

operation	frequency	binary heap
delete min	E	$\log E$
insert	E	$\log E$

NO ONE HAS FOUND A LINEAR TIME ALGORITHM FOR MST

Does a linear-time MST algorithm exist?

deterministic compare-based MST algorithms

year	worst case	discovered by
1975	$E \log \log V$	Yao
1976	$E \log \log V$	Cheriton-Tarjan
1984	$E \log^* V,\ E + V \log V$	Fredman-Tarjan
1986	$E \log (\log^* V)$	Gabow-Galil-Spencer-Tarjan
1997	$E\, \alpha(V) \log \alpha(V)$	Chazelle
2000	$E\, \alpha(V)$	Chazelle
20xx	E	???

log* -> number of times you Take the log to get to one

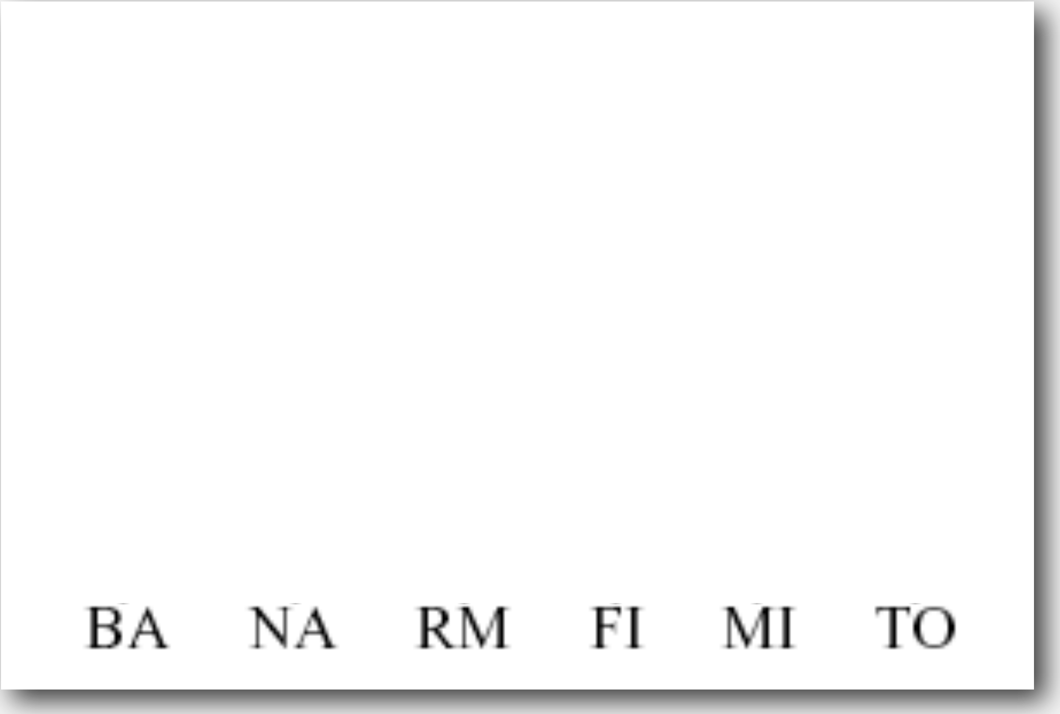
$\alpha(V)$ is a function that grows slower than log

Remark. Linear-time randomized MST algorithm (Karger-Klein-Tarjan 1995).

CLUSTERING USING PRIMS

Dendrogram

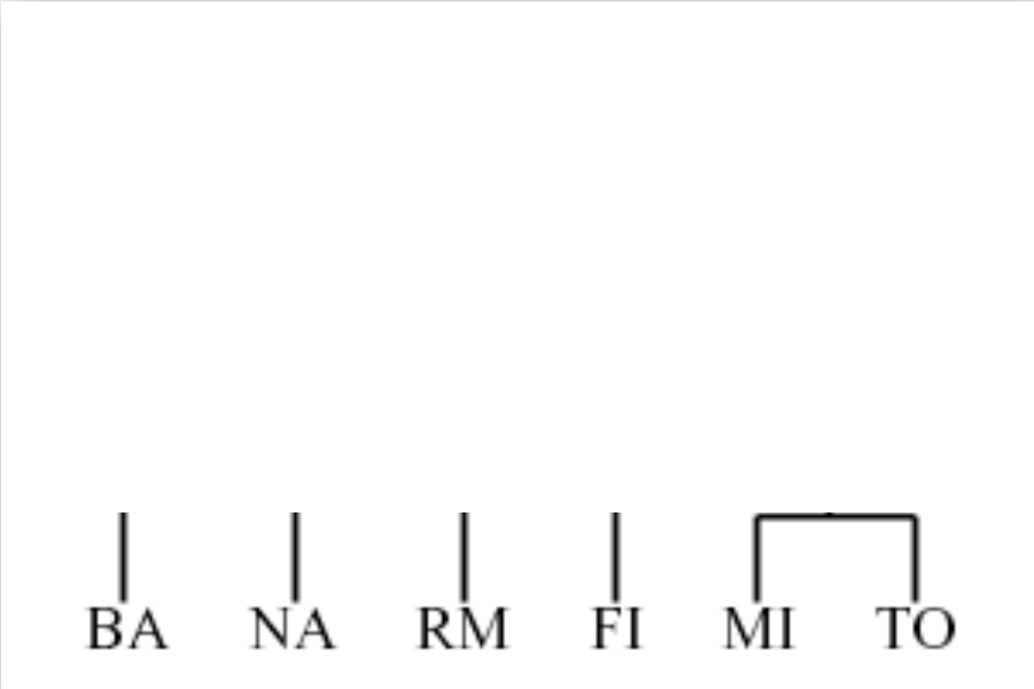
Dendrogram. Tree diagram that illustrates arrangement of clusters.



http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/hierarchical.html

Dendrogram

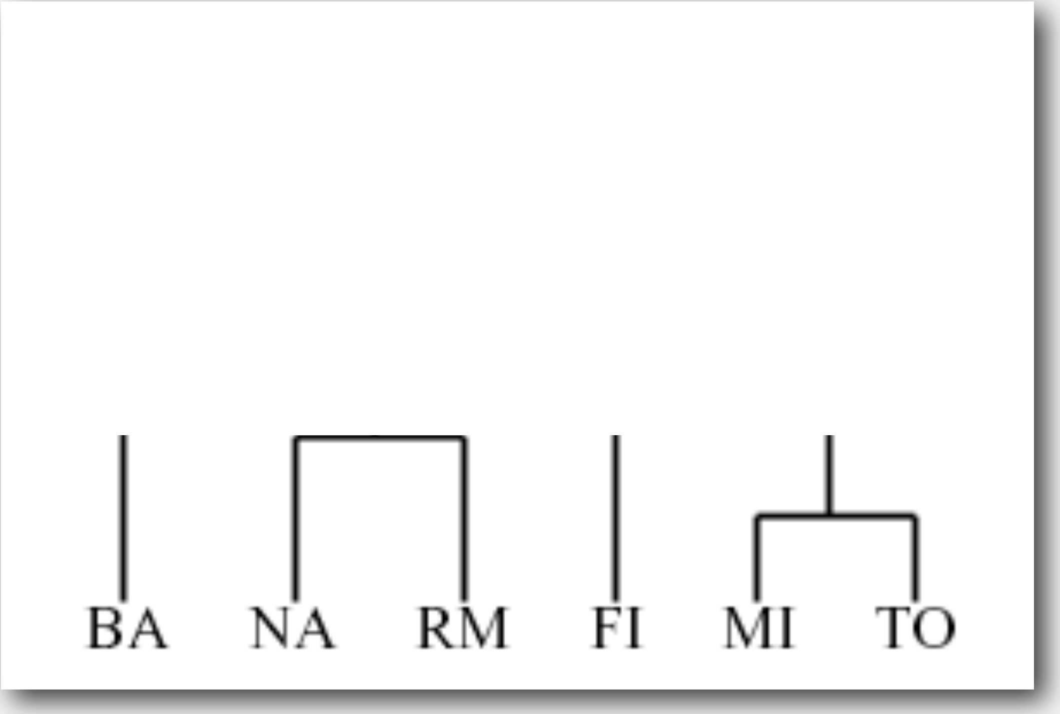
Dendrogram. Tree diagram that illustrates arrangement of clusters.



http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/hierarchical.html

Dendrogram

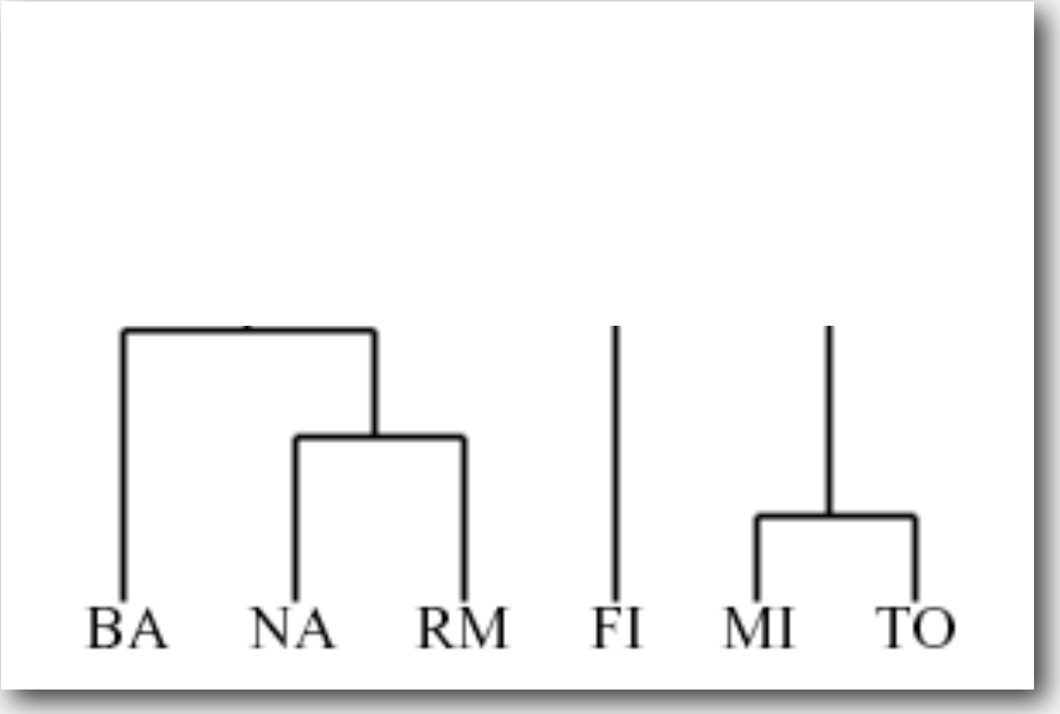
Dendrogram. Tree diagram that illustrates arrangement of clusters.



http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/hierarchical.html

Dendrogram

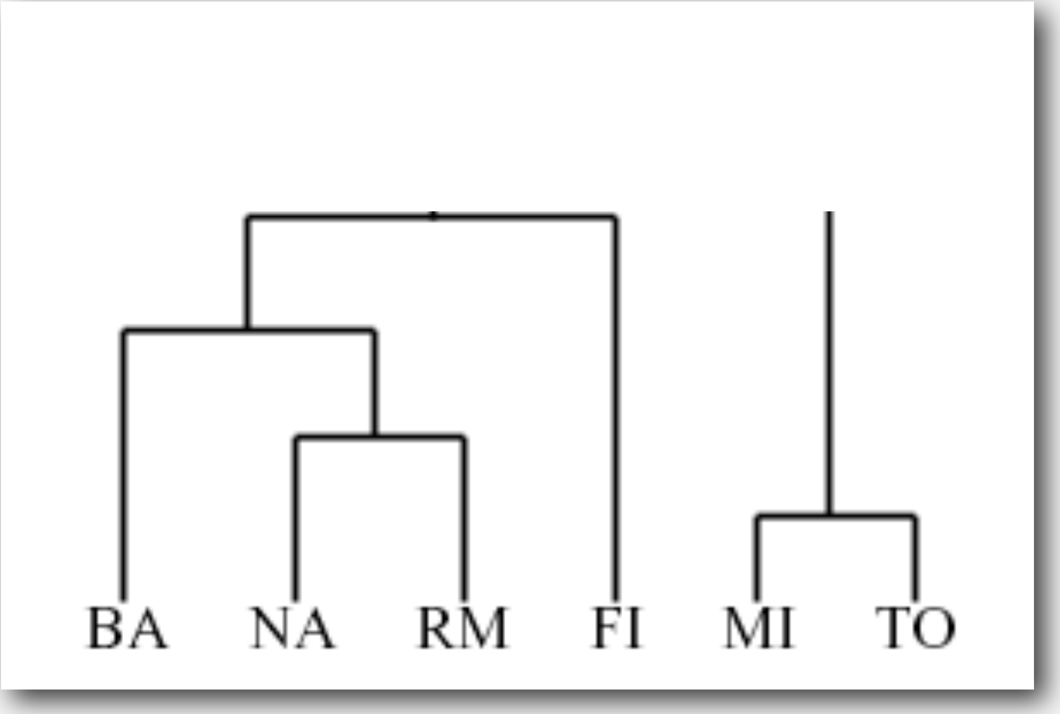
Dendrogram. Tree diagram that illustrates arrangement of clusters.



http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/hierarchical.html

Dendrogram

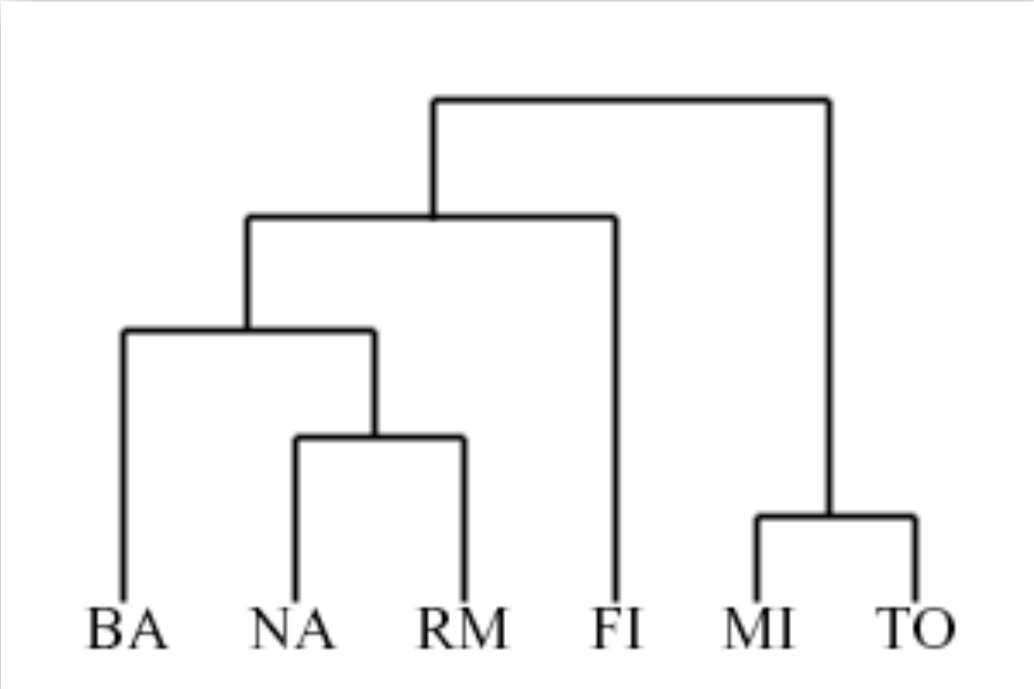
Dendrogram. Tree diagram that illustrates arrangement of clusters.



http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/hierarchical.html

Dendrogram

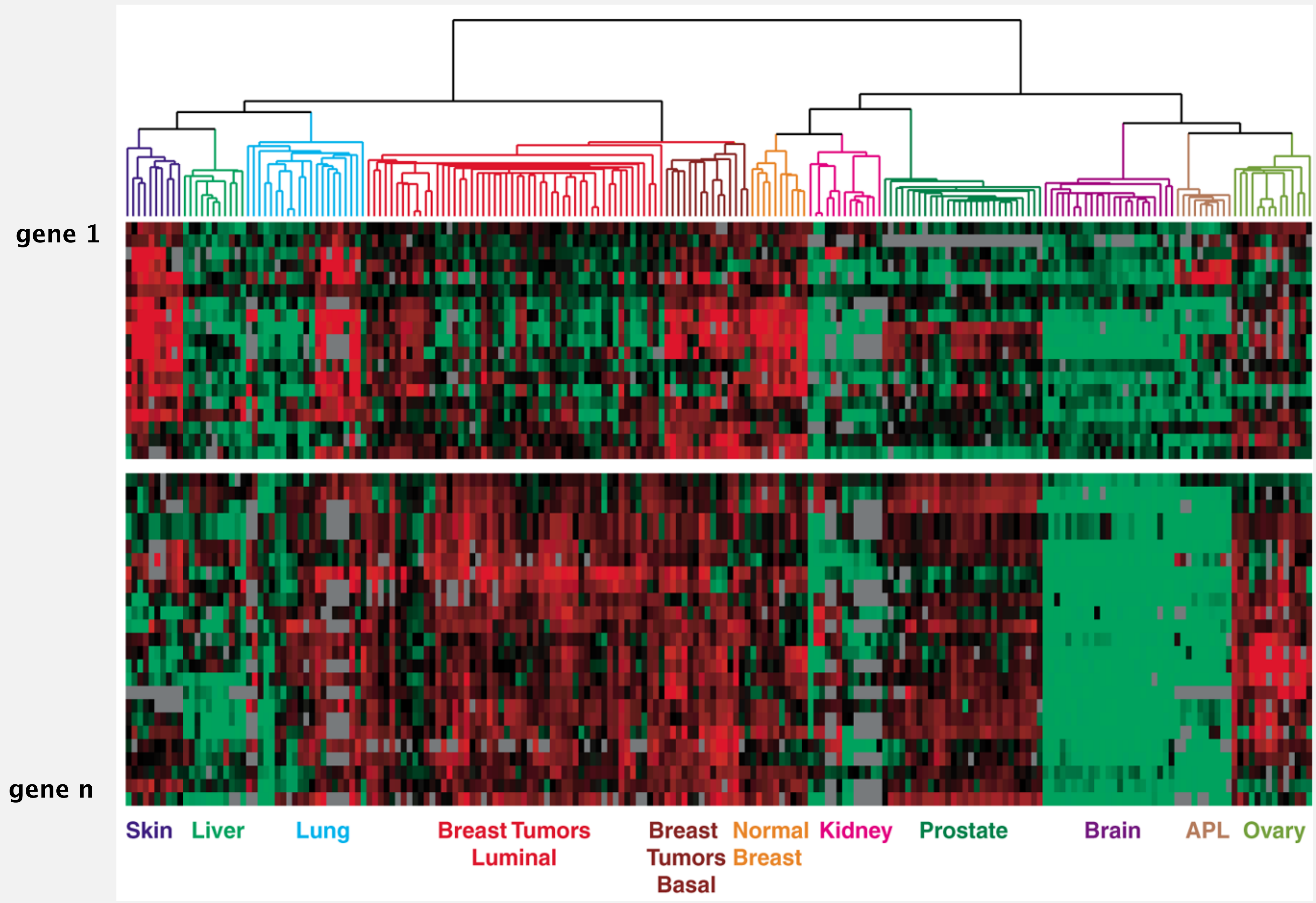
Dendrogram. Tree diagram that illustrates arrangement of clusters.



http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/hierarchical.html

Dendrogram of cancers in human

Tumors in similar tissues cluster together.



Reference: Botstein & Brown group

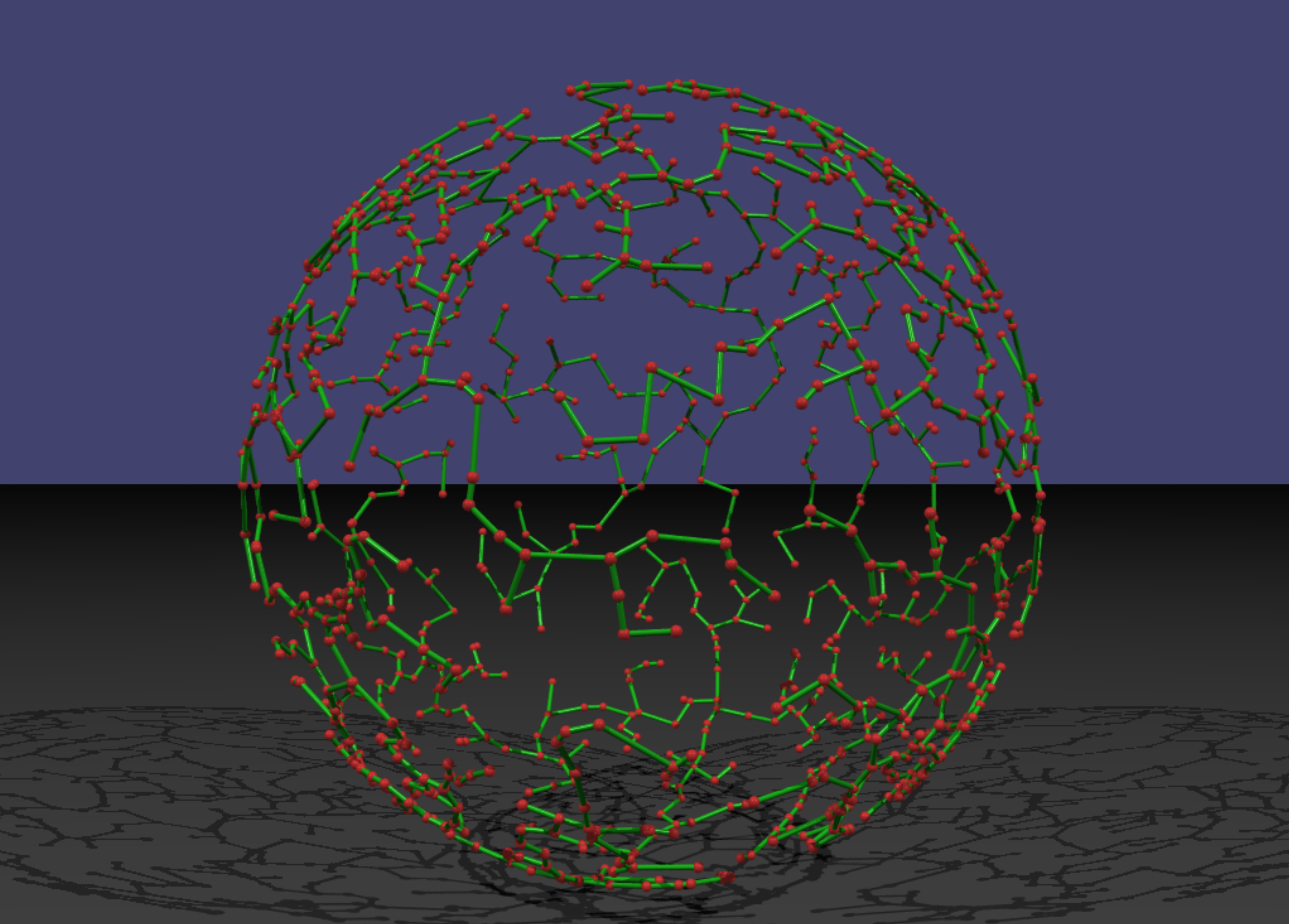
gene expressed

gene not expressed

http://algs4.cs.princeton.edu

4.3 Minimum Spanning

- *introduction*
- *greedy algorithm*
- *edge-weighted graph API*
- *Kruskal's algorithm*
- *Prim's algorithm*
- *context*



http://ginger.indstate.edu/ge/gfx