# INTRODUCTION TO ALGORITHMS
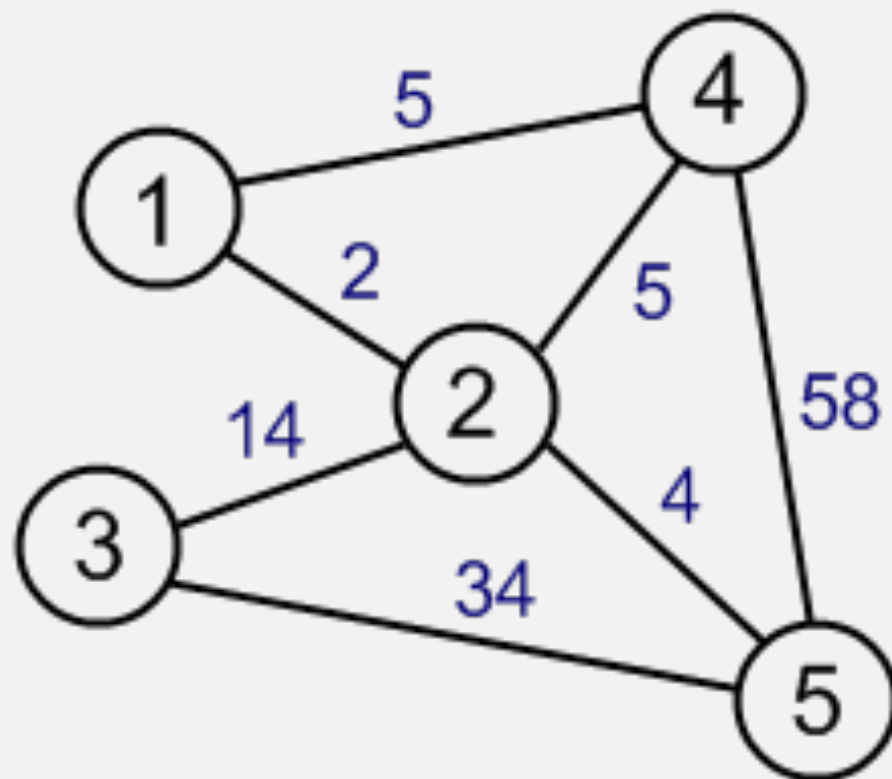
Lecture 10: Minimum Spanning Tree Algorithms

Yao-Chung Fan
yfan@nchu.edu.tw

# Minimum Spanning Trees

# Edge-Weighted Graph

# Weighted edge API

Edge abstraction needed for weighted edges.

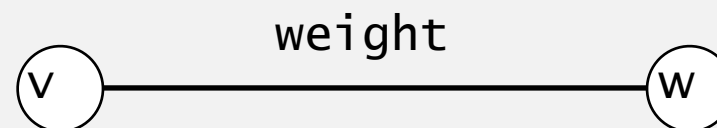| | public class Edge implements Comparable<Edge> | |
|---|---|---|
| | Edge(int v, int w, double weight) | *create a weighted edge v-w* |
| int | either() | *either endpoint* |
| int | other(int v) | *the endpoint that's not v* |
| int | compareTo(Edge that) | *compare this edge to that edge* |
| double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

```
        weight
  v ———————————————— w
```

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

# Weighted edge:  Java implementation

```java
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;                                              ← constructor
      this.weight = weight;
   }

   public int either()
   { return v; }                                              ← either endpoint

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

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

5

# 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* |

Maintain vertex-indexed array of Edge lists.
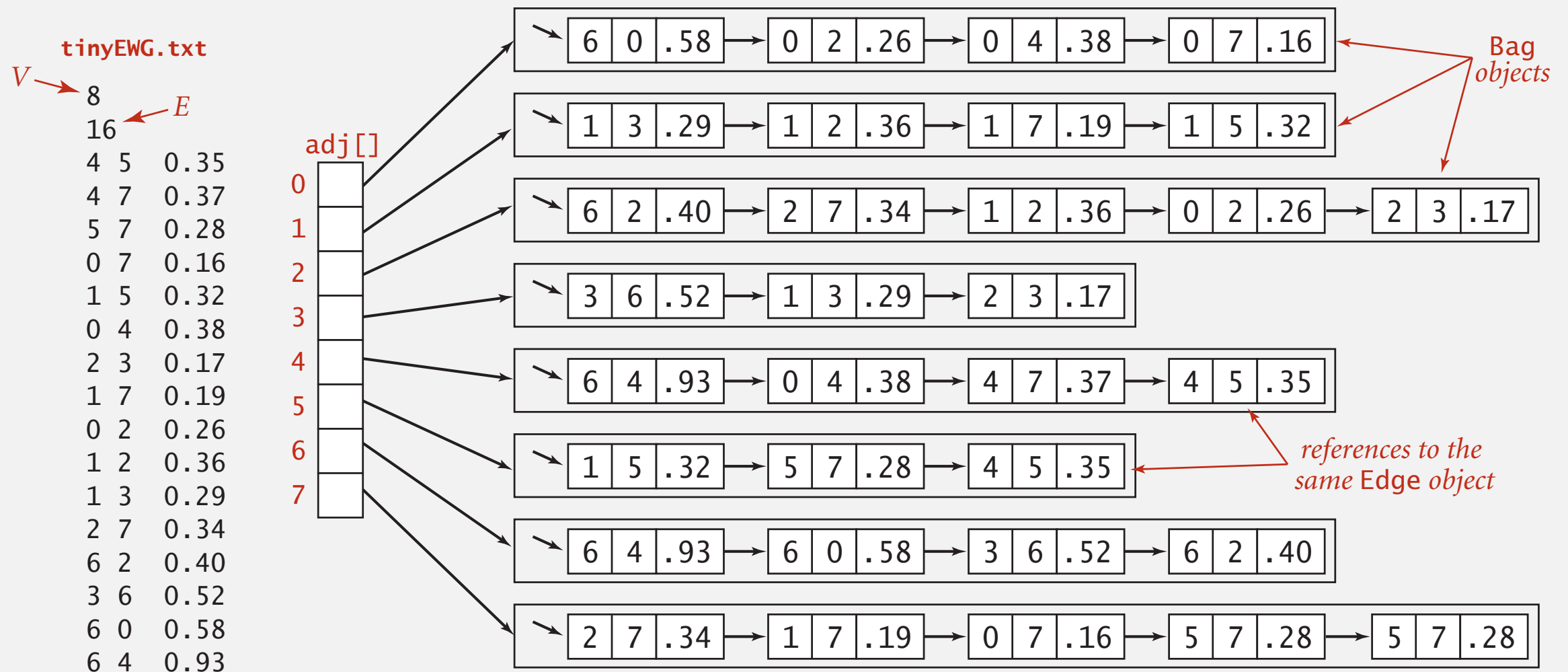
# Edge-weighted graph: adjacency-lists implementation

```java
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

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

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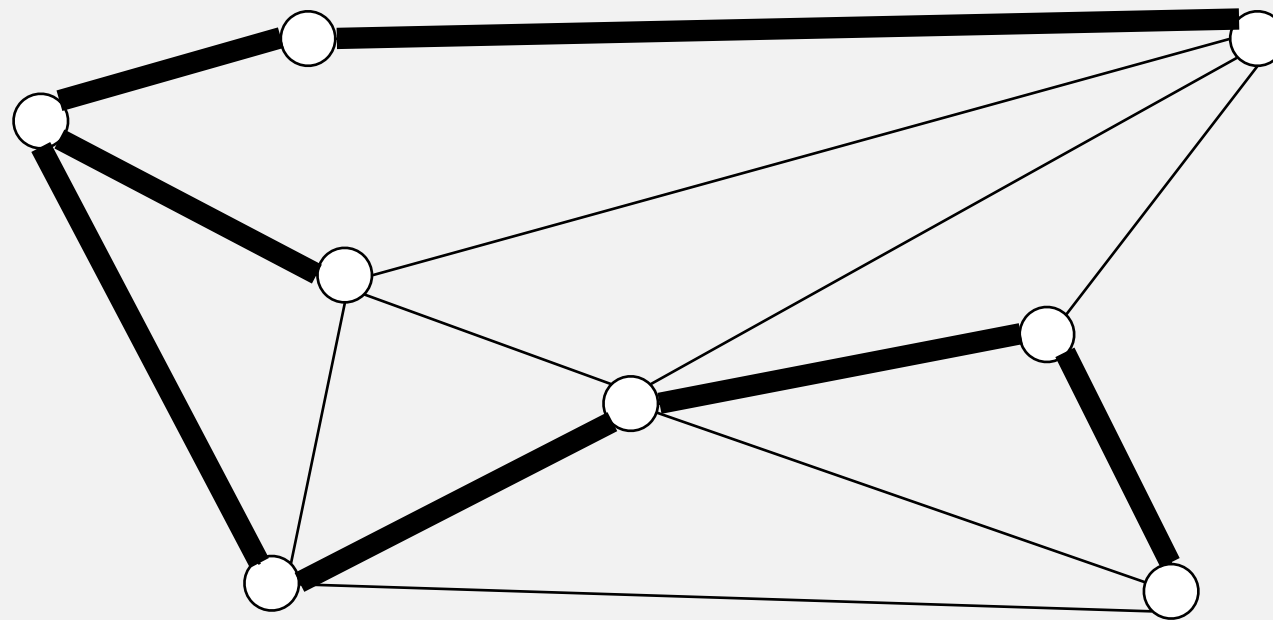


**graph G**

# Minimum spanning tree

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

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



**not connected**
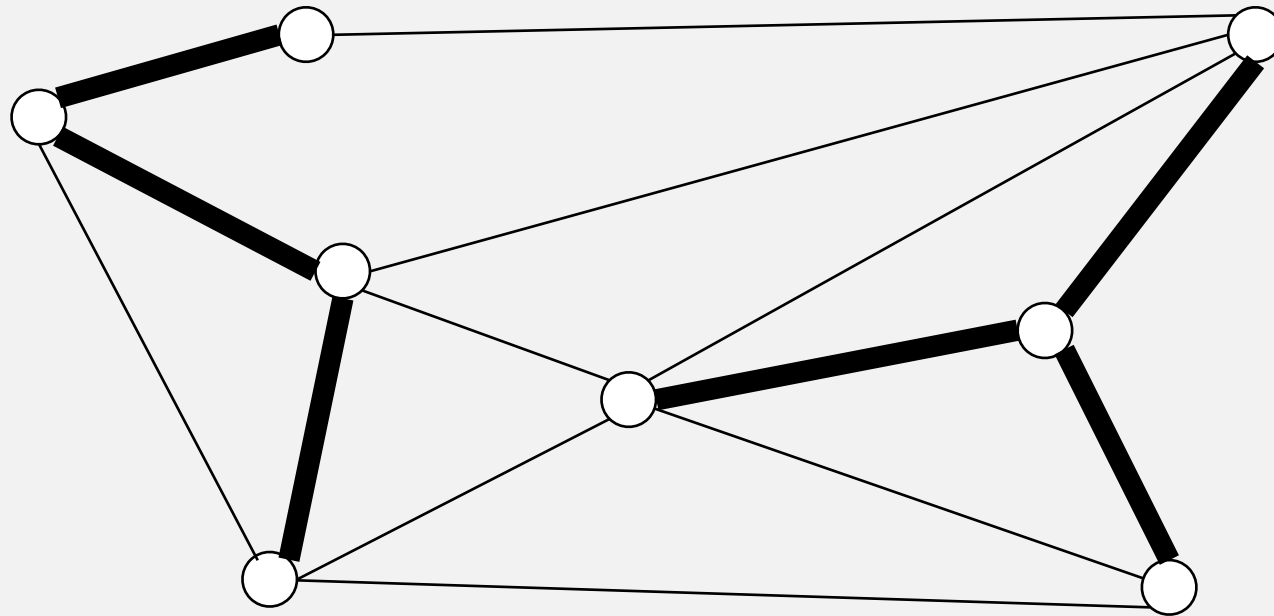
# Minimum spanning tree

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

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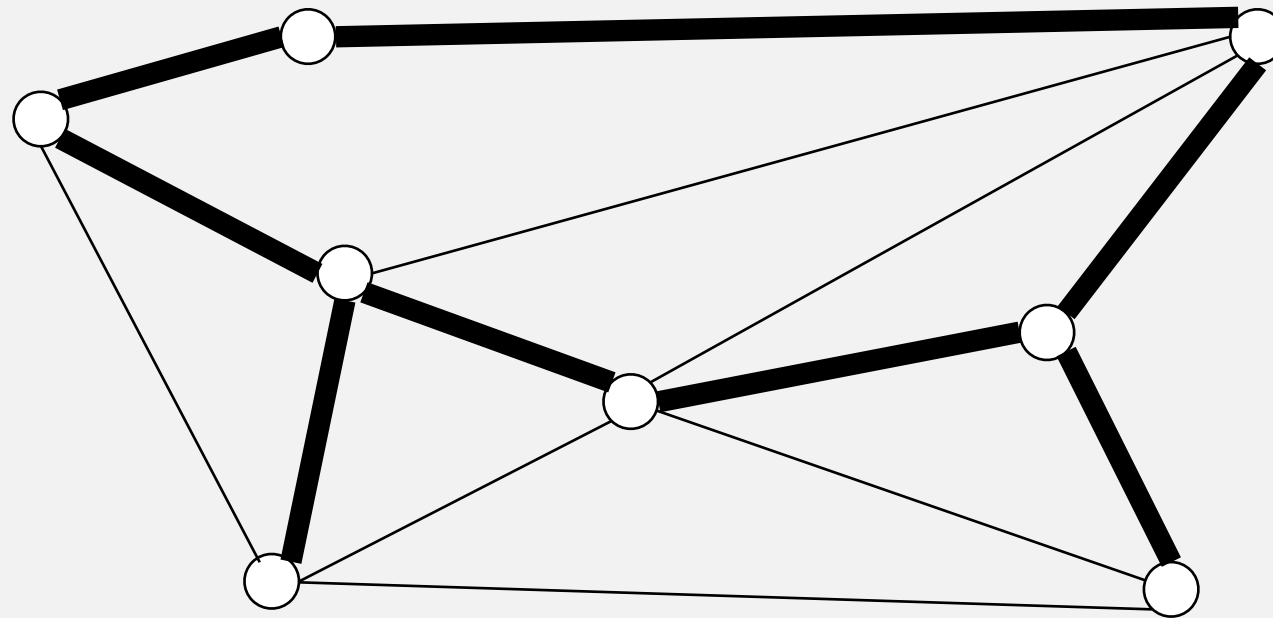- Acyclic.
- Includes all of the vertices.



**not acyclic**

# Minimum spanning tree

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

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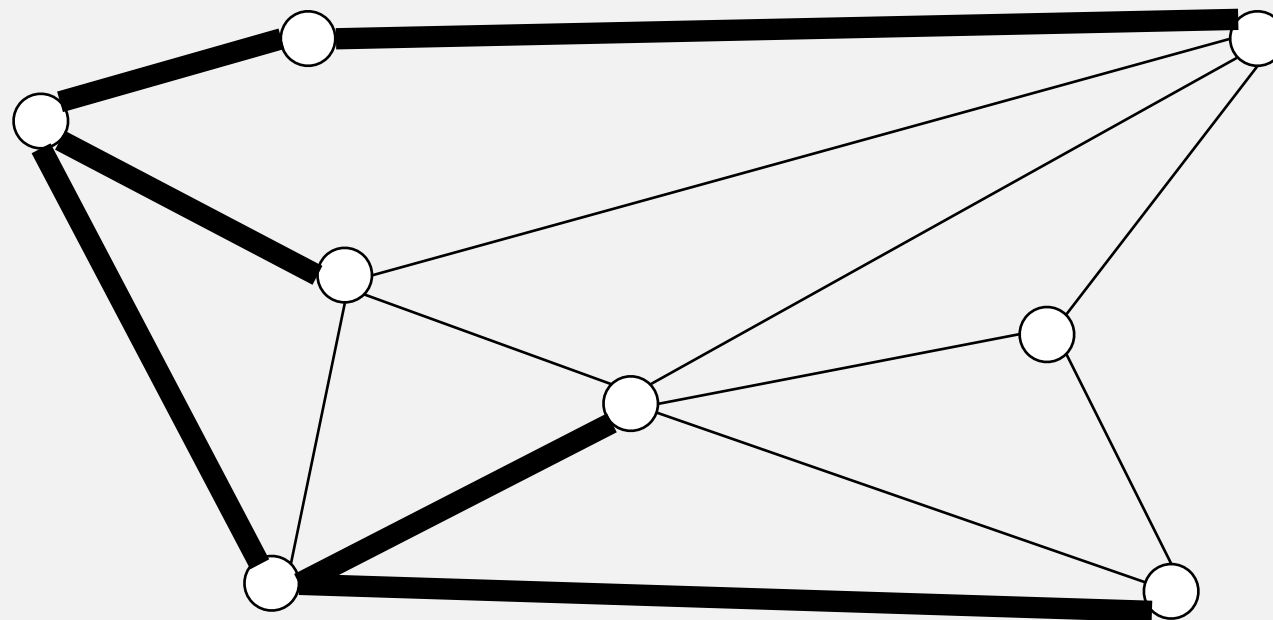


**not spanning**

# Minimum spanning tree problem

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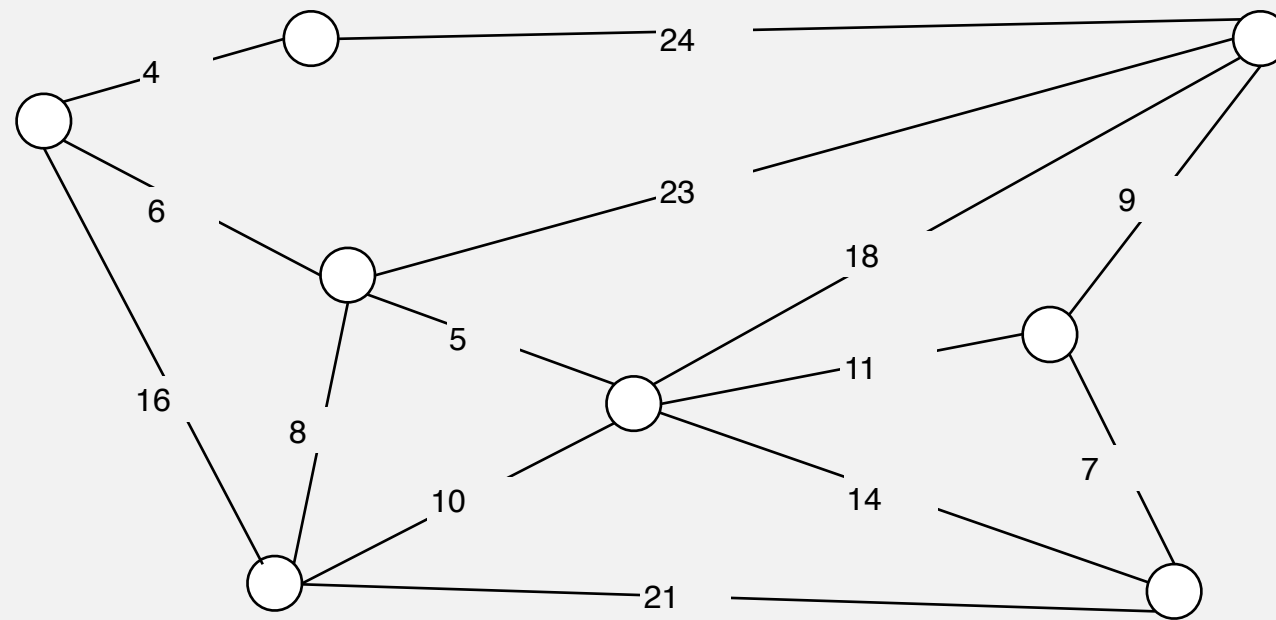


**edge-weighted graph G**

# 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?

Q. How to represent the MST?

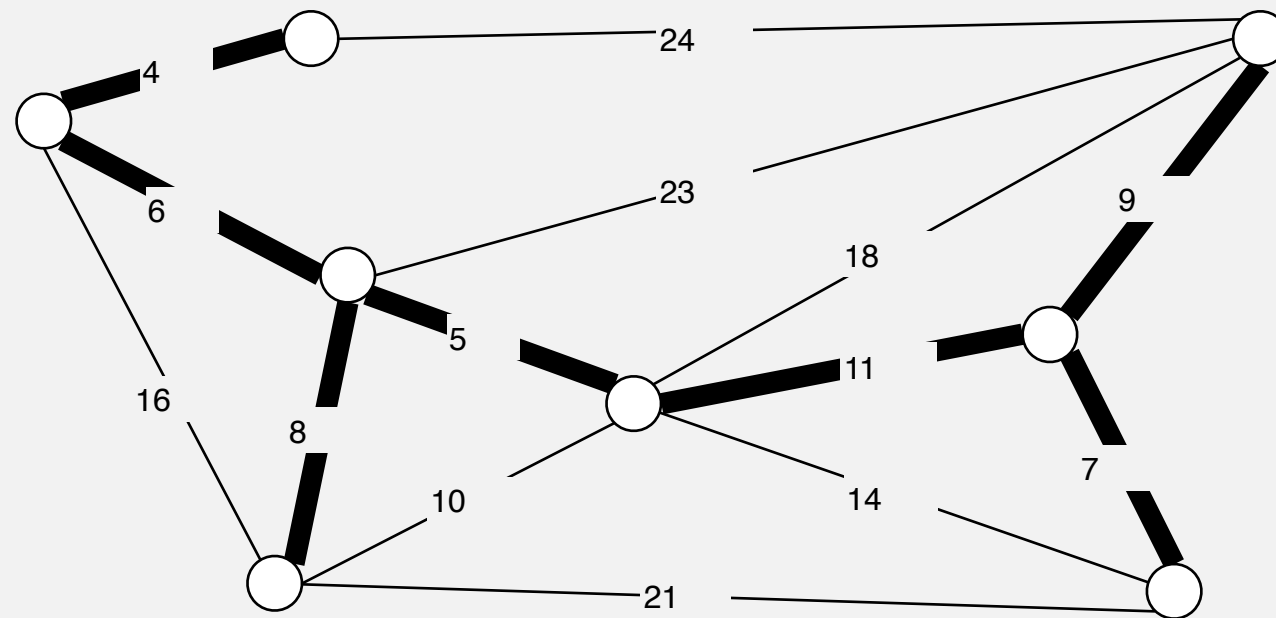| public class MST | | |
| --- | --- | --- |
| | MST(EdgeWeightedGraph G) | *constructor* |
| Iterable<Edge> | edges() | *edges in MST* |
| double | weight() | *weight of MST* |

**tinyEWG.txt**
$V$ → 8
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

*MST edge (black)*

*non-MST edge (gray)*



```
% 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
```
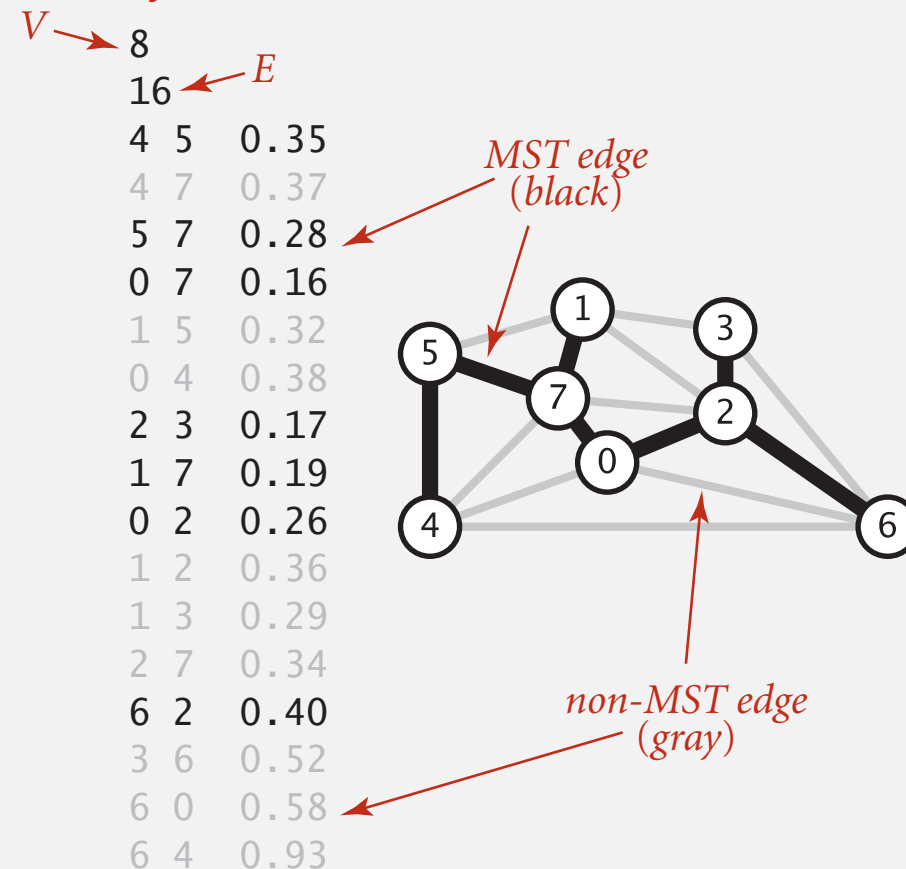
# Minimum spanning tree API

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
```

# Minimum Spanning Trees

# 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**

| | |
|---|---|
| 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 $\longrightarrow$  0-7  0.16



does not 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.



does not
create a cycle

0-7   0.16

in MST ⟶   2-3   0.17

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



```
0-7   0.16
2-3   0.17
1-7   0.19
```

in MST

# Kruskal's algorithm demo

Consider edges in ascending order of weight.

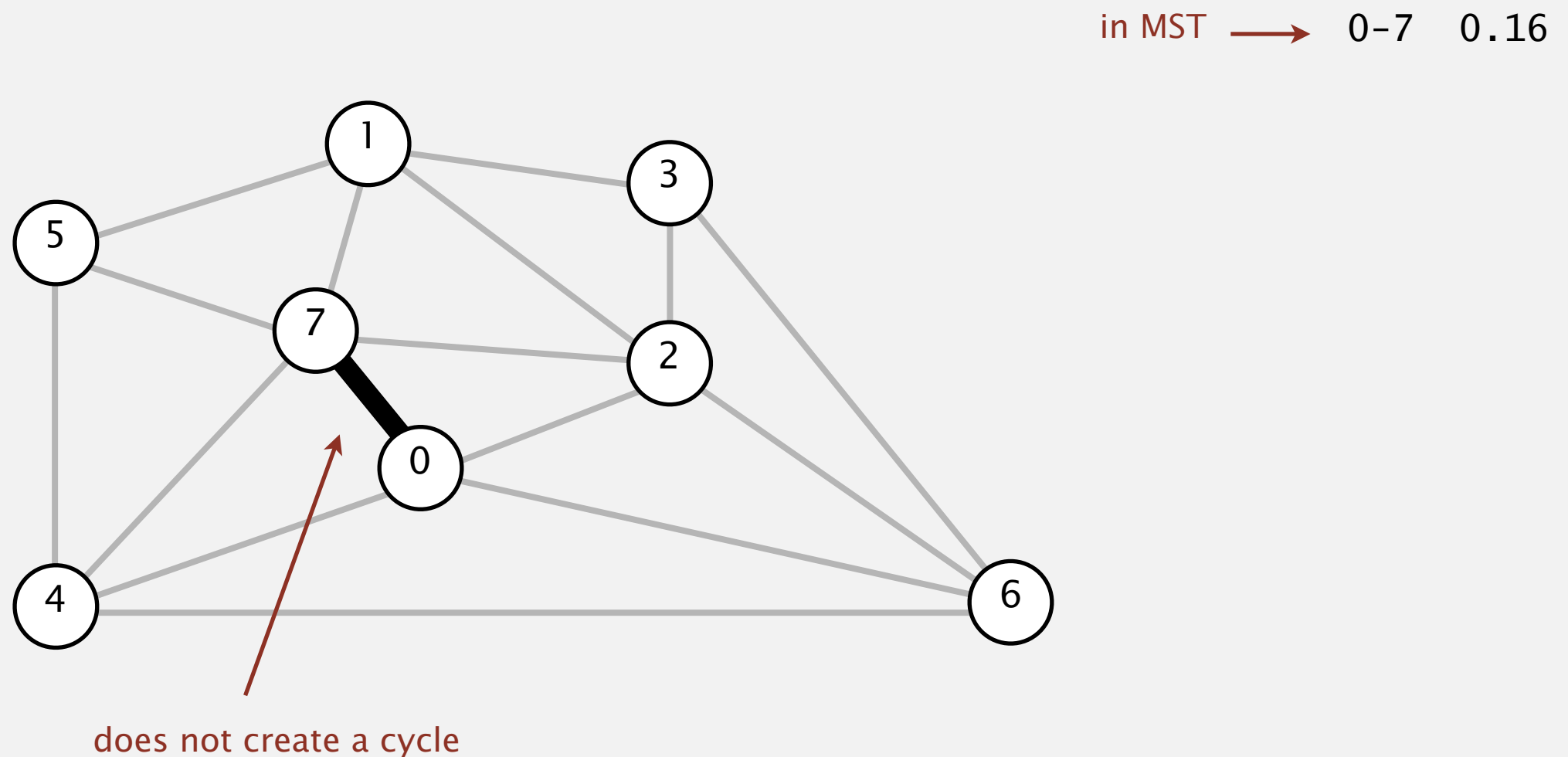• Add next edge to tree $T$ unless doing so would create a cycle.



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

does not 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
```

in MST $\longrightarrow$ 5-7   0.28

does not create a cycle

# Kruskal's algorithm demo

Consider edges in ascending order of weight.

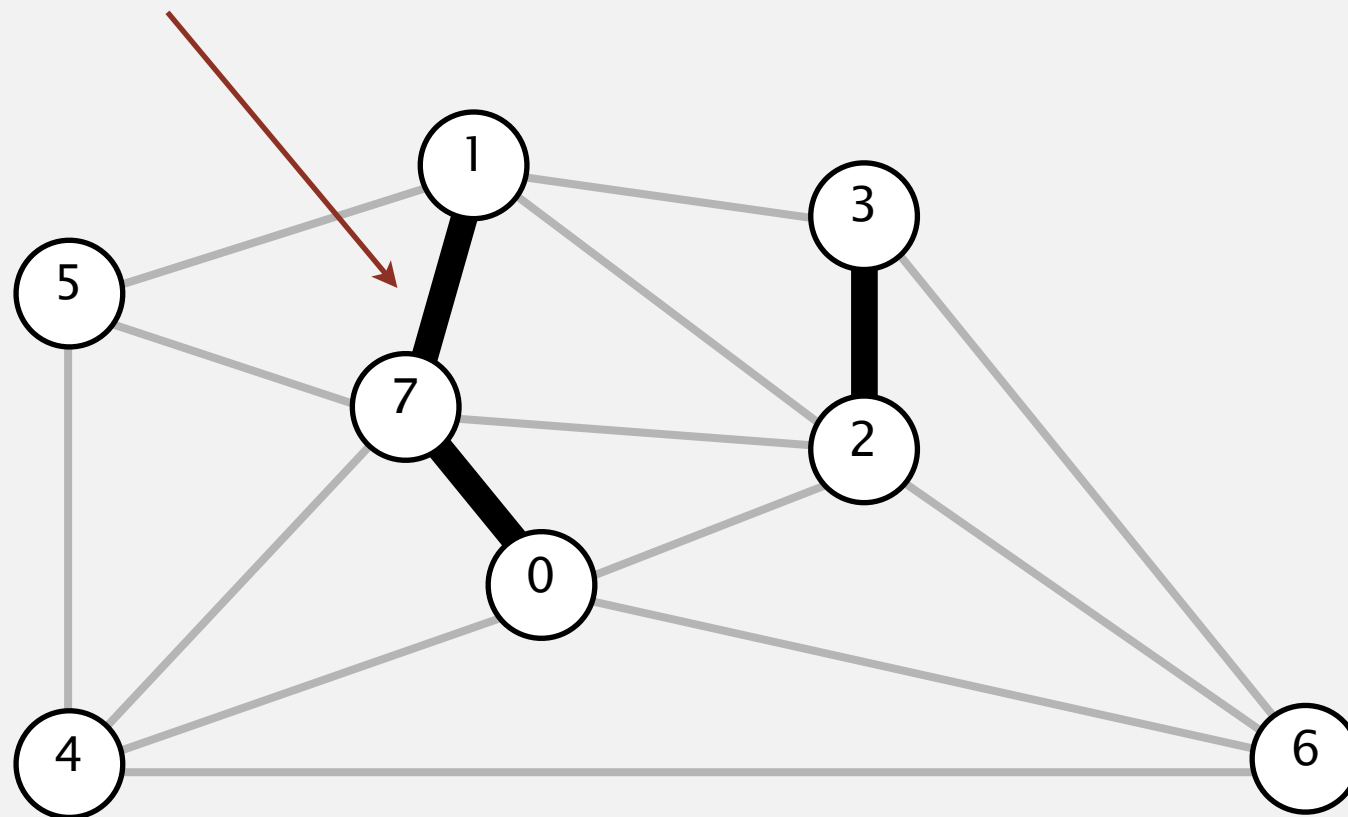- Add next edge to tree $T$ unless doing so would create a cycle.



creates a cycle

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

not in MST ⟶ `1-3   0.29`

# Kruskal's algorithm demo

Consider edges in ascending order of weight.

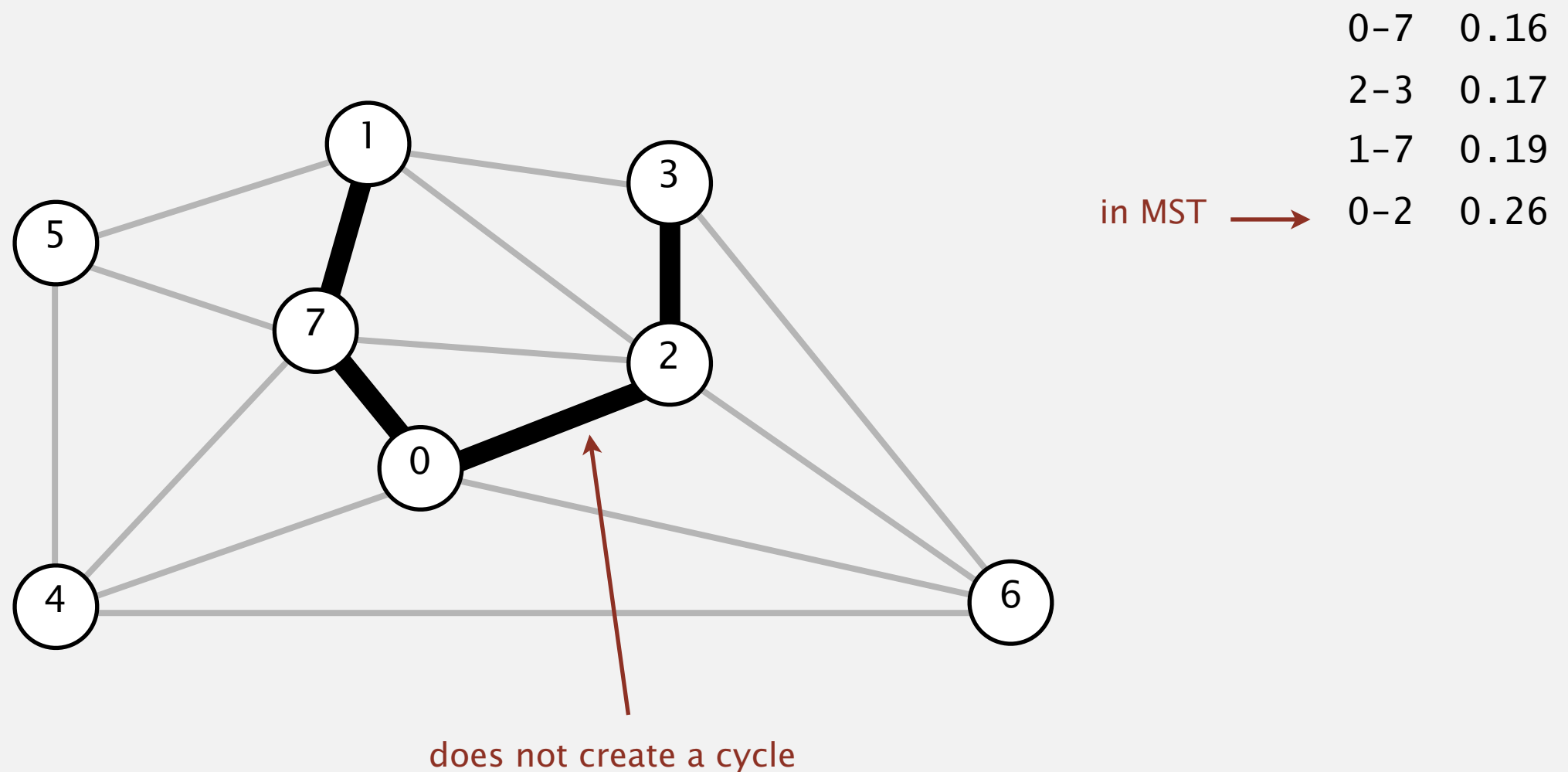- Add next edge to tree $T$ unless doing so would create a cycle.



creates 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
```

Consider edges in ascending order of weight.

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



creates 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
```

# 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
```

in MST ⟶

does not 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.



creates 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 |

# Kruskal's algorithm demo

Consider edges in ascending order of weight.

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



creates 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
```

creates a cycle

not in MST ⟶

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

| | |
|---|---|
| 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 |

in MST →

# Kruskal's algorithm demo

Consider edges in ascending order of weight.

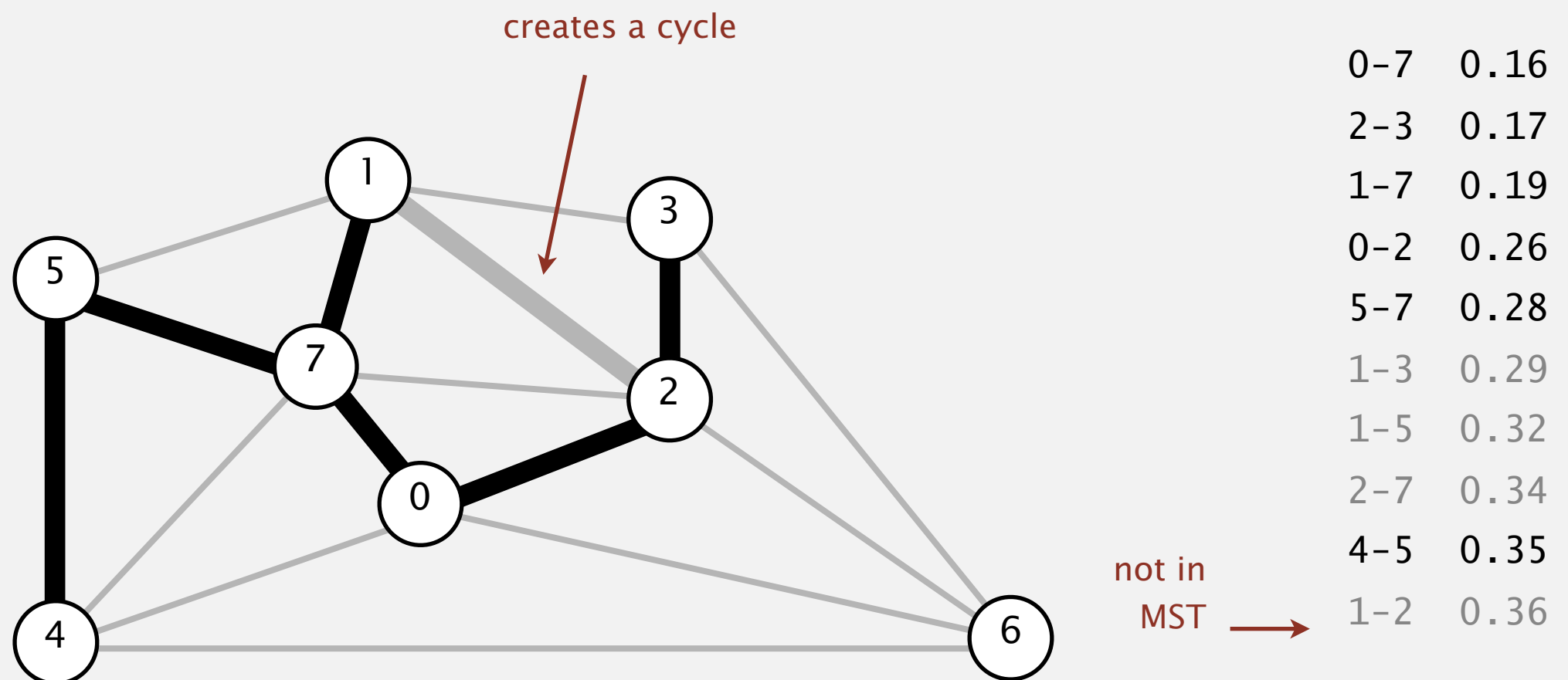- Add next edge to tree $T$ unless doing so would create a cycle.



creates 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.



creates 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
0-4  0.38
6-2  0.40
3-6  0.52
6-0  0.58
```

# Kruskal's algorithm demo

Consider edges in ascending order of weight.

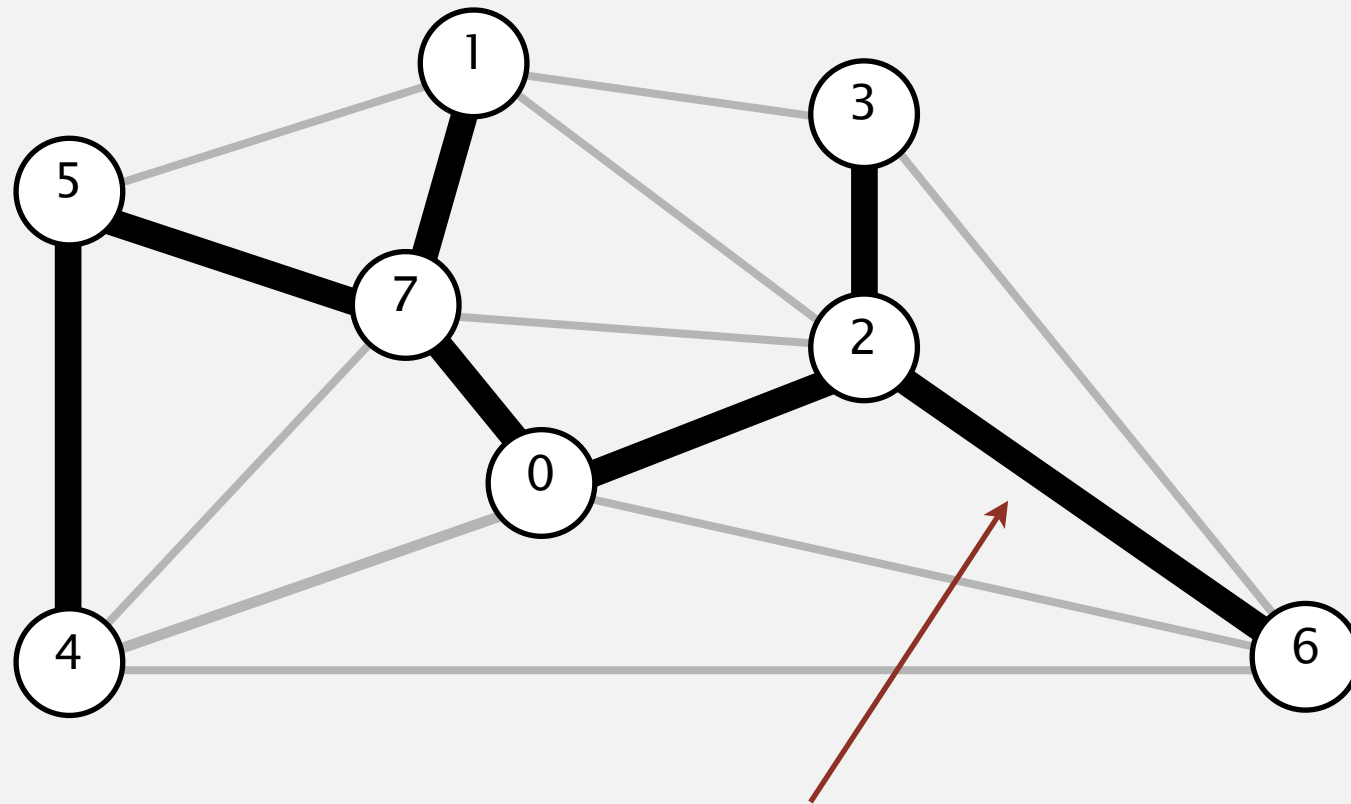• Add next edge to tree $T$ unless doing so would create a cycle.

| | |
|---|---|
| 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 |

creates a cycle

not in MST ⟶

34

# Kruskal's algorithm demo

Consider edges in ascending order of weight.

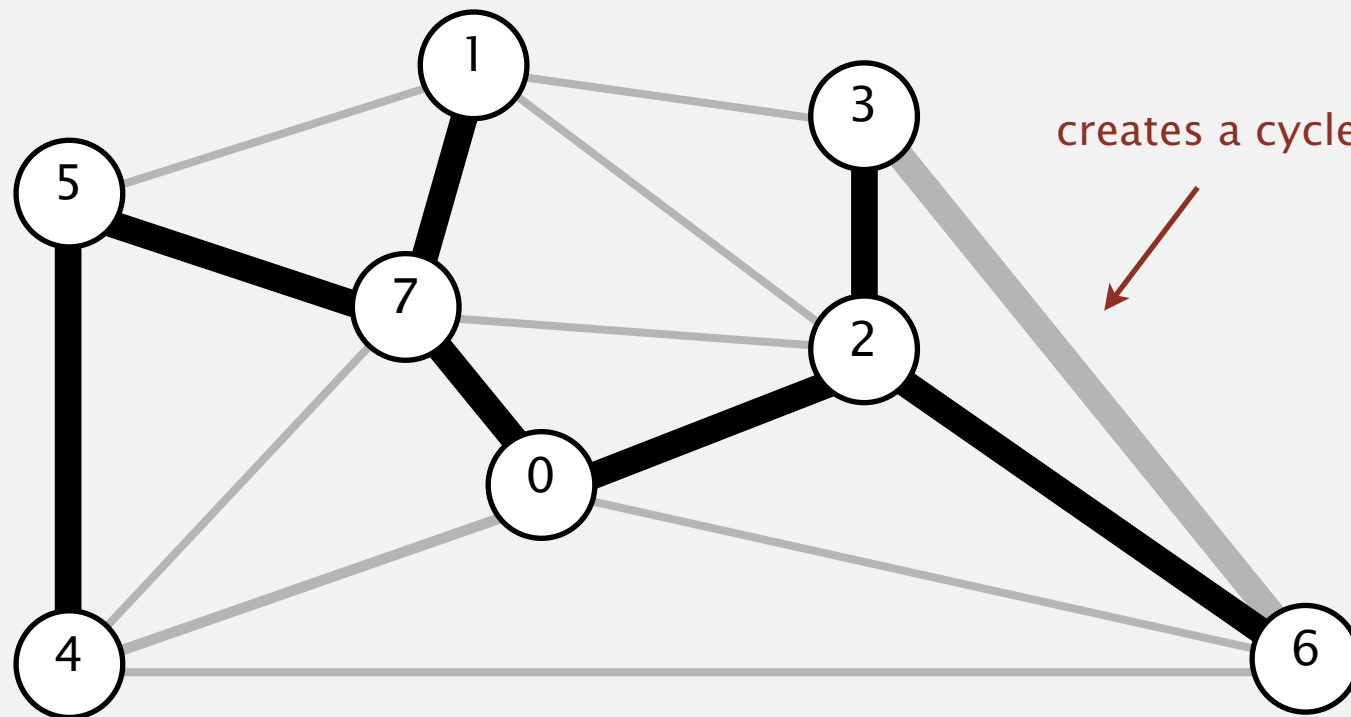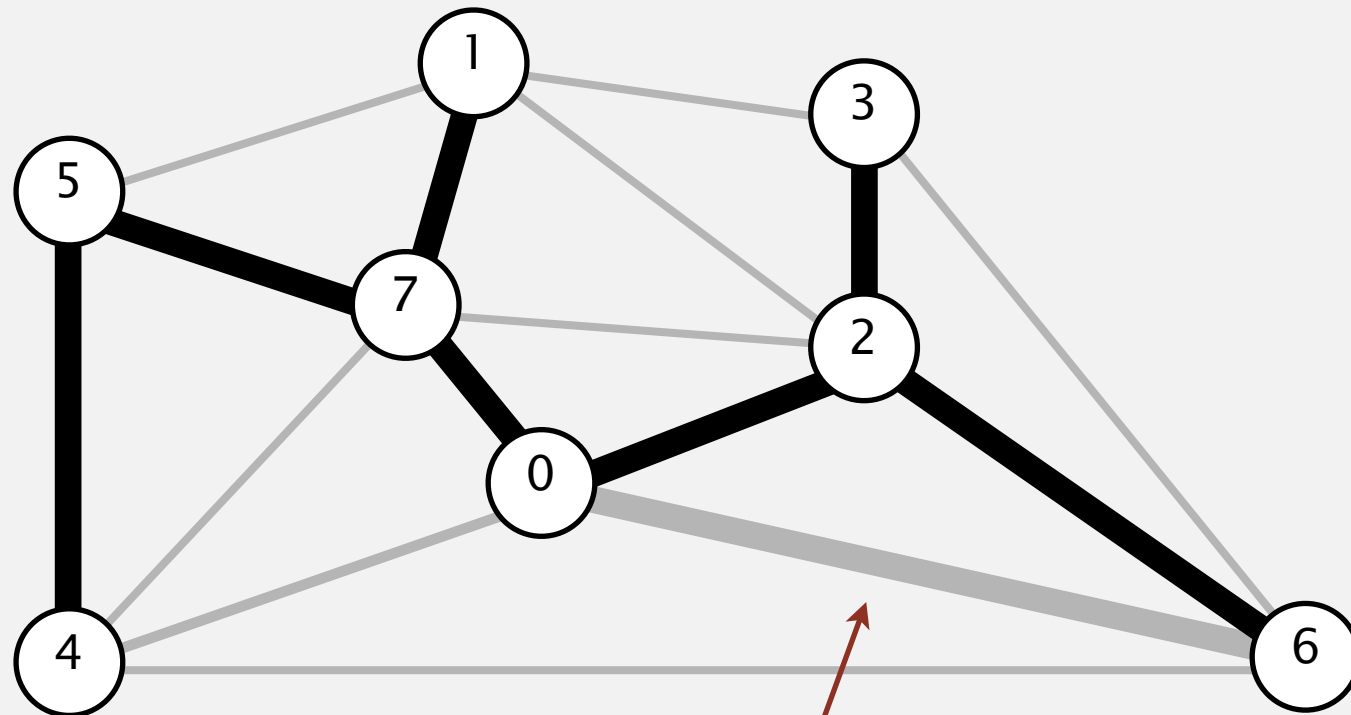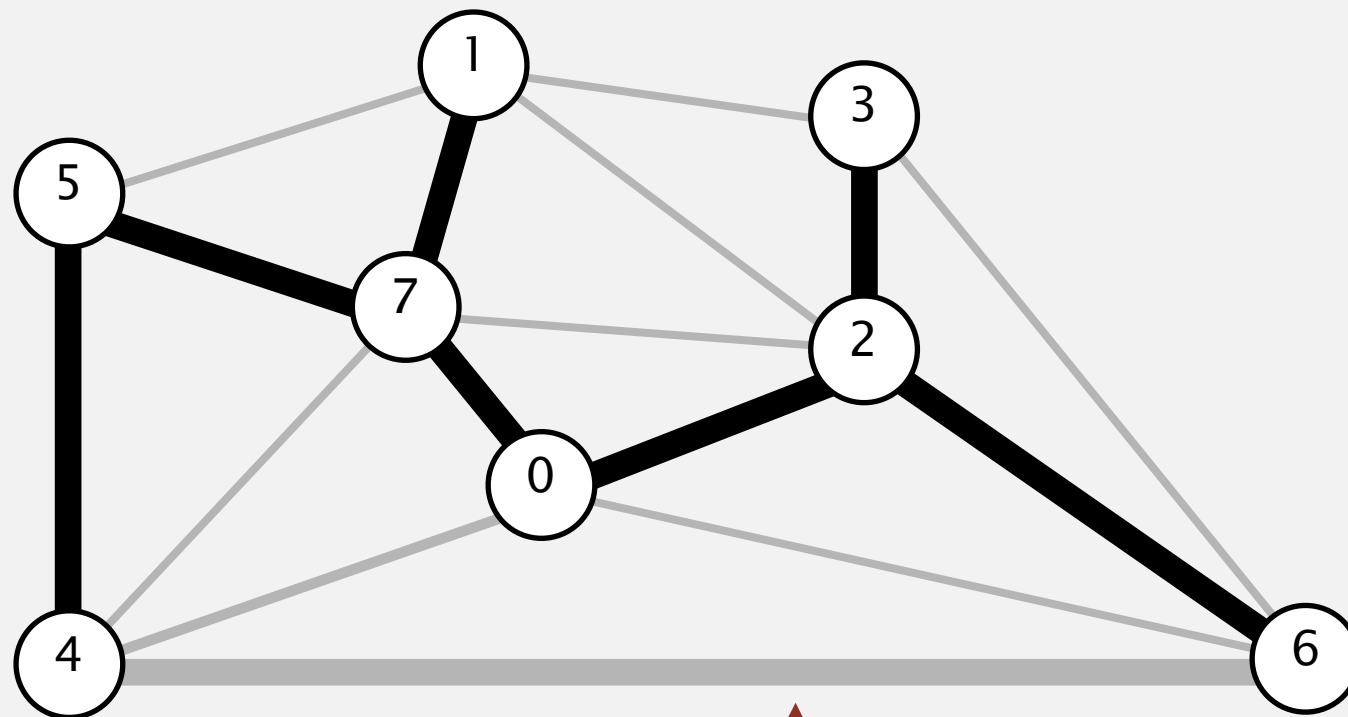- Add next edge to tree $T$ unless doing so would create a cycle.



**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
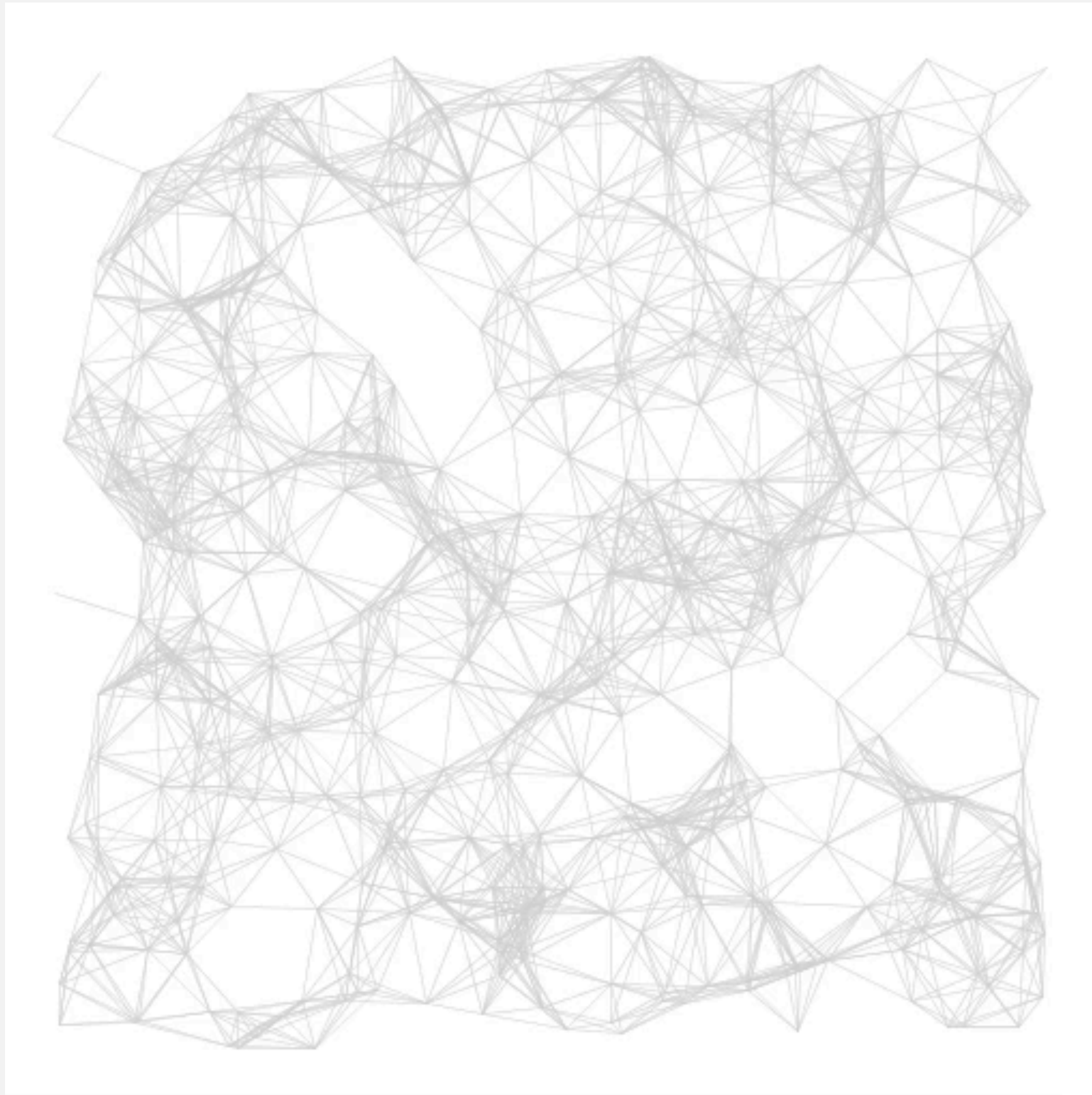
# Kruskal's algorithm: correctness proof

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 (by the algorithm).
- No crossing edge has lower weight. Why?
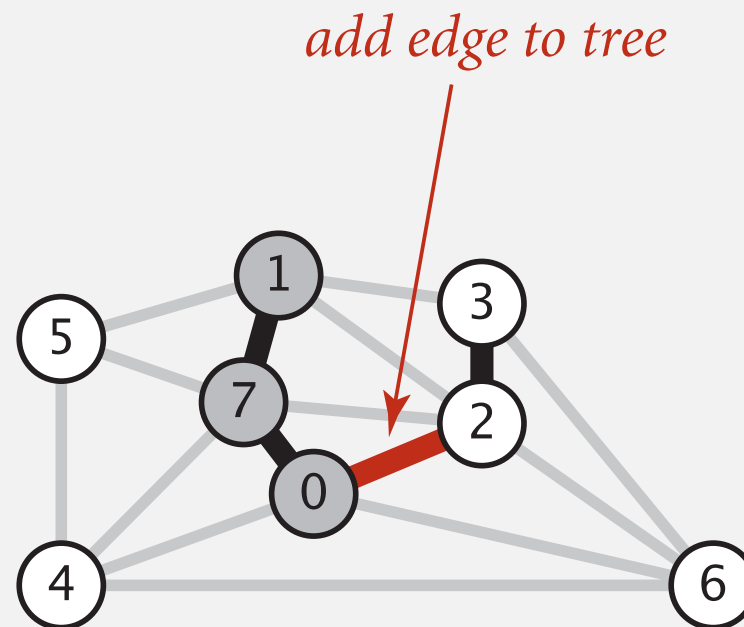
*add edge to tree*

# Kruskal's algorithm:  implementation challenge

Challenge.  Would adding edge $v$–$w$ 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 !
- $1$
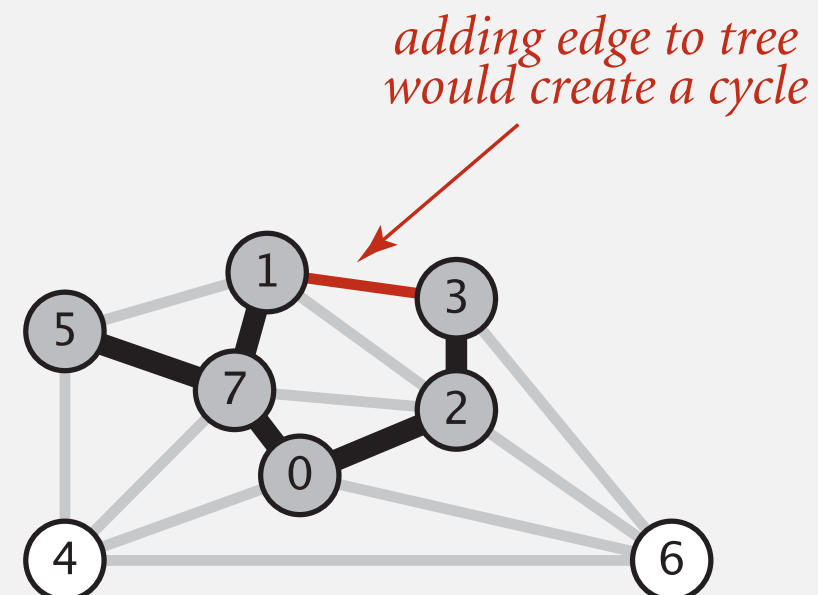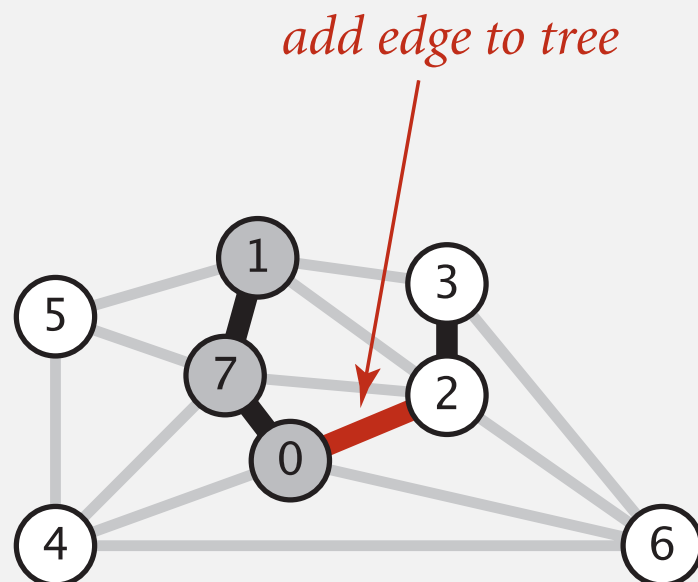


*add edge to tree*

*adding edge to tree would create a cycle*

# 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 $w$.
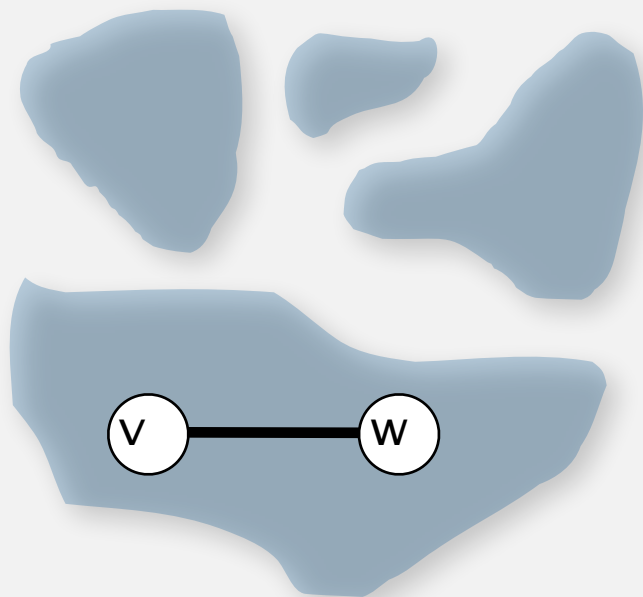


**Case 1: adding v–w creates a cycle**   **Case 2: add v–w to T and merge sets containing v and w**
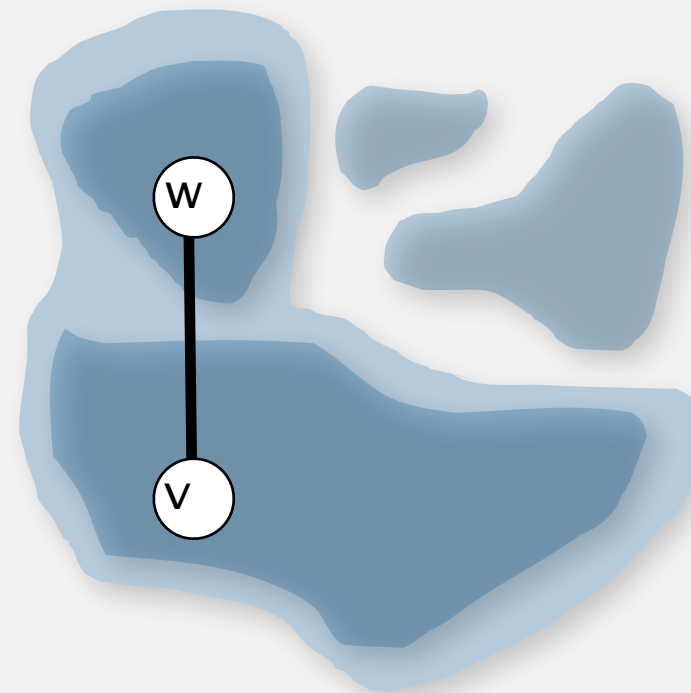
# Kruskal's algorithm:  Java implementation

$E \log E$

```java
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

40

# 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

# Minimum Spanning Trees

# 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
```

- 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.

-

```
0-7   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.
-



| | |
|---|---|
| 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 |

min weight edge with
exactly one endpoint in T

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

# 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.
- 

```
0-7   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.
-



```
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
```
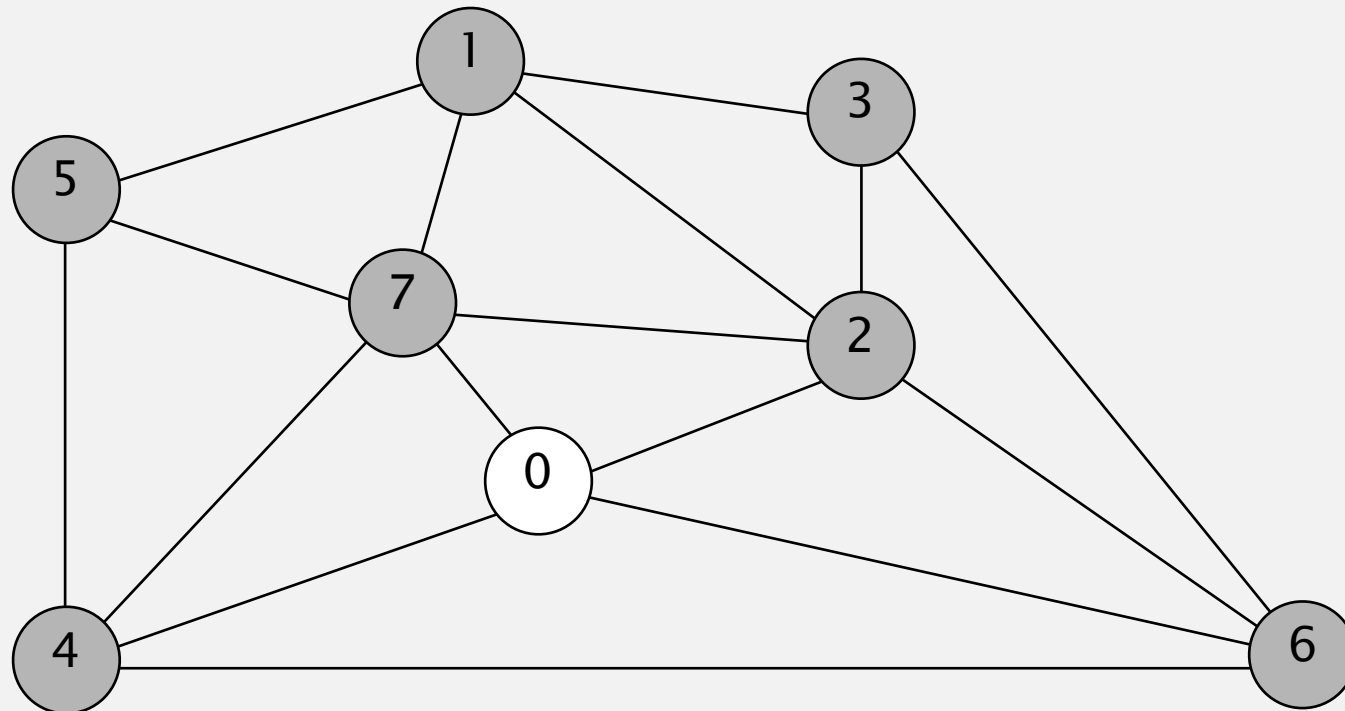
min weight edge with
exactly one endpoint in T

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**

48

# 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 $T$.
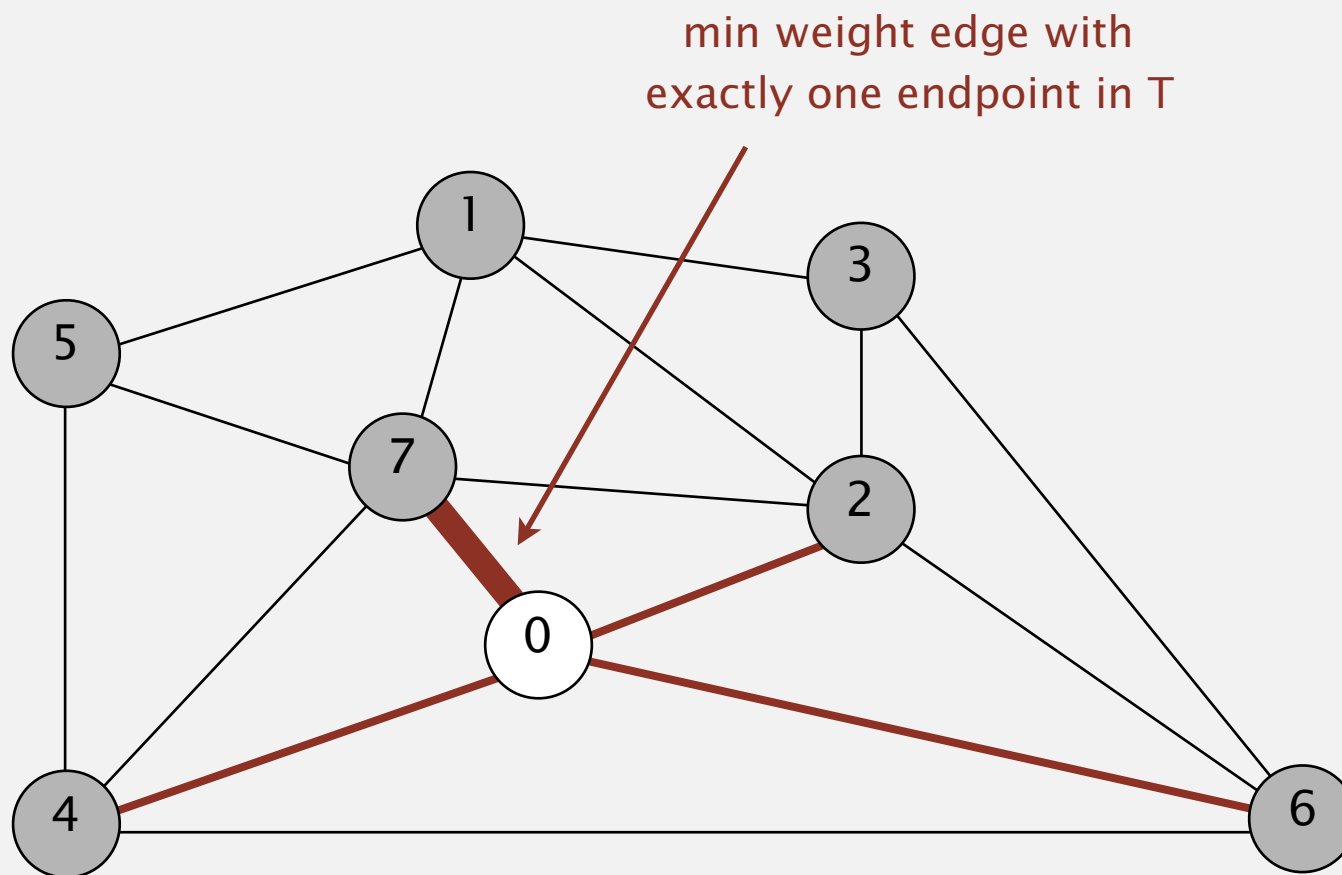- Repeat until $V - 1$ edges.
-

```
0-7  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

- 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.



```
0-7   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
```

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
```
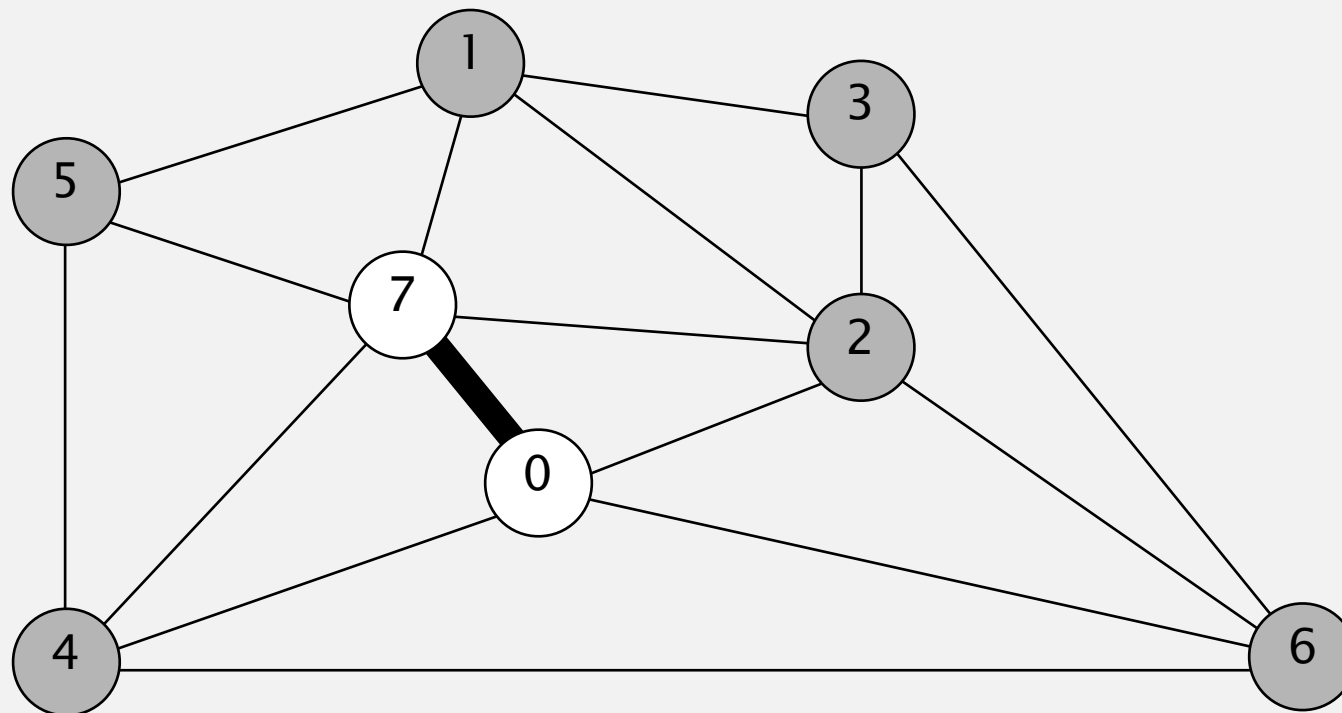
**MST edges**

0-7   1-7

50

# 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.
- 

```
0-7   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
```
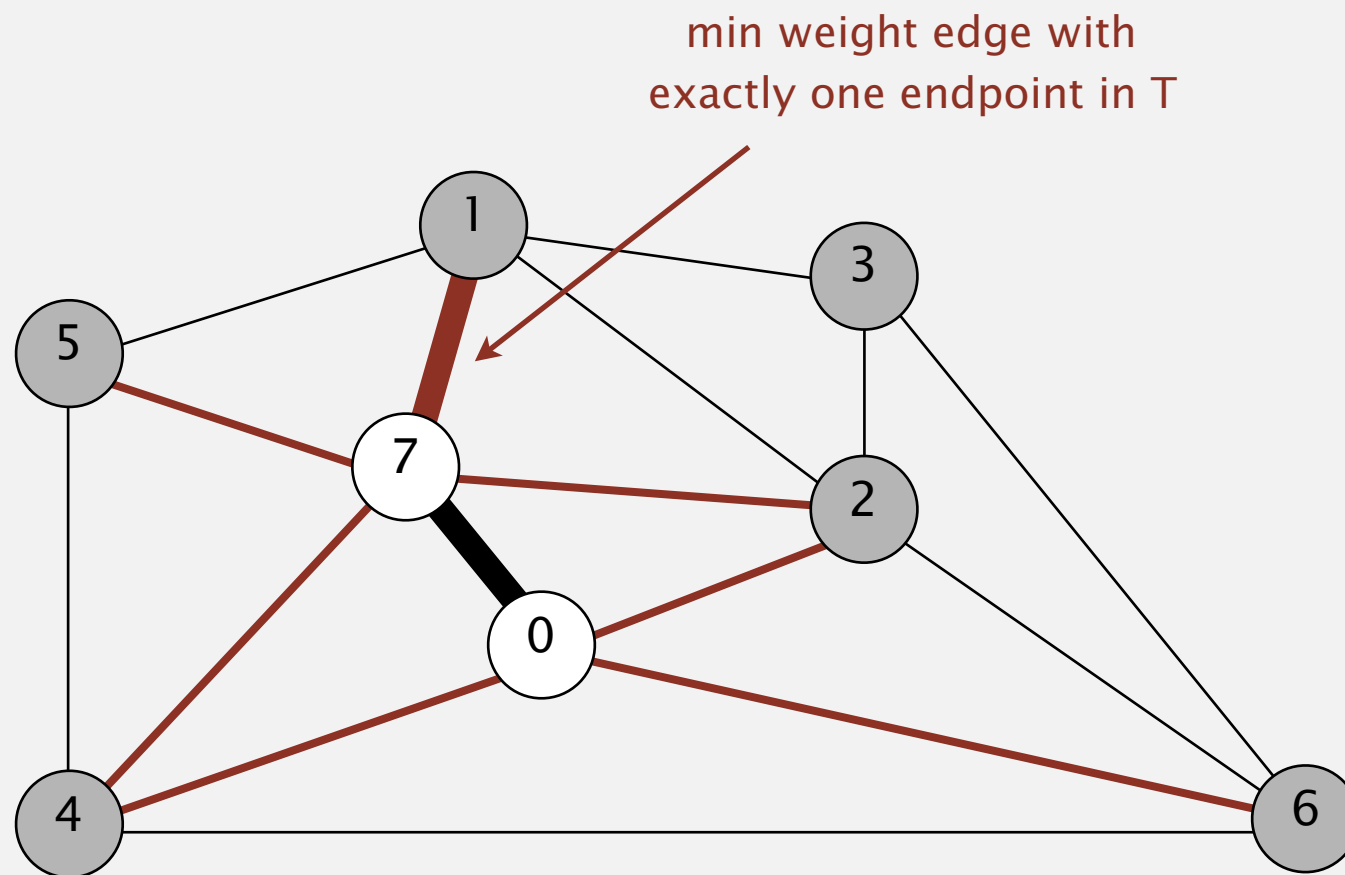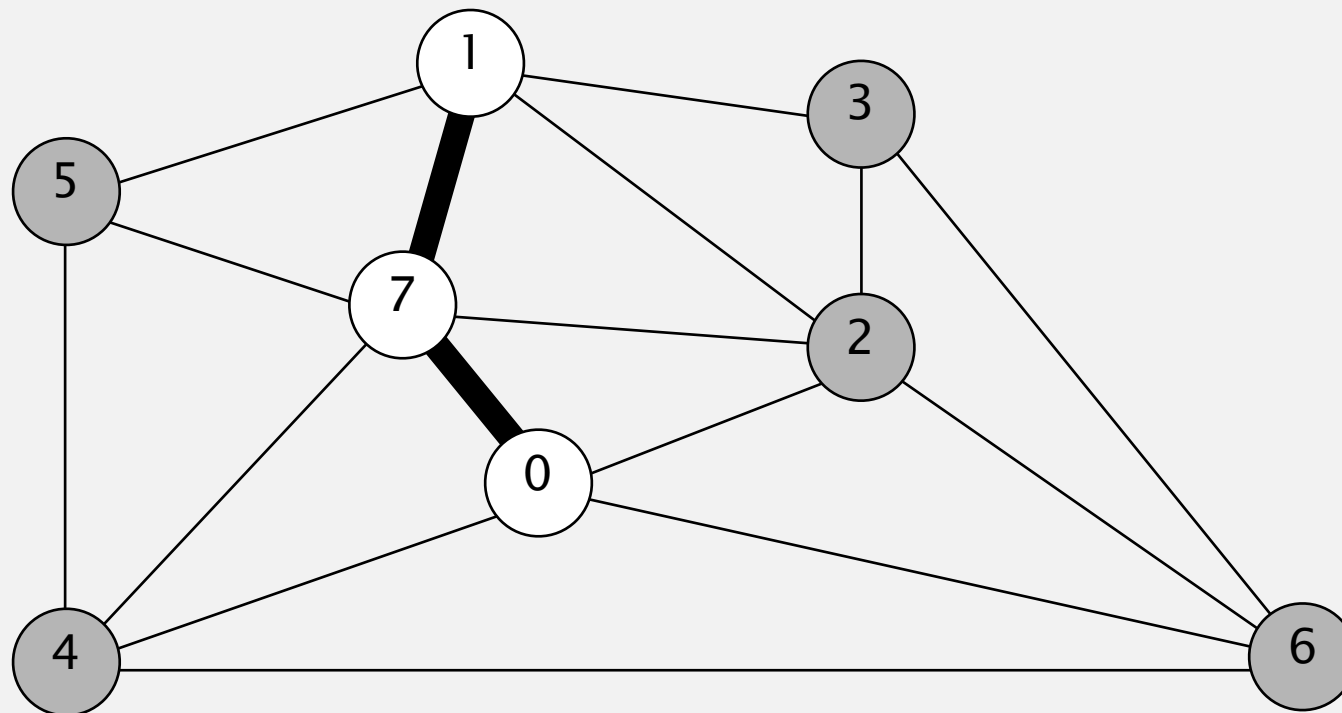


**MST edges**
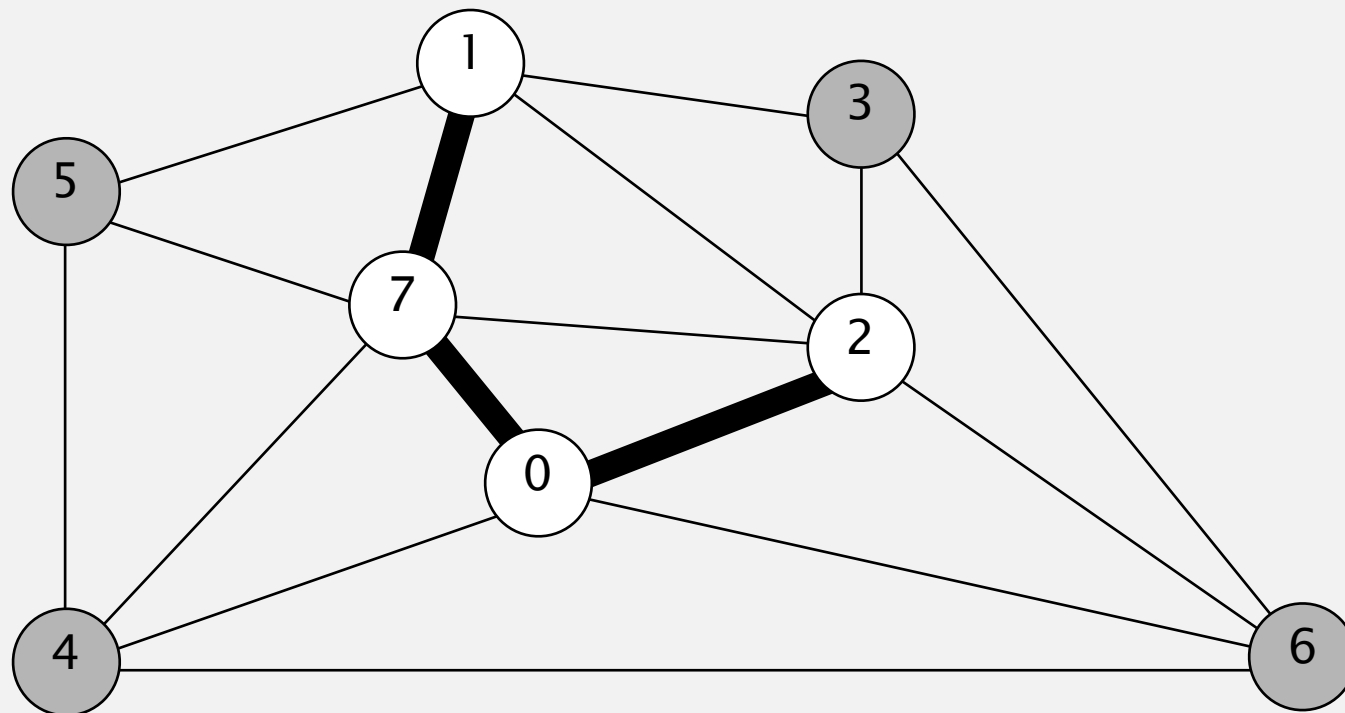
**0-7   1-7   0-2**

# 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.
-



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

min weight edge with
exactly one endpoint in T

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**

- 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.

- 

```
0-7   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
```
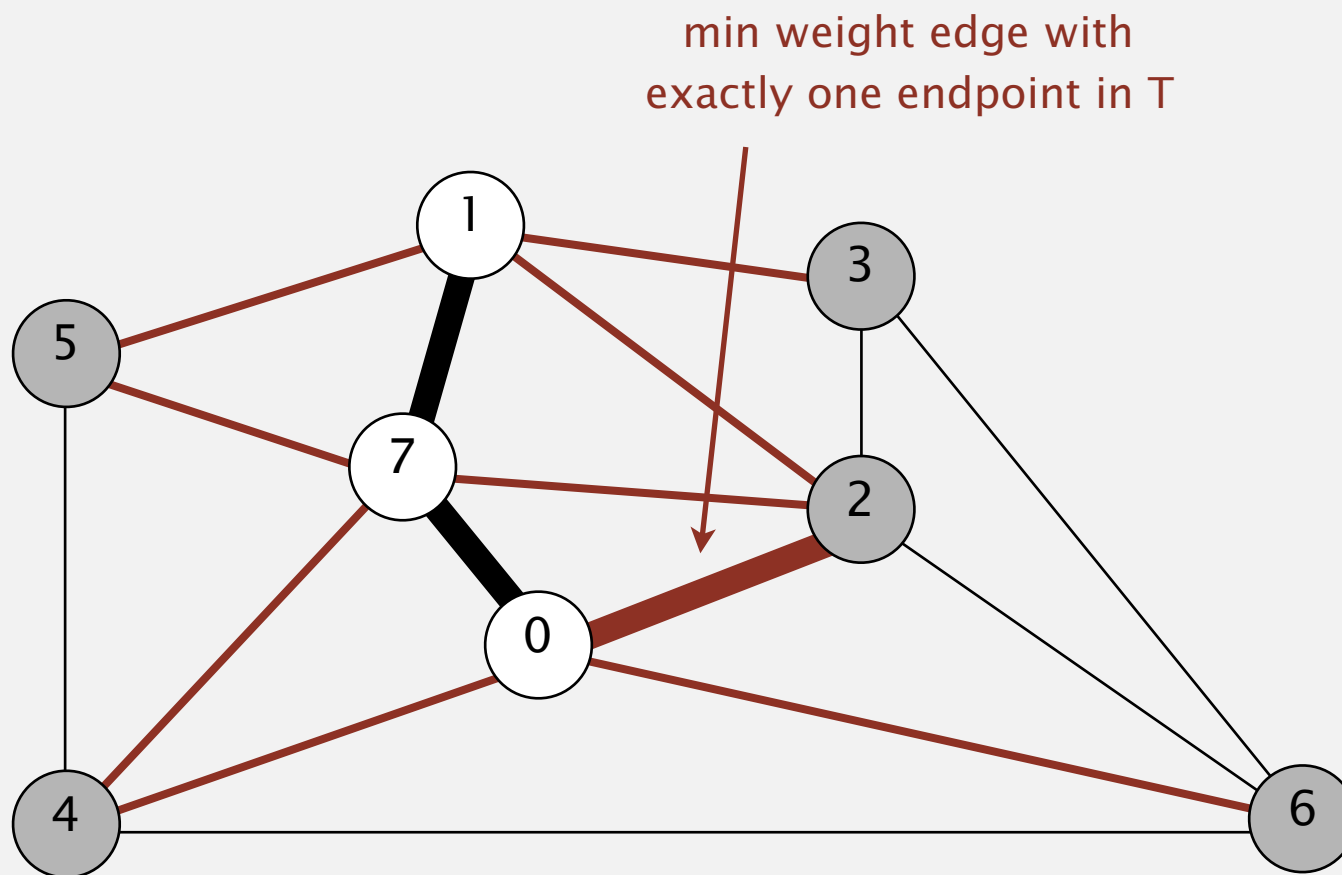


**MST edges**

**0-7   1-7   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
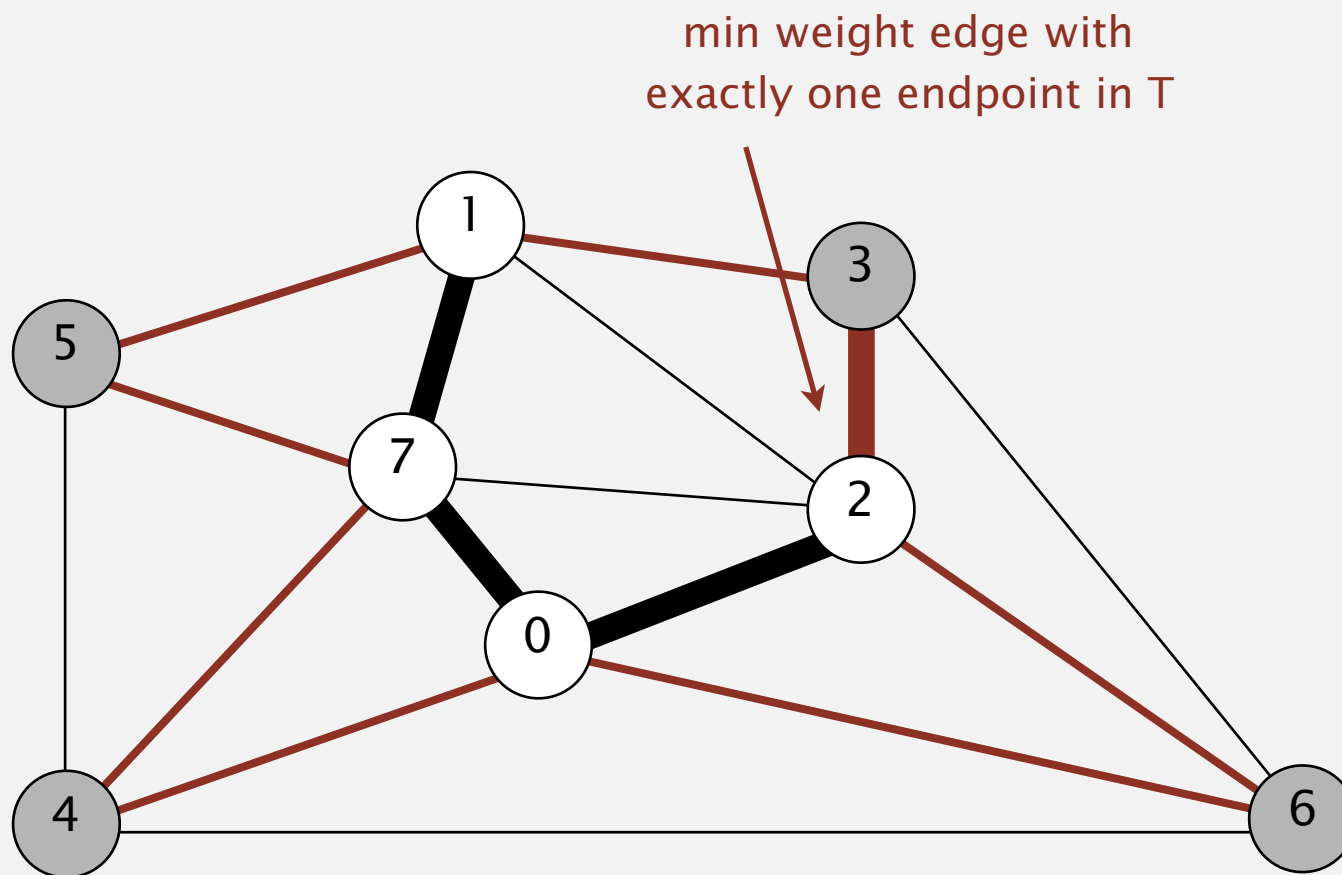          6-2  0.40
          3-6  0.52
          6-0  0.58

**MST edges**

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

- 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.
- 



| | |
|---|---|
| 0-7 | 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   0-2   2-3   5-7**

- 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.

- 

```
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
```
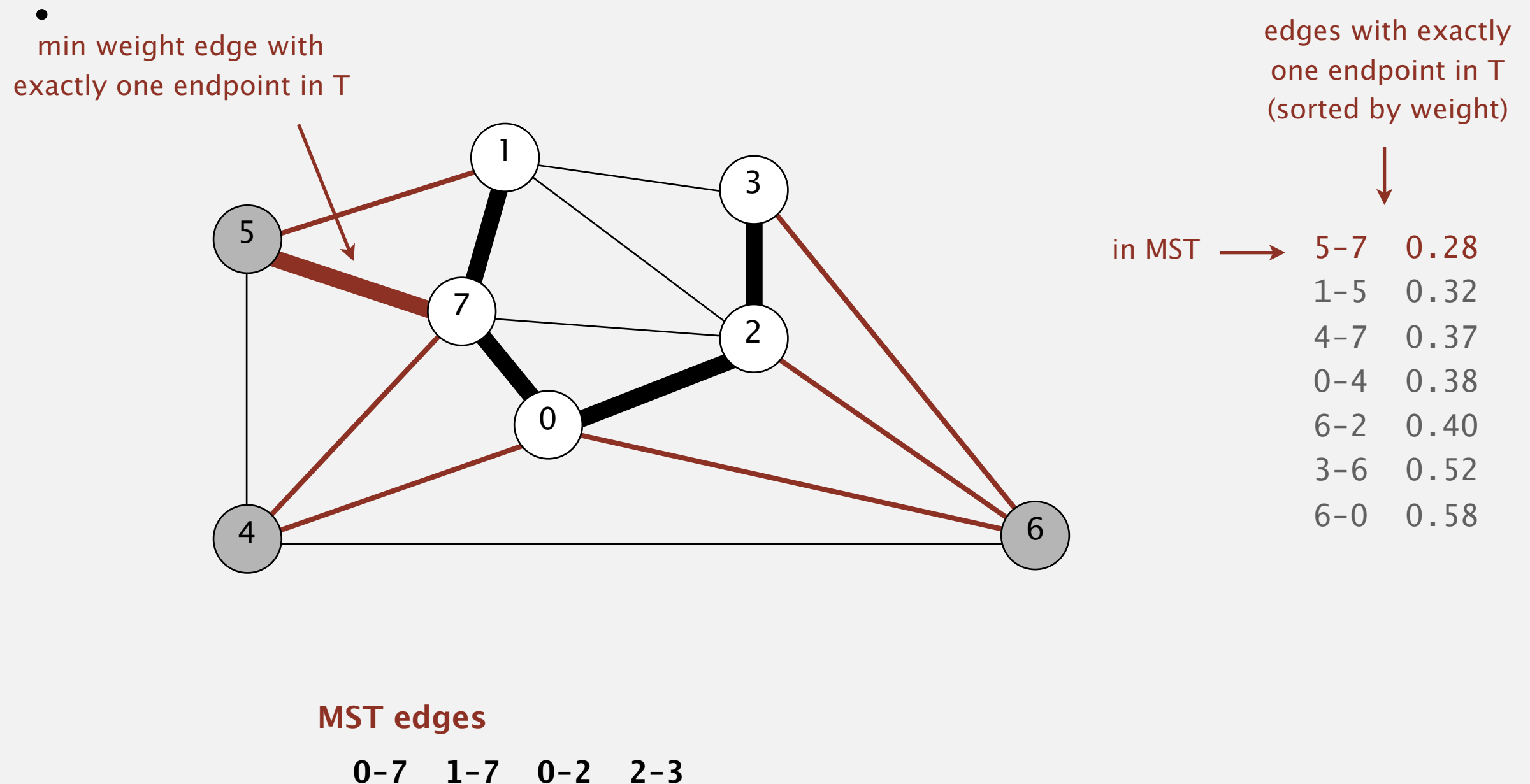


**MST edges**
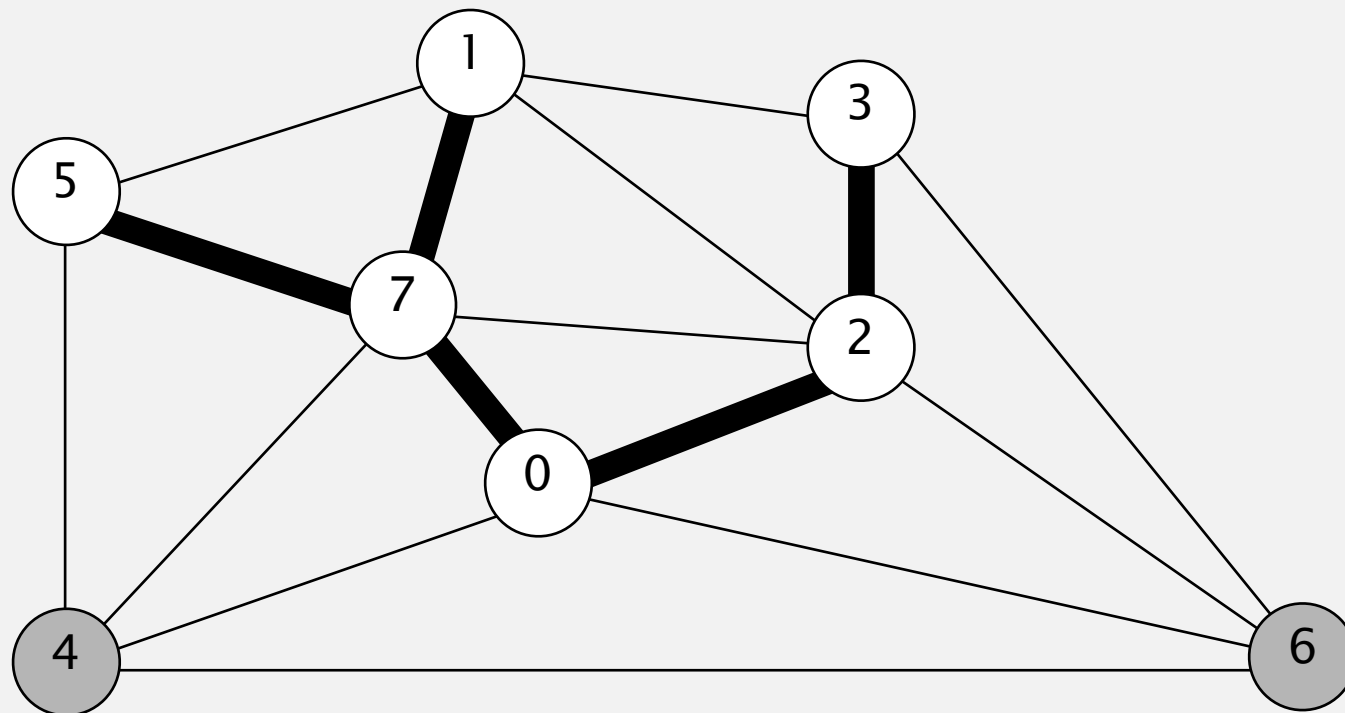
0-7   1-7   0-2   2-3   5-7   4-5

# 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.

- 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.
- 

```
0-7   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
```
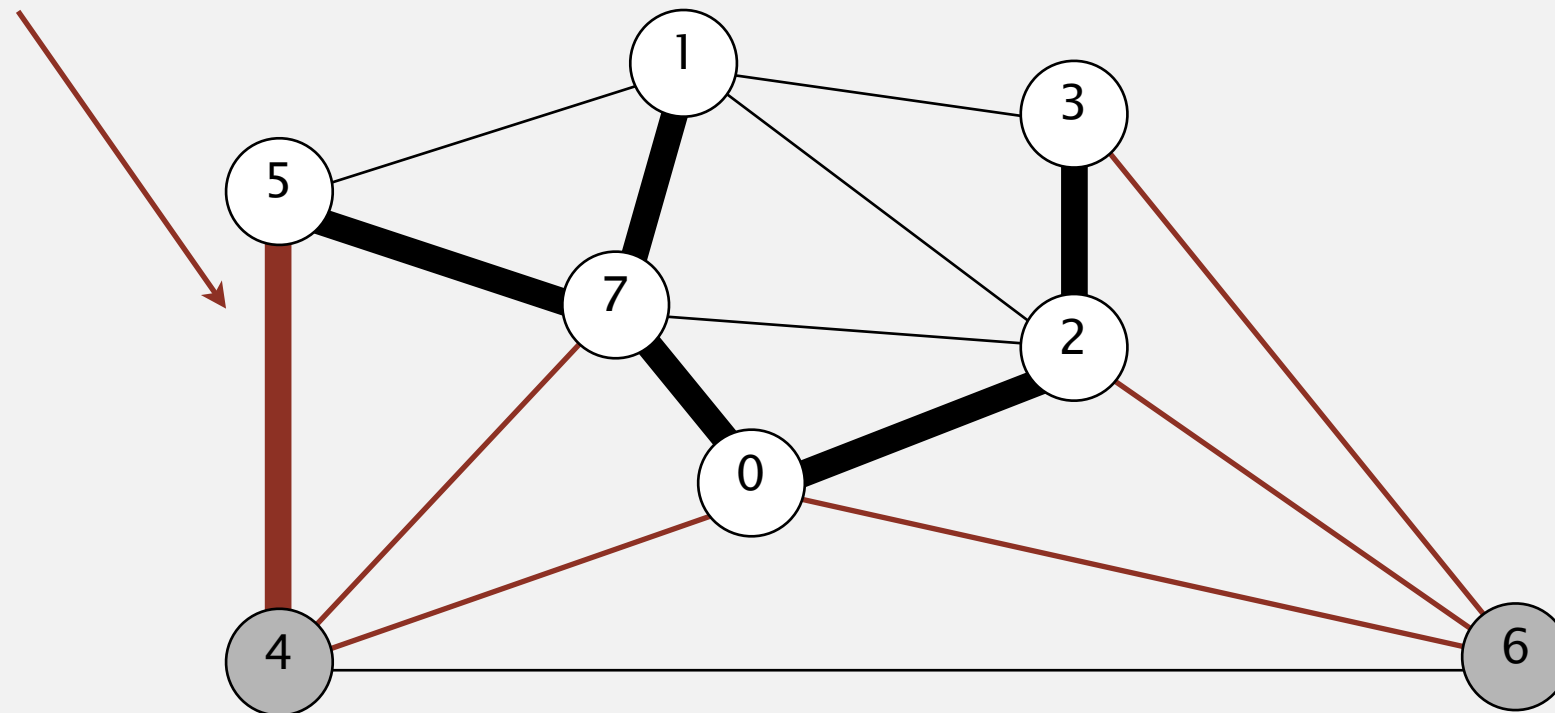


**MST edges**

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

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

How difficult?

- $E$ ⟵ try all edges

- $E \log E$

- $\log E$ ⟵ 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
```
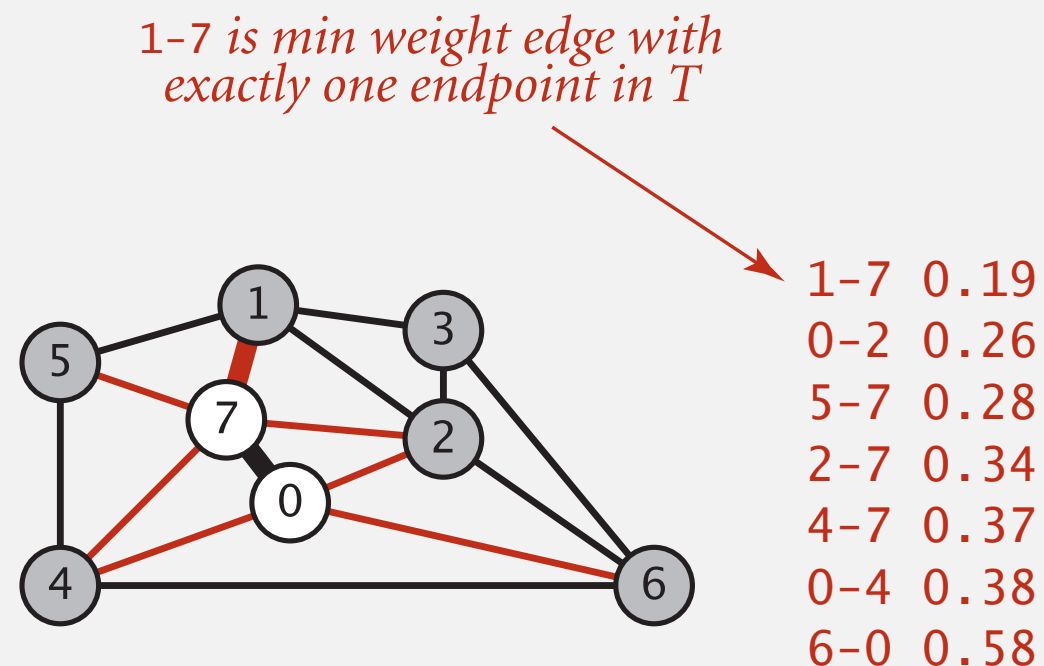
# Prim's algorithm:  lazy implementation

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 $e = v$–$w$ 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 $w$

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

*priority queue*
*of crossing edges*



```
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

```
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);                                    ← assume G is connected


         while (!pq.isEmpty() && mst.size() < G.V() - 1)
         {                                               ← repeatedly delete the
             Edge e = pq.delMin();                          min weight edge e = v–w from PQ
             int v = e.either(), w = e.other(v);
             if (marked[v] && marked[w]) continue;       ← ignore if both endpoints in T
             mst.enqueue(e);                             ← add edge e to tree
             if (!marked[v]) visit(G, v);
             if (!marked[w]) visit(G, w);                ← add v or w to tree
         }
     }
}
```

heap

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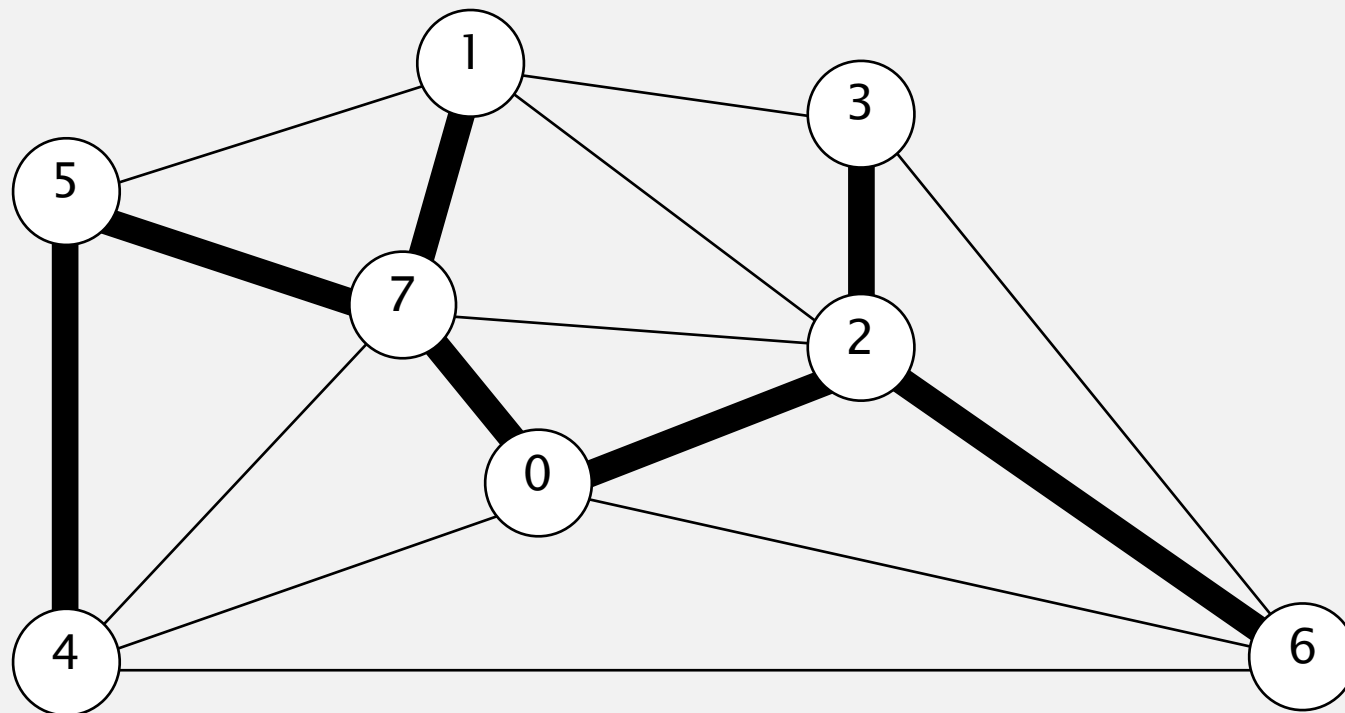
- Start with vertex $0$ and greedily grow tree $T$.
- Add to $T$ the min weight edge with exactly one endpoint in $T$.
- Repeat until $V - 1$ edges.

# 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**
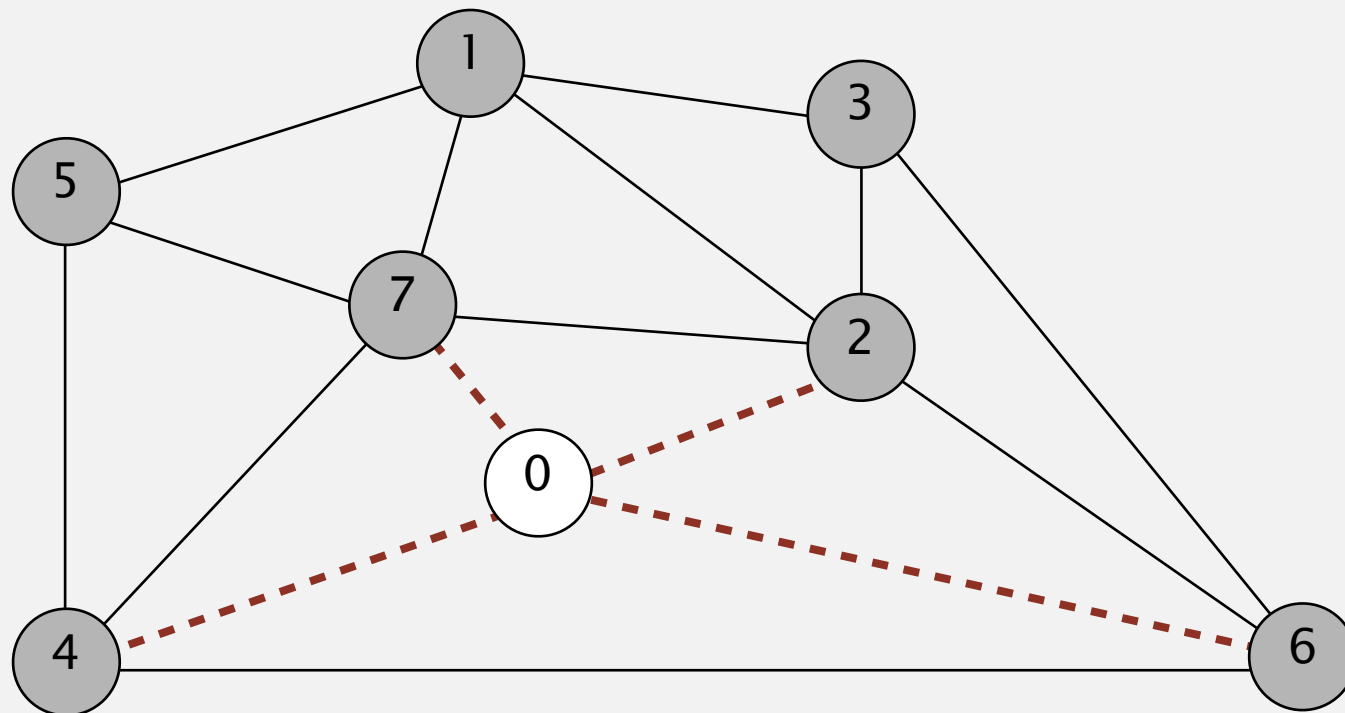


edges on PQ
(sorted by weight)

* 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 $T$.
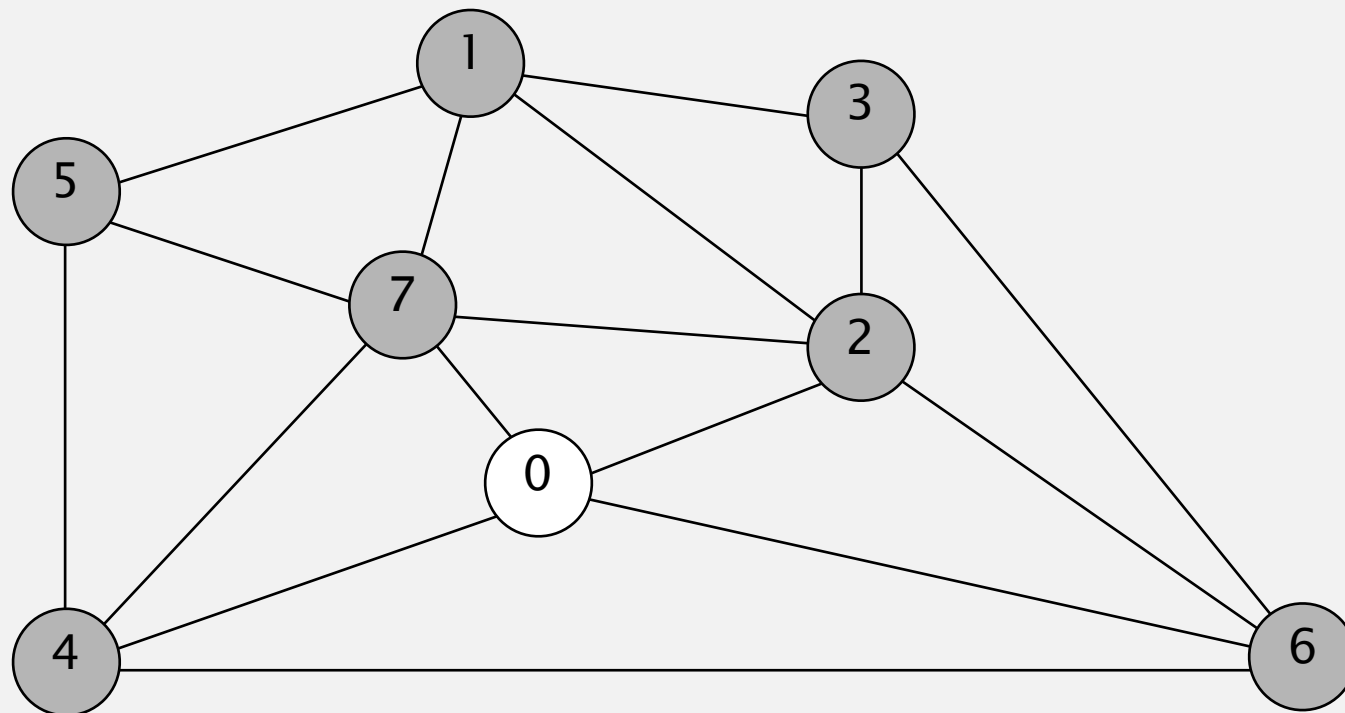- Repeat until $V - 1$ edges.

**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**

- 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 $T$.
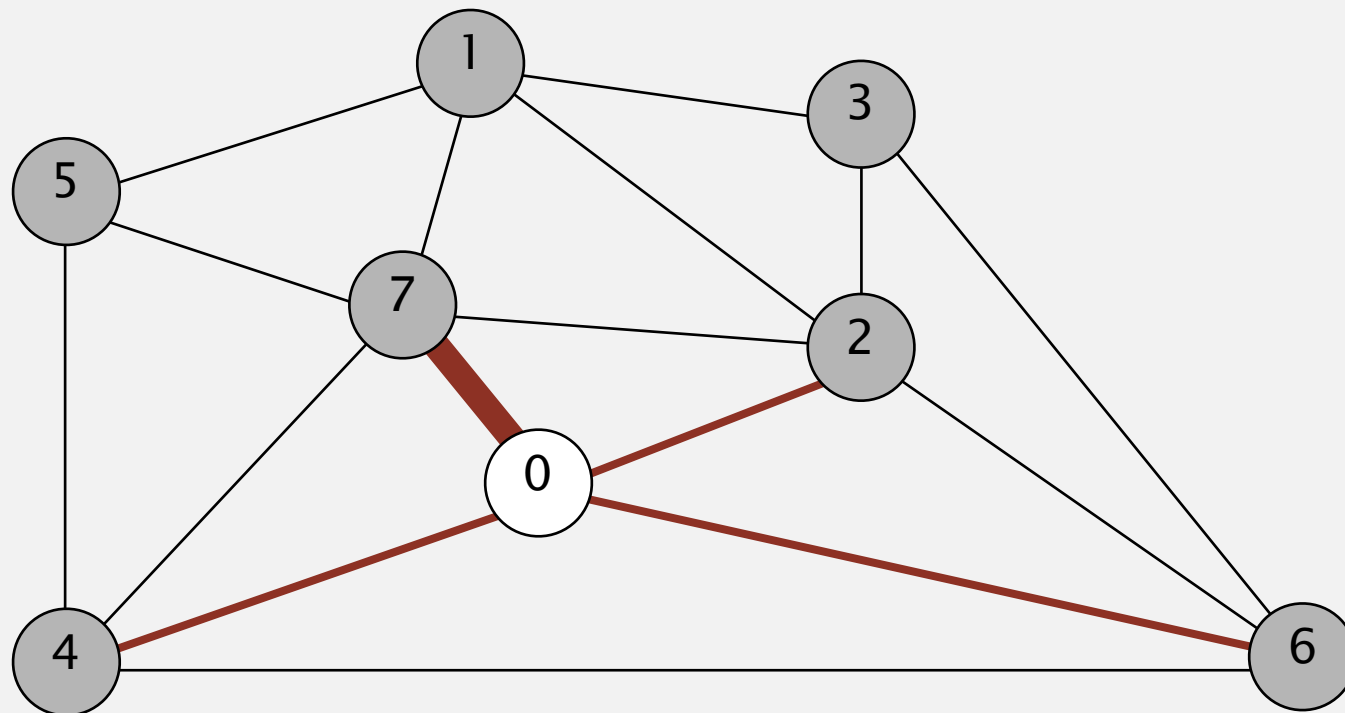- Repeat until $V - 1$ edges.

**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 $T$.
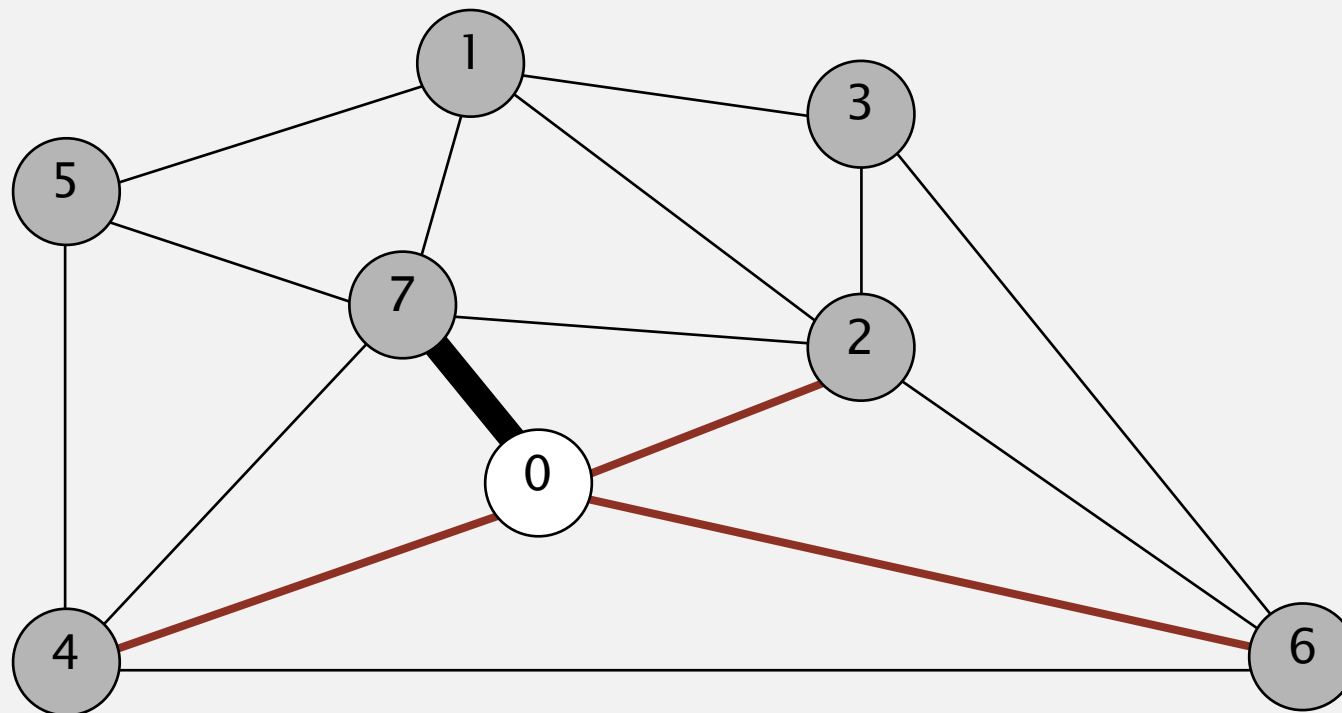- Repeat until $V - 1$ edges.

**delete edge 0–2 and add to MST**



edges on PQ
(sorted by weight)

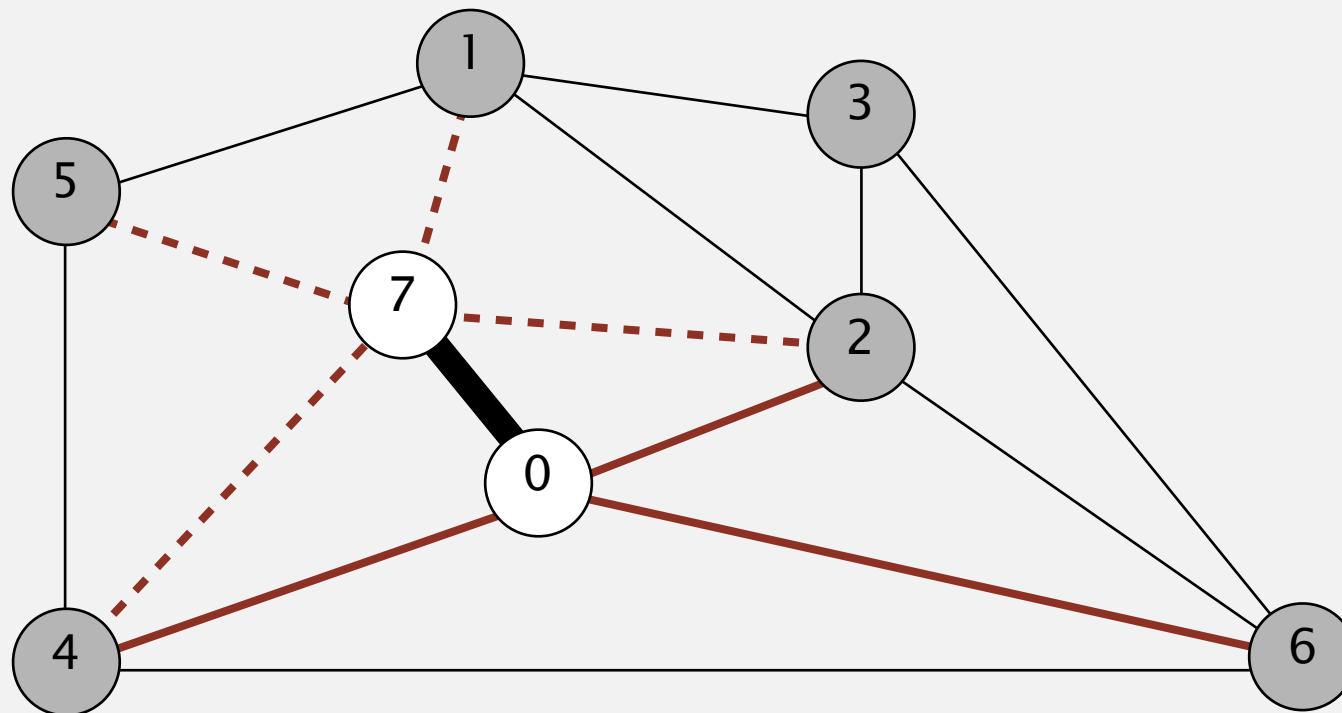| | |
|---|---|
| 0-2 | 0.26 |
| 5-7 | 0.28 |
| 1-3 | 0.29 |
| 1-5 | 0.32 |
| 2-7 | 0.34 |
| 1-2 | 0.36 |
| 4-7 | 0.37 |
| 0-4 | 0.38 |
| 6-0 | 0.58 |

**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 $T$.
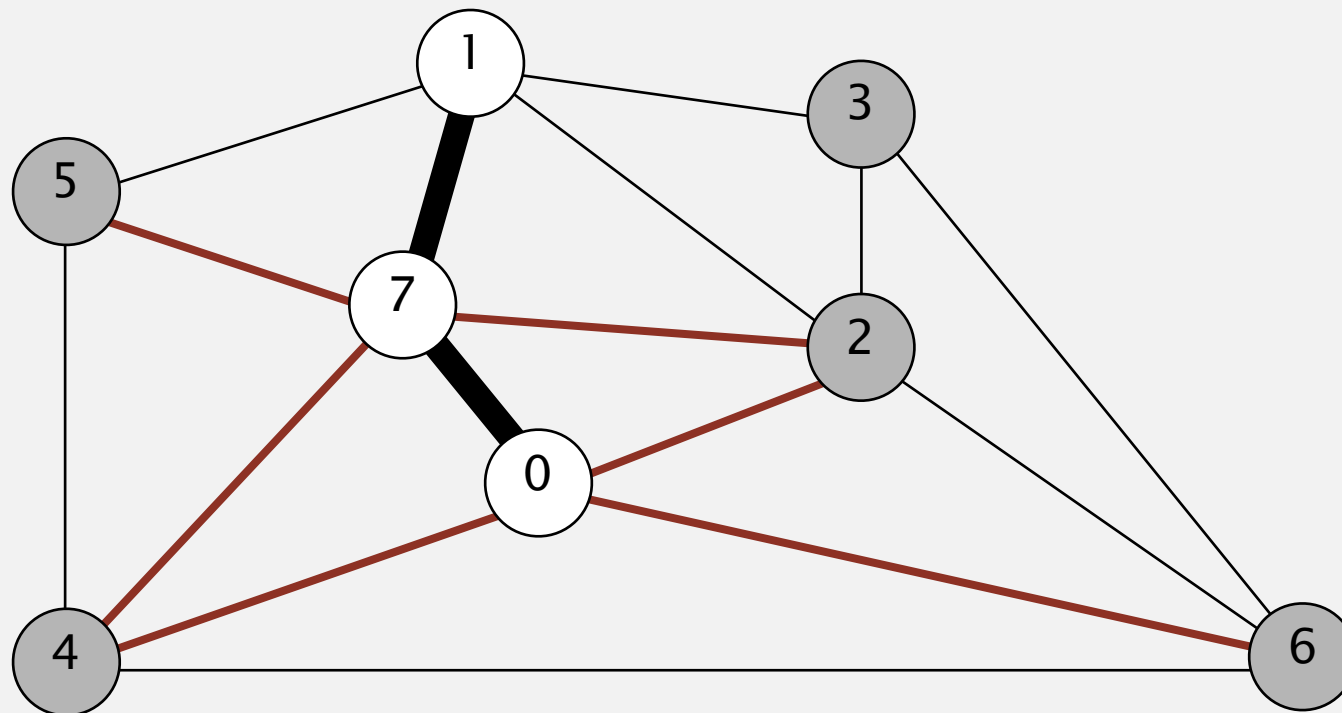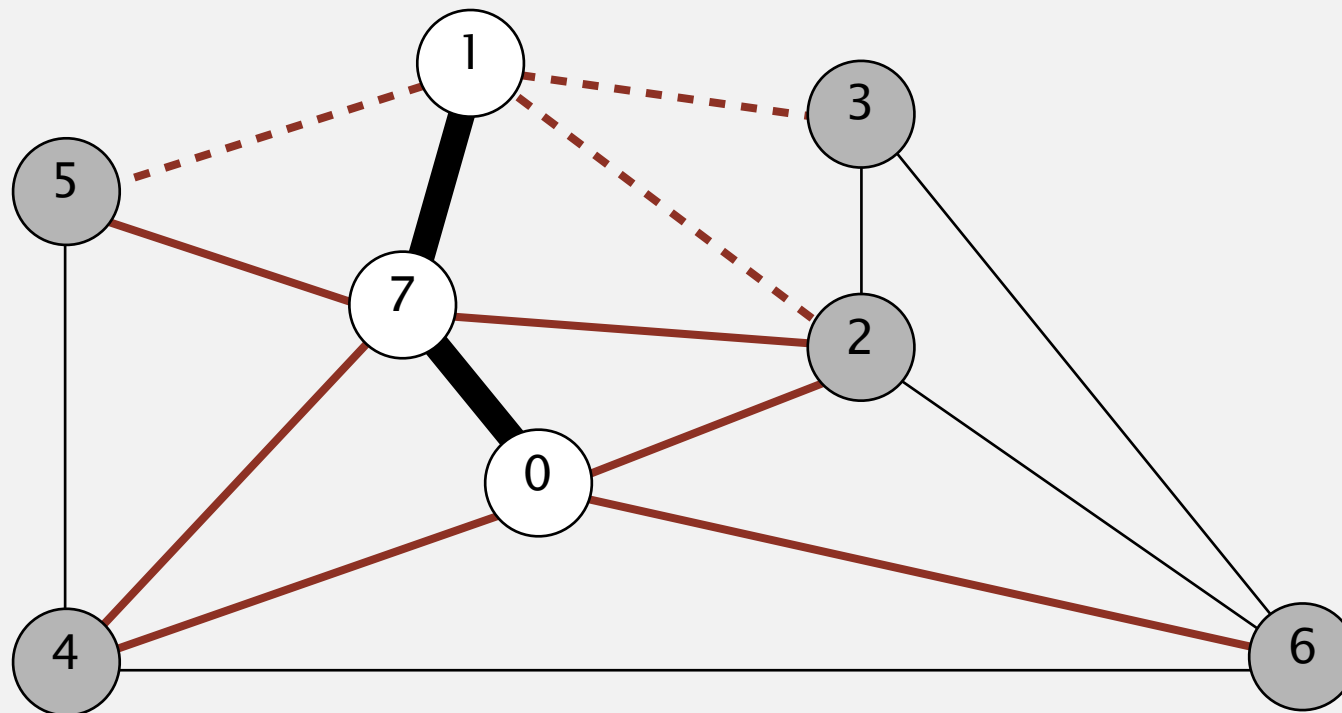- Repeat until $V - 1$ edges.

**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**
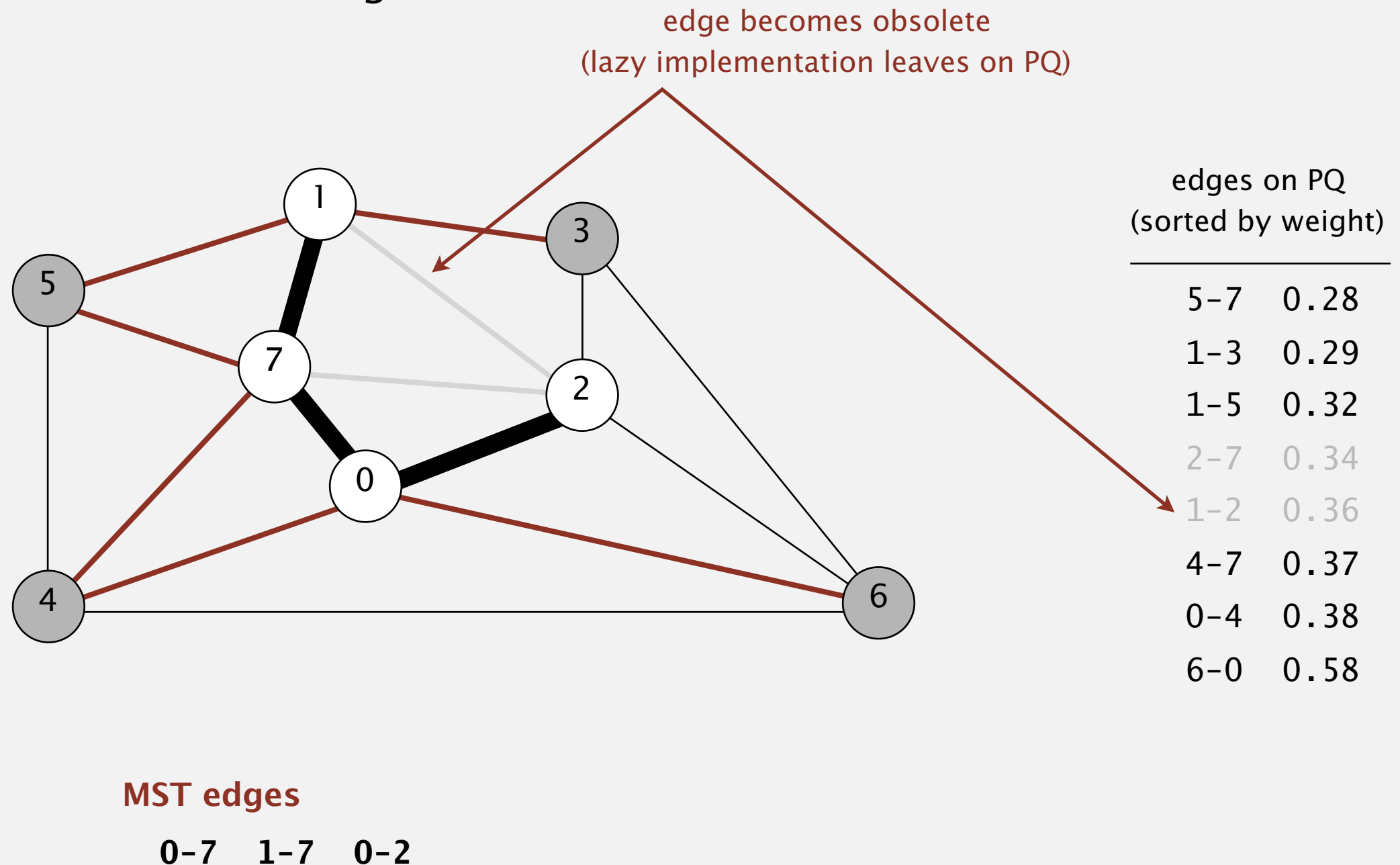


edges on PQ
(sorted by weight)

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

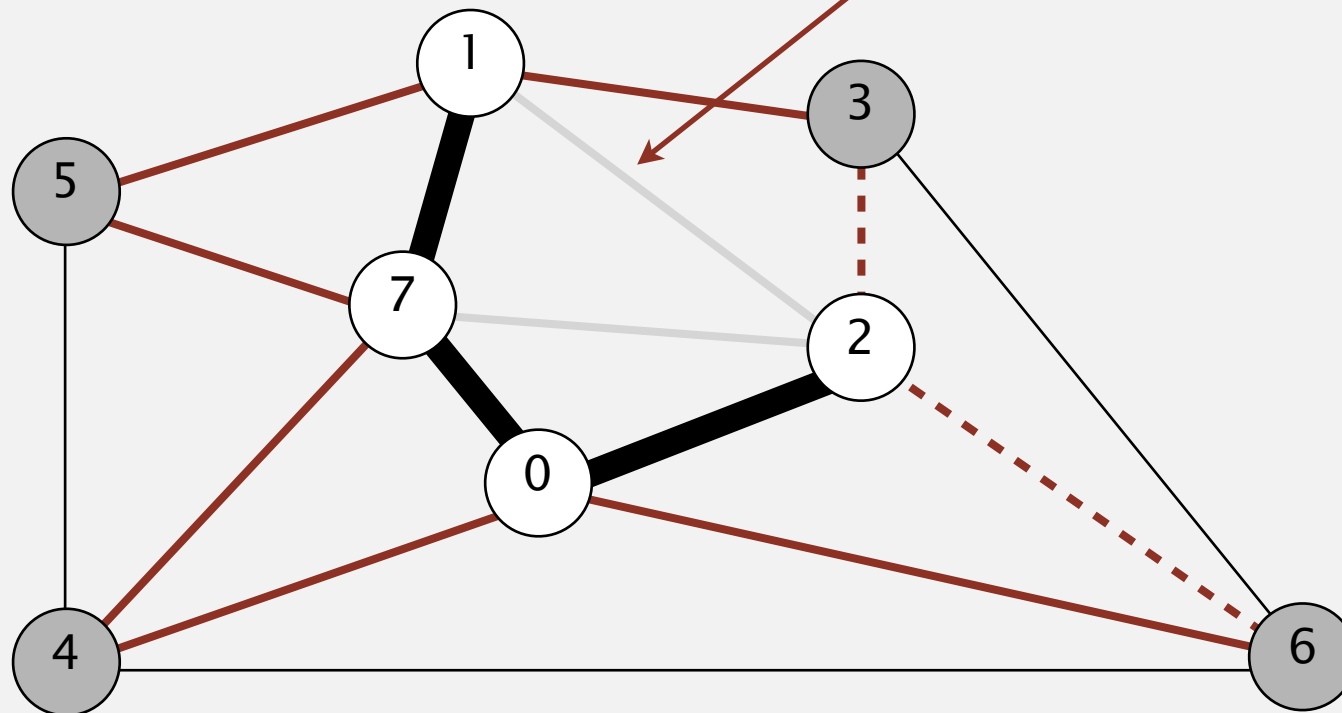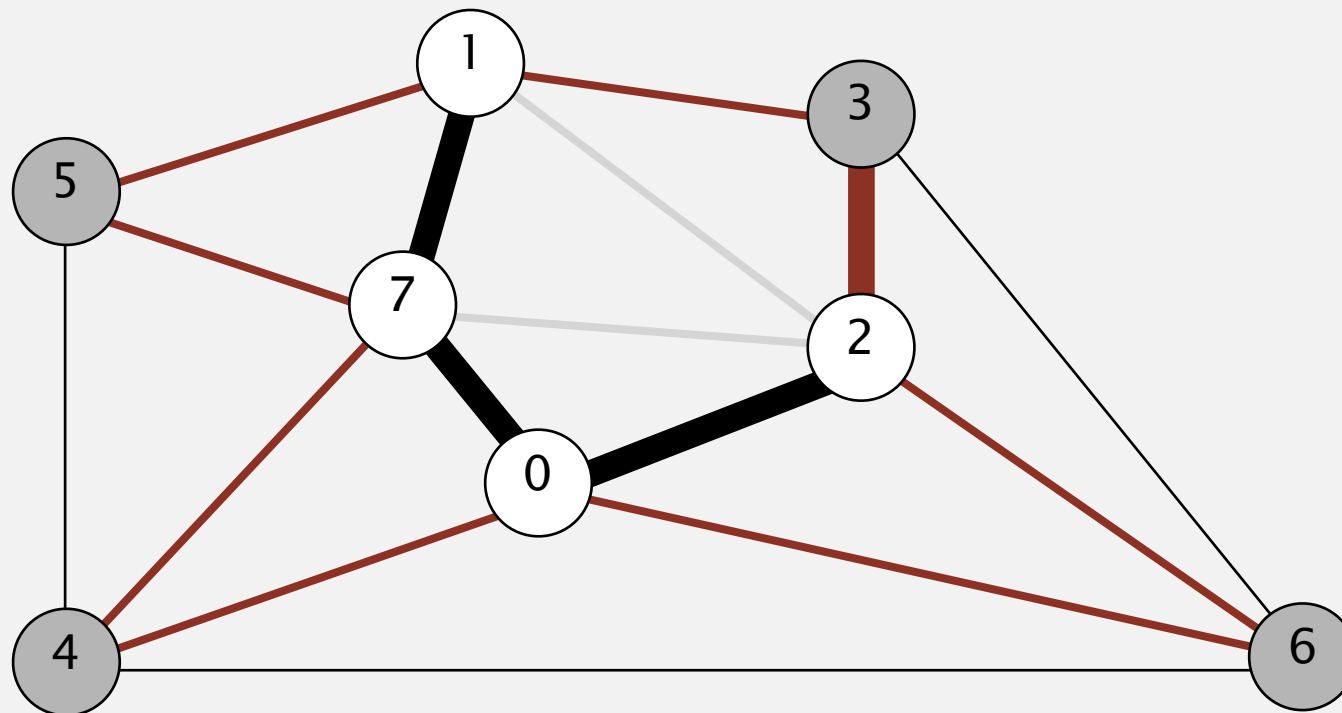# Prim's algorithm:  lazy implementation demo

- 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**
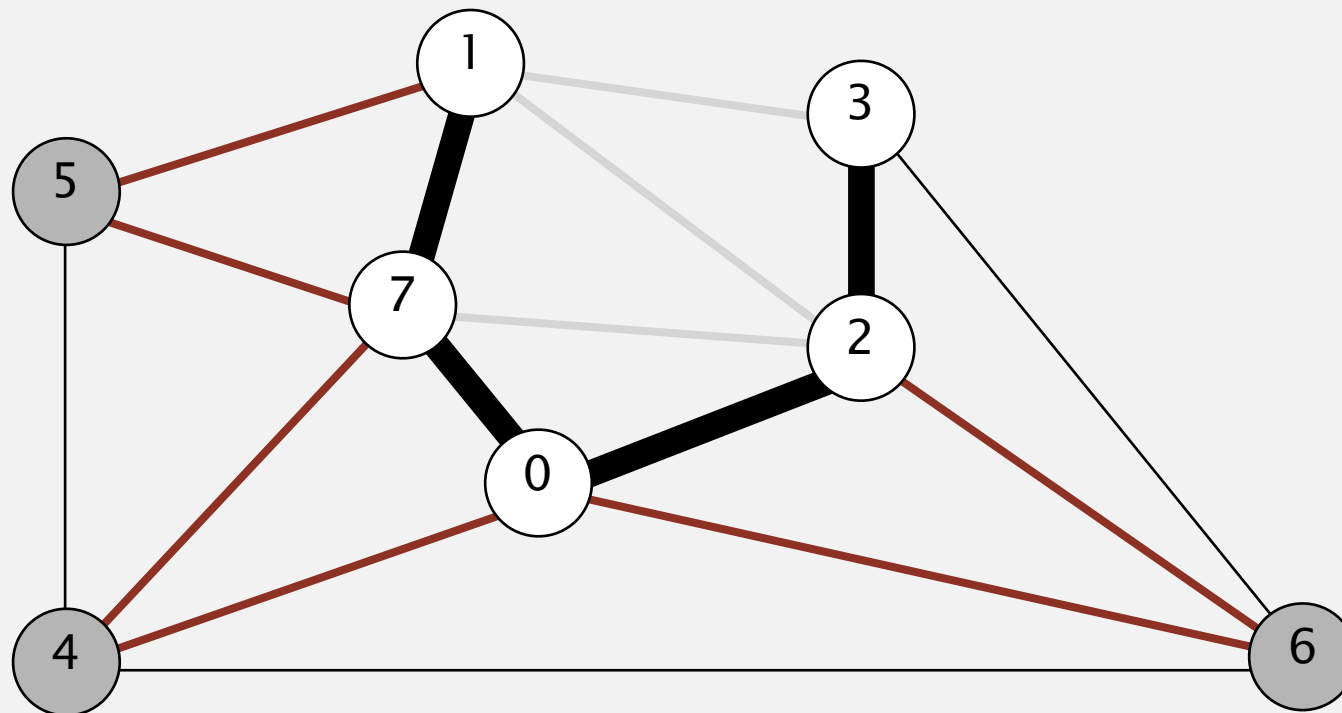


edges on PQ
(sorted by weight)

| | |
|---|---|
| 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)

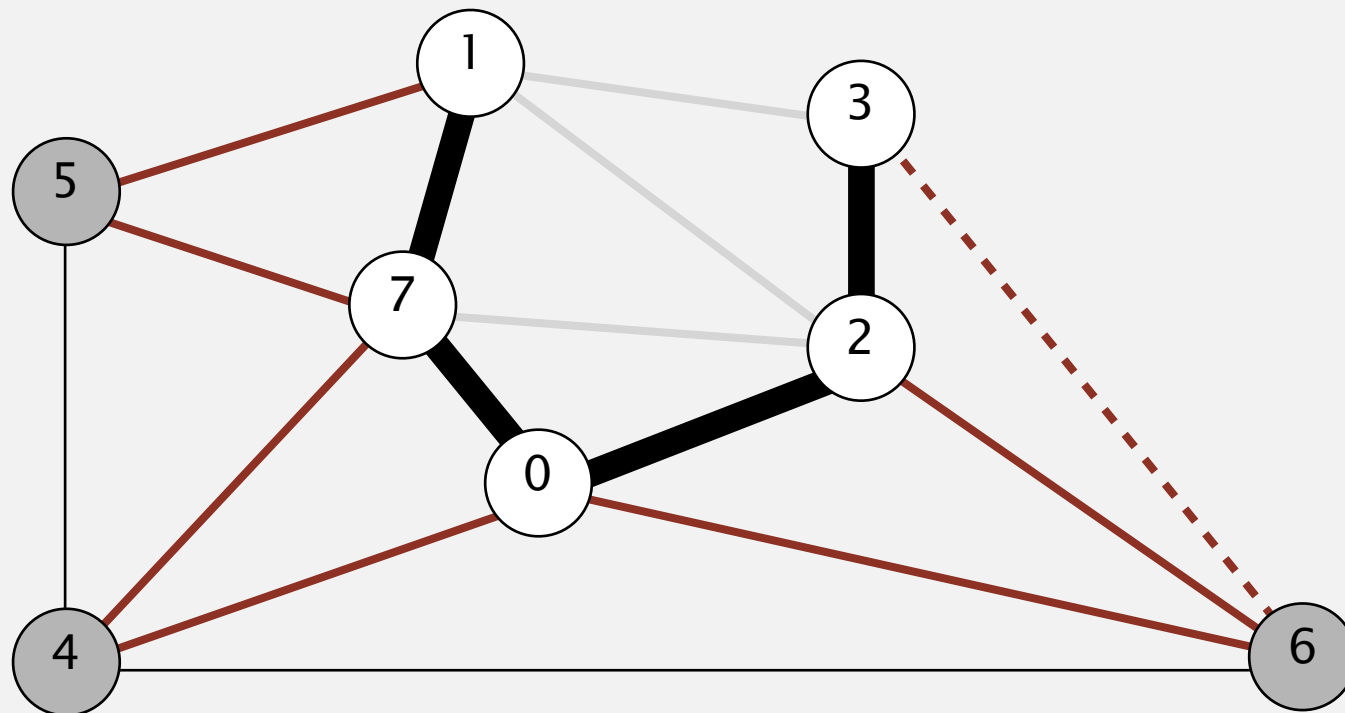| | |
|---|---|
| 1–3 | 0.29 |
| 1–5 | 0.32 |
| 2–7 | 0.34 |
| 1–2 | 0.36 |
| 4–7 | 0.37 |
| 0–4 | 0.38 |
| 6–2 | 0.40 |
| 3–6 | 0.52 |
| 6–0 | 0.58 |

**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
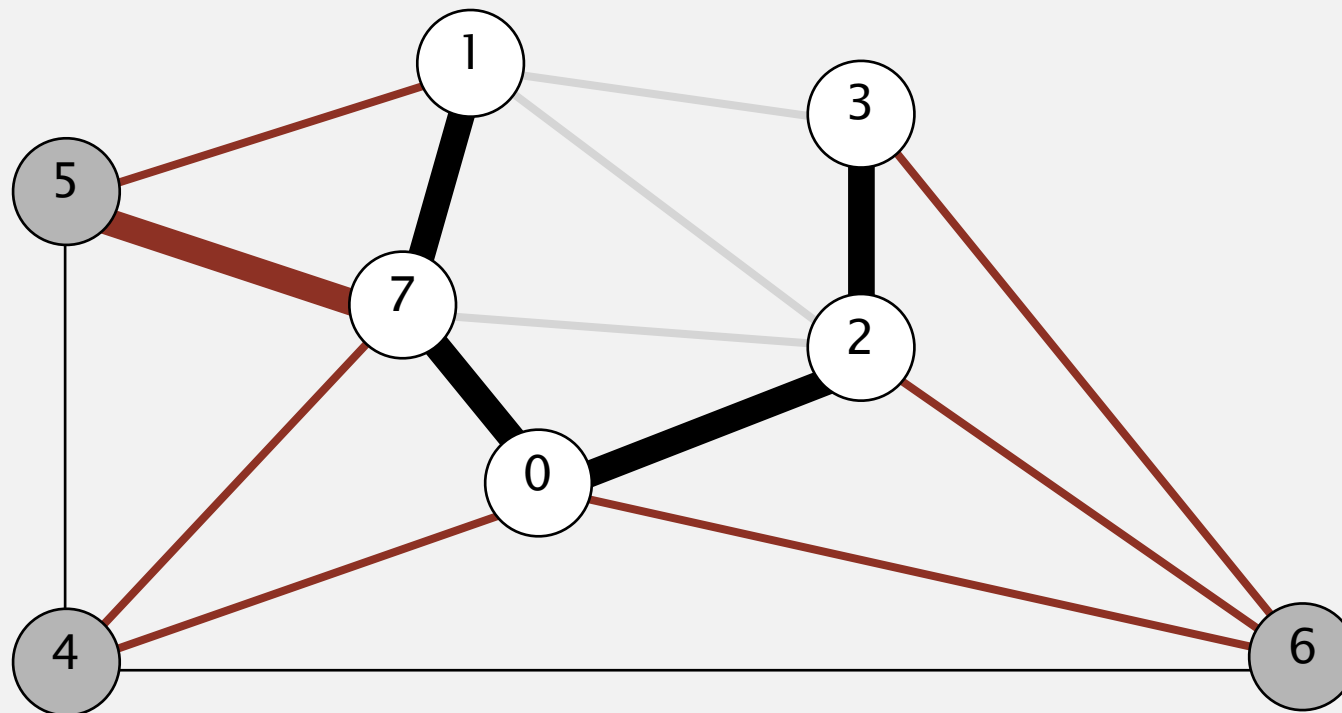(sorted by weight)

| | |
|---|---|
| 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

- 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**
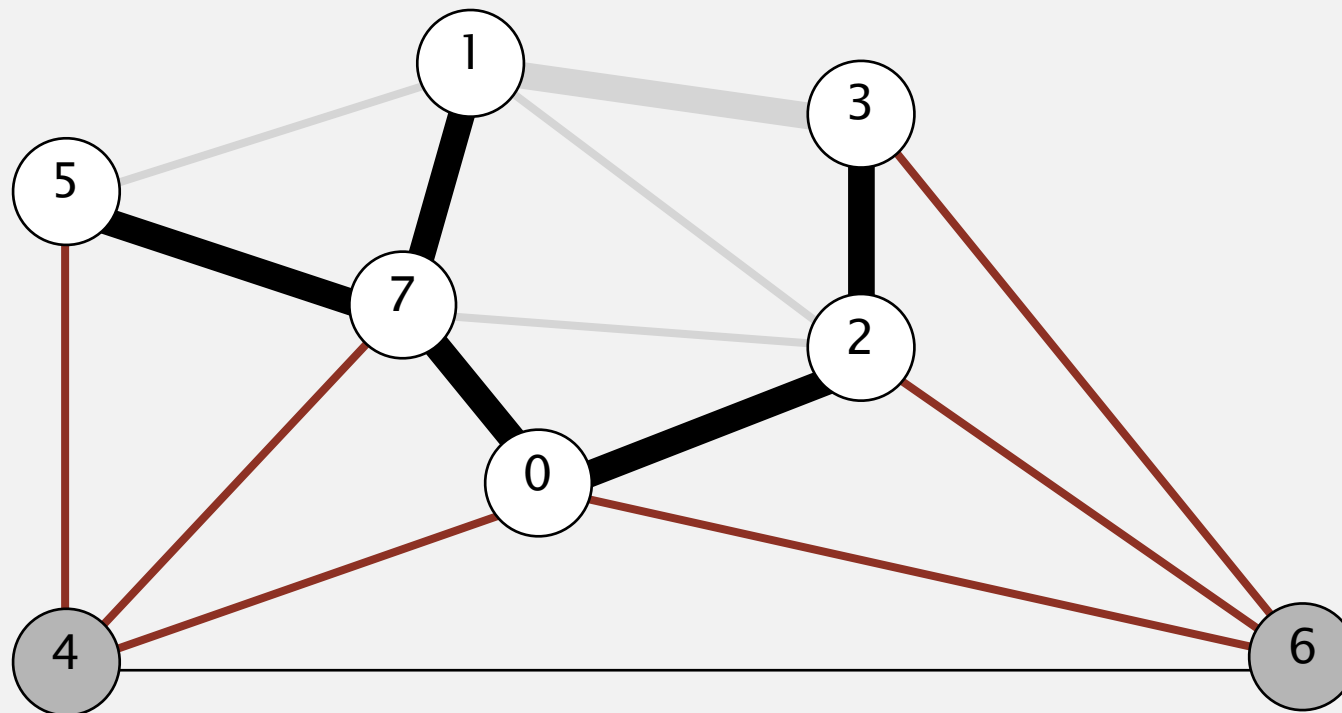


edges on PQ
(sorted by weight)

| | |
|---|---|
| 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 $T$.
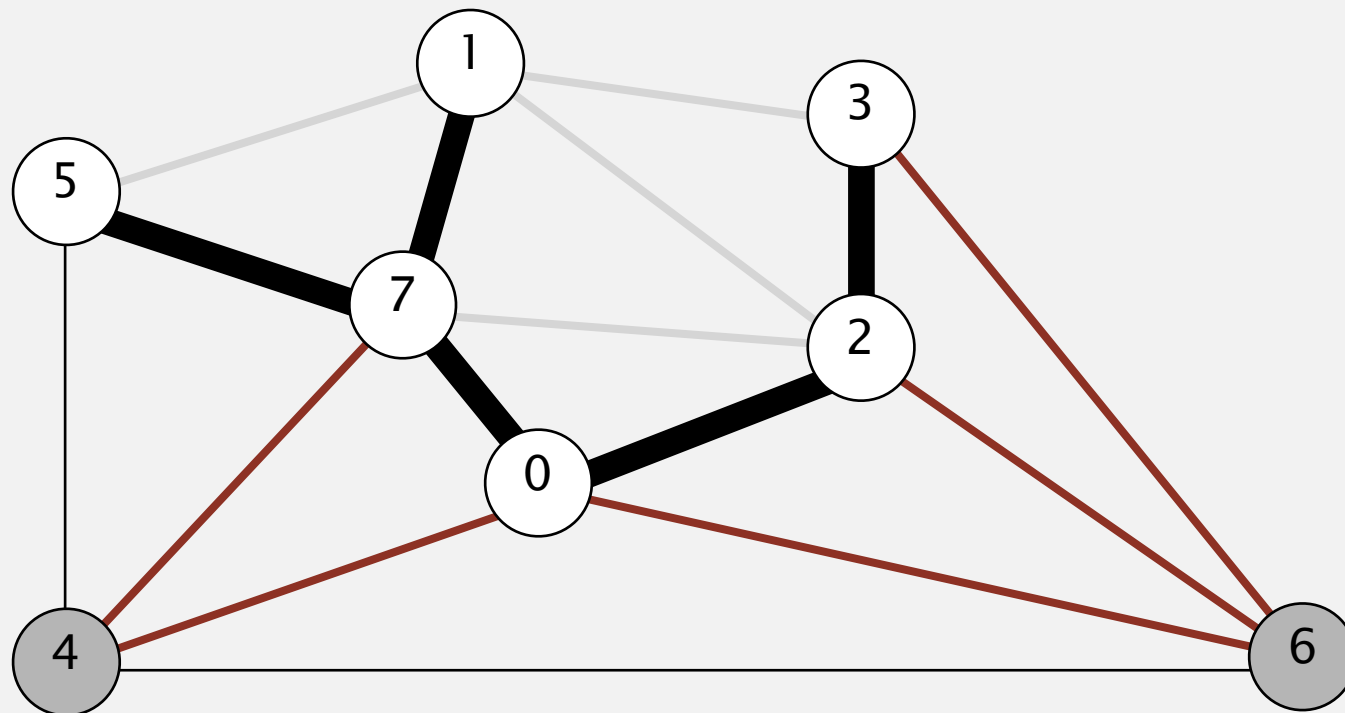- Repeat until $V - 1$ edges.

**delete 2–7 and discard obsolete edge**
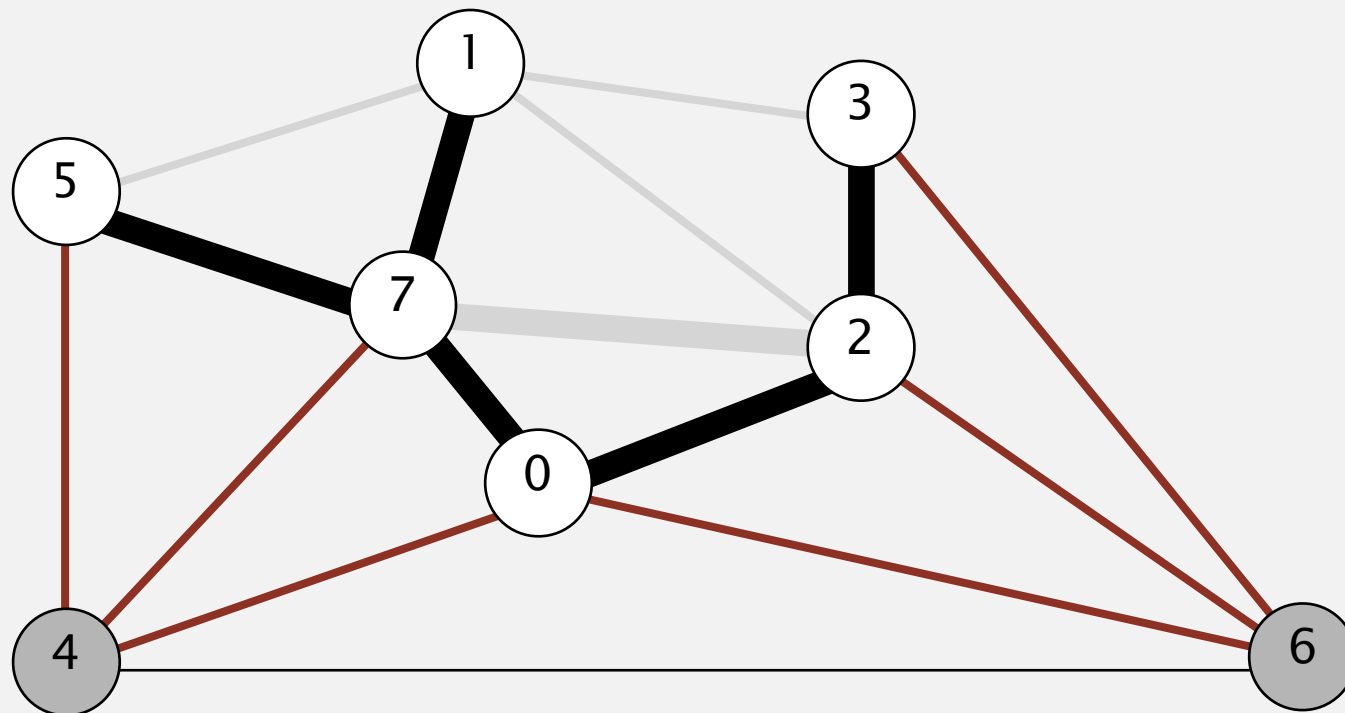


edges on PQ
(sorted by weight)

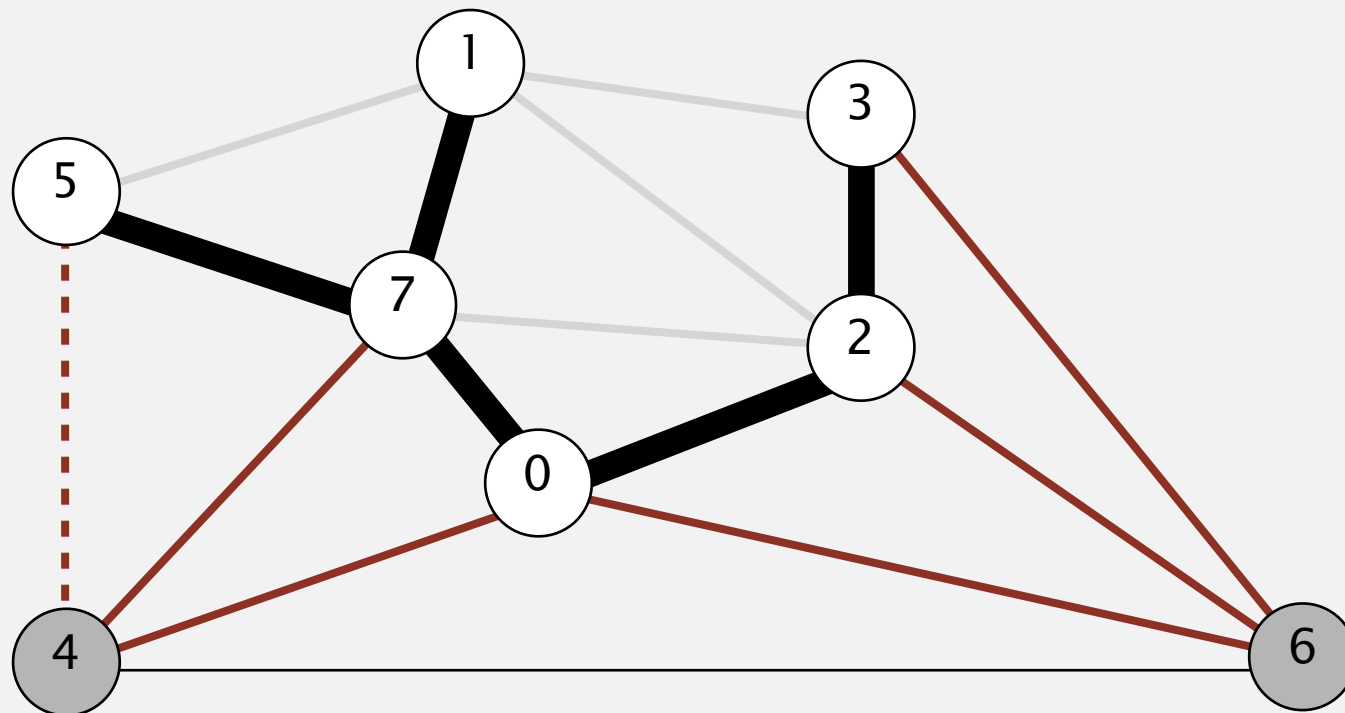| | |
|---|---|
| 2–7 | 0.34 |
| 4–5 | 0.35 |
| 1–2 | 0.36 |
| 4–7 | 0.37 |
| 0–4 | 0.38 |
| 6–2 | 0.40 |
| 3–6 | 0.52 |
| 6–0 | 0.58 |

**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**
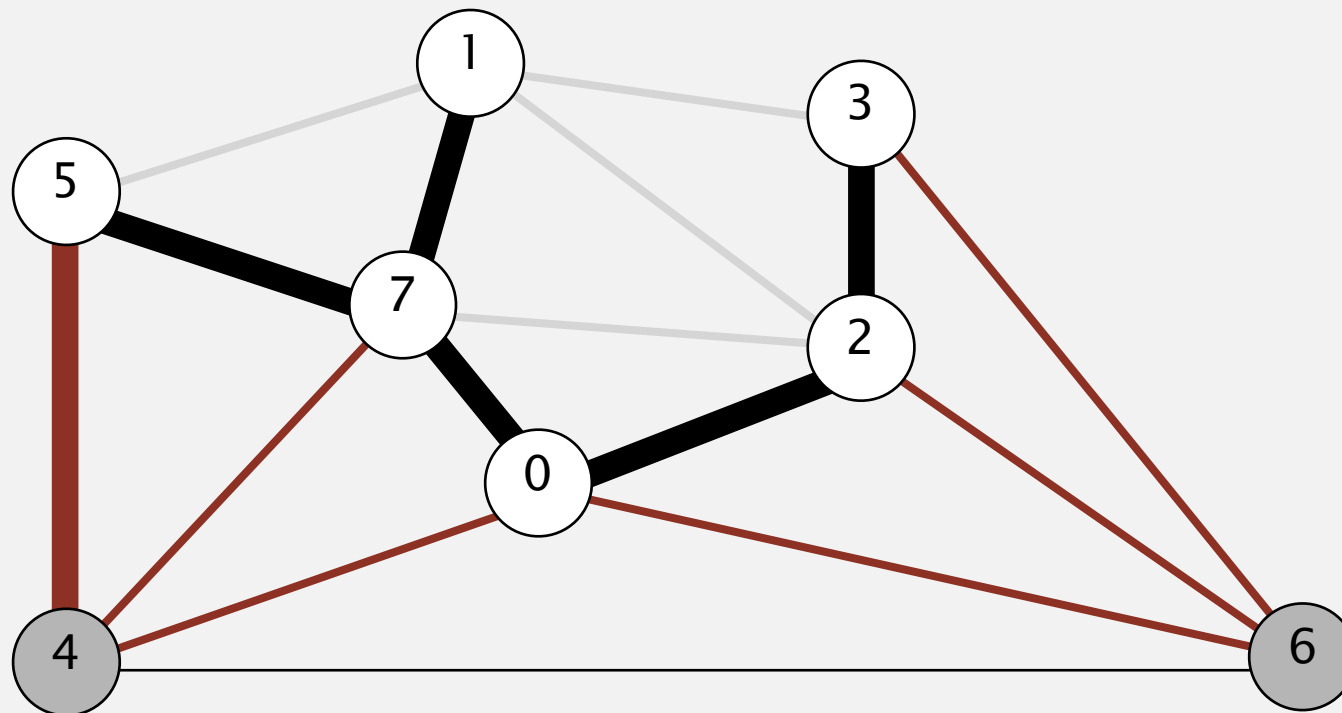


edges on PQ
(sorted by weight)

| | |
|---|---|
| 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

- 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.
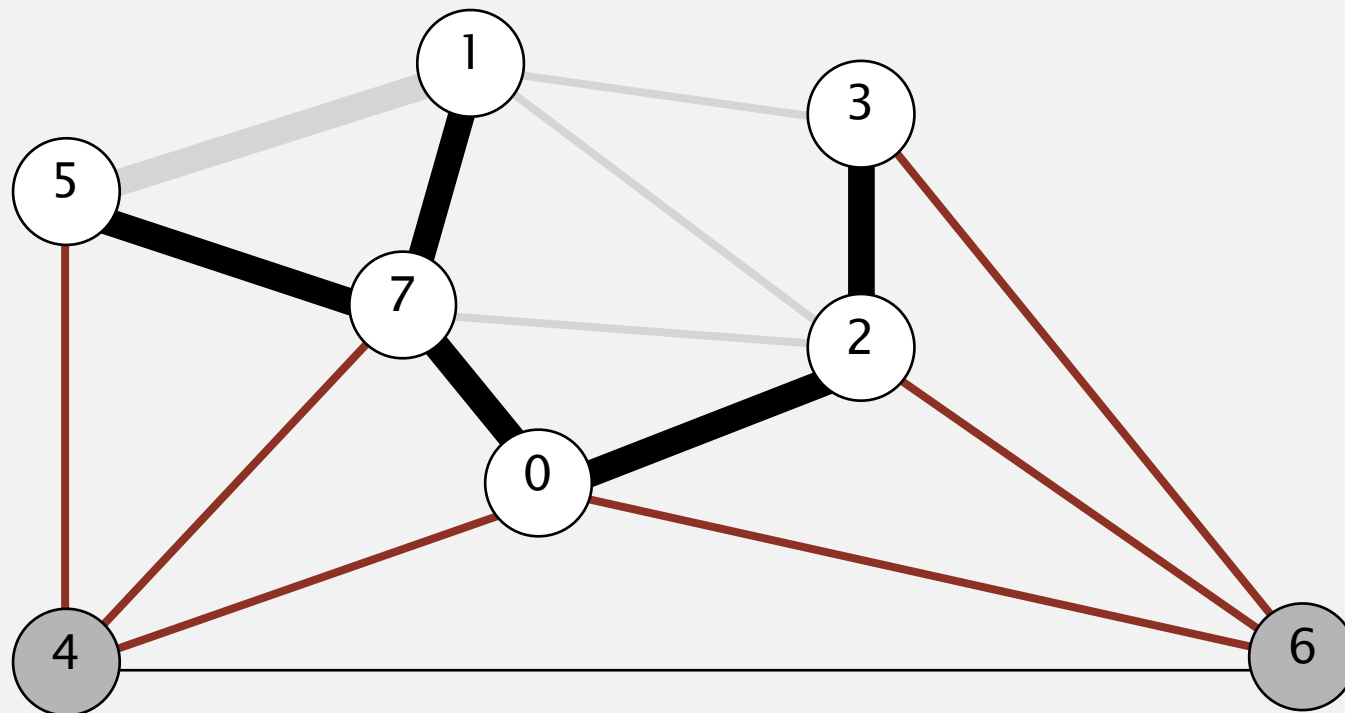
**delete 1–2 and discard obsolete edge**



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

- 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

- 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**
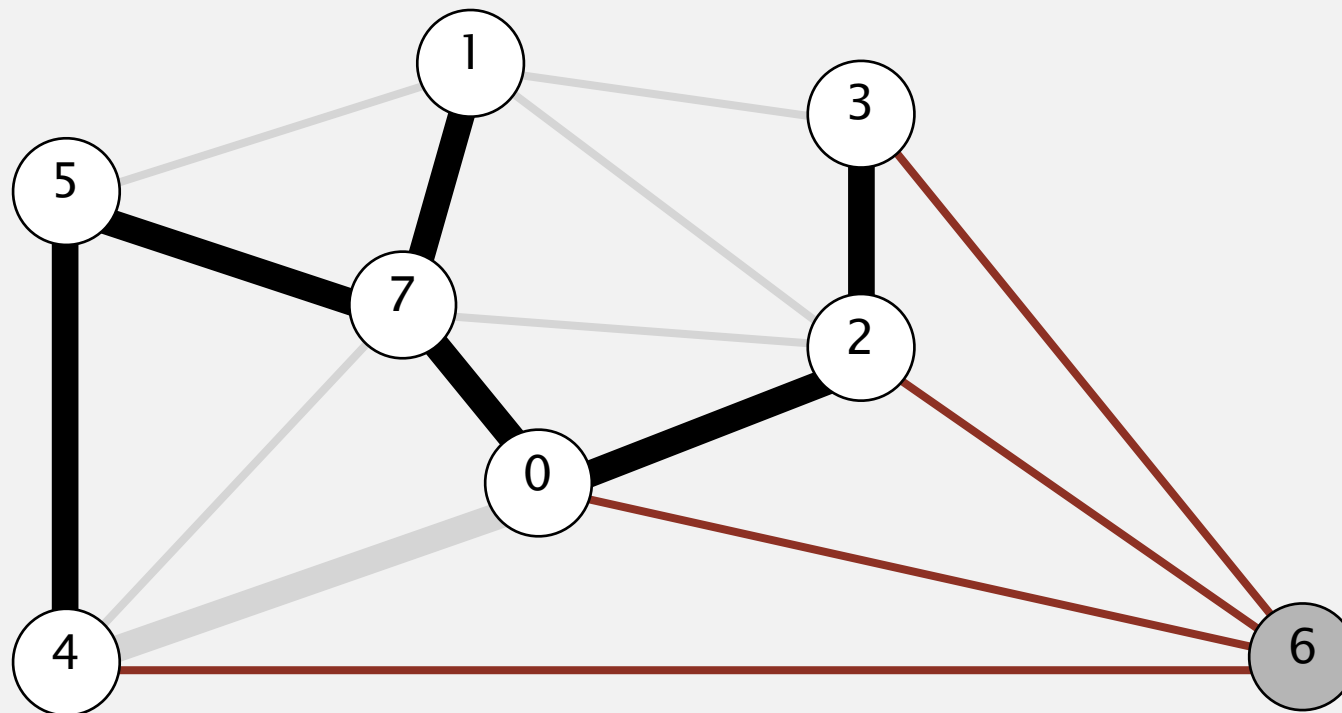


edges on PQ
(sorted by weight)

| | |
|---|---|
| 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**
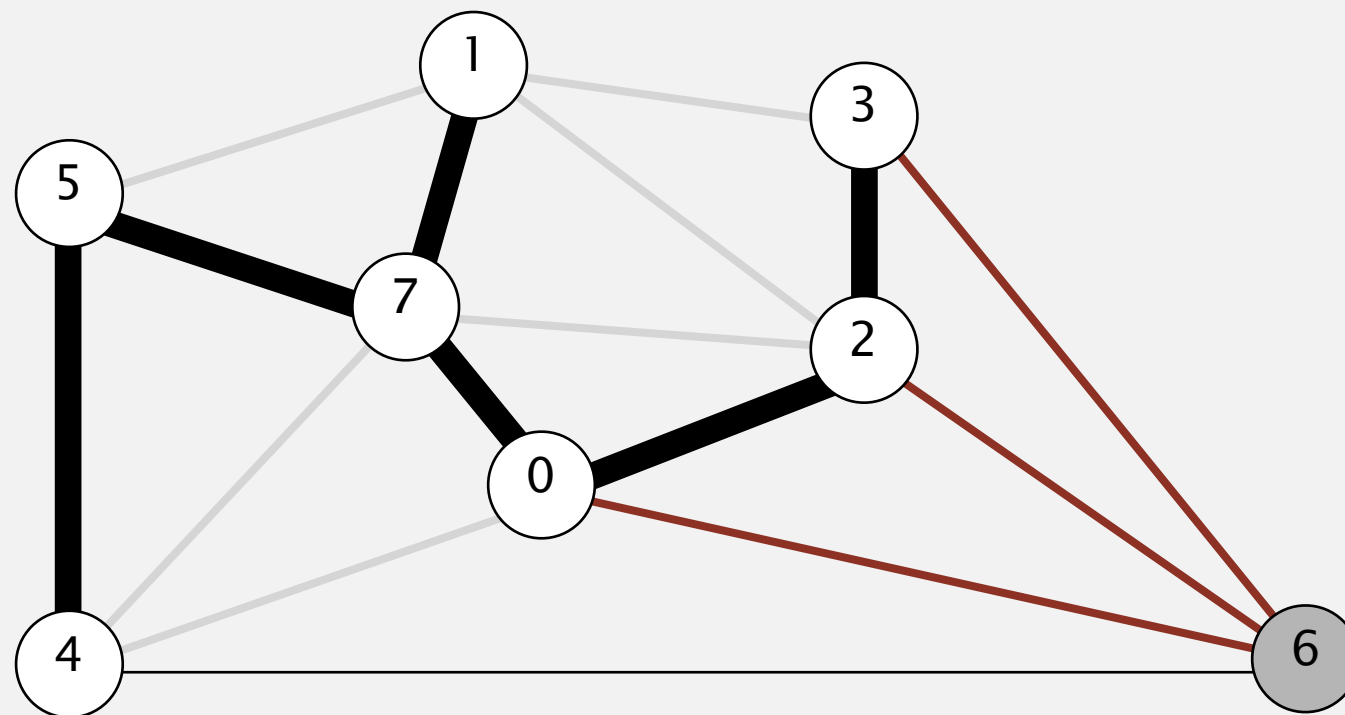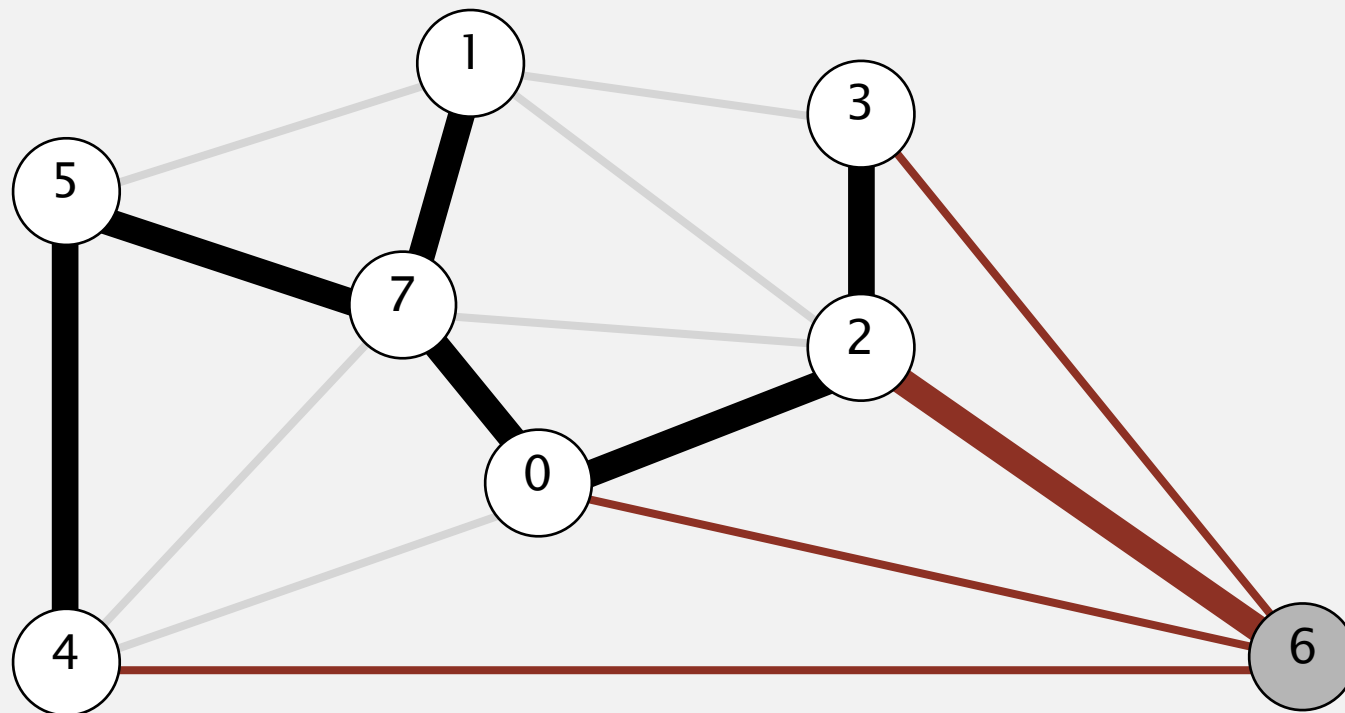


edges on PQ
(sorted by weight)

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

**MST edges**
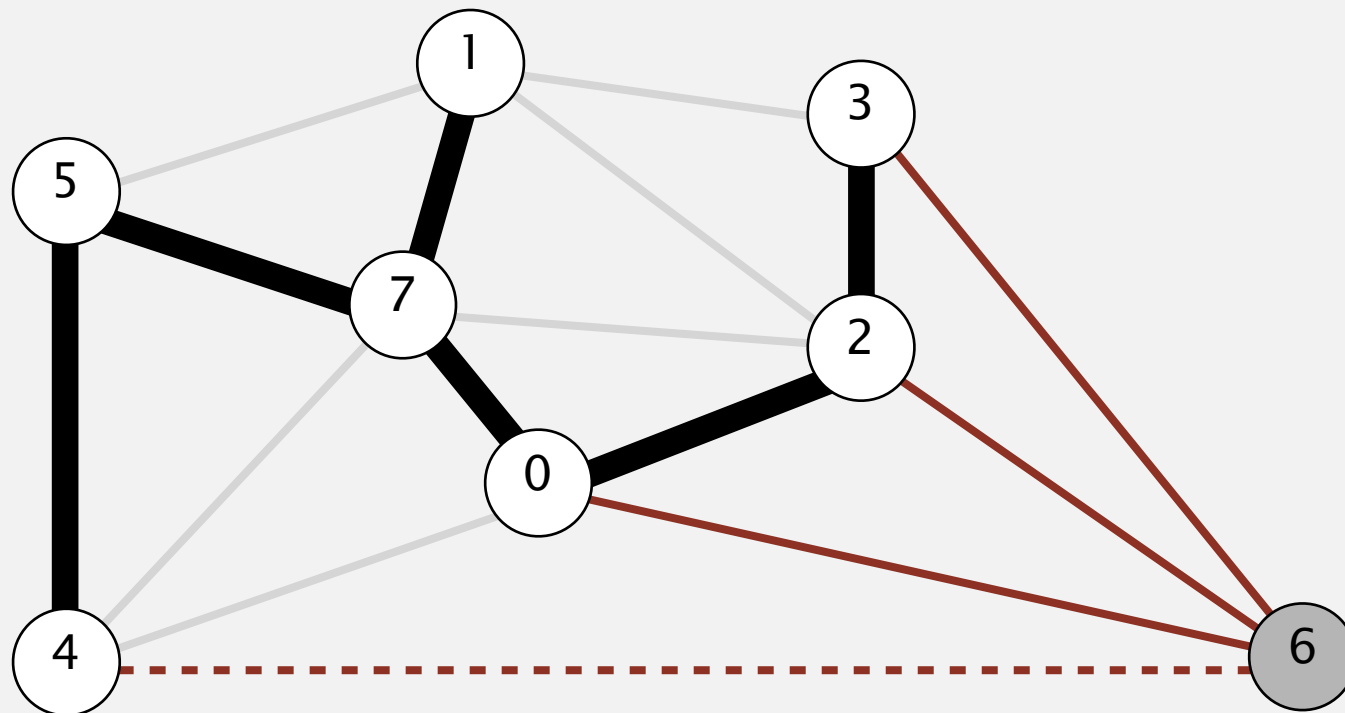
0–7   1–7   0–2   2–3   5–7   4–5

# 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)

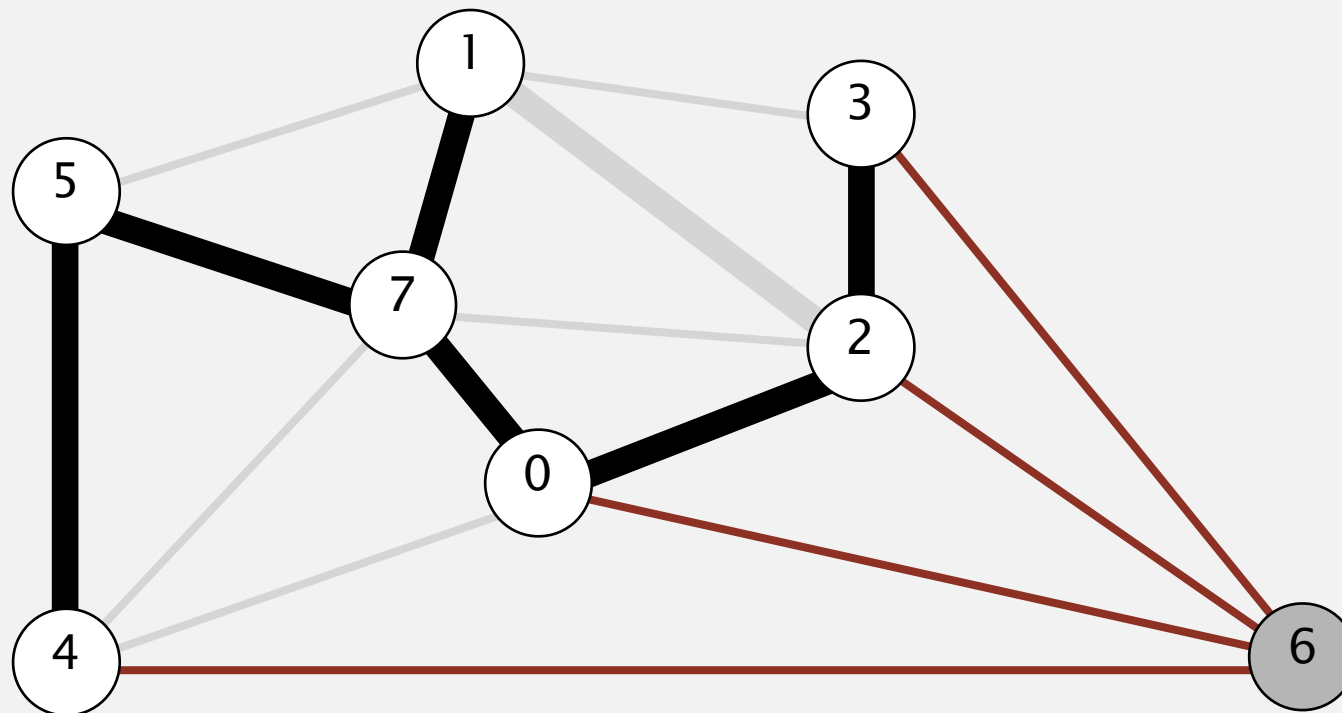| | |
|---|---|
| 3–6 | 0.52 |
| 6–0 | 0.58 |
| 6–4 | 0.93 |
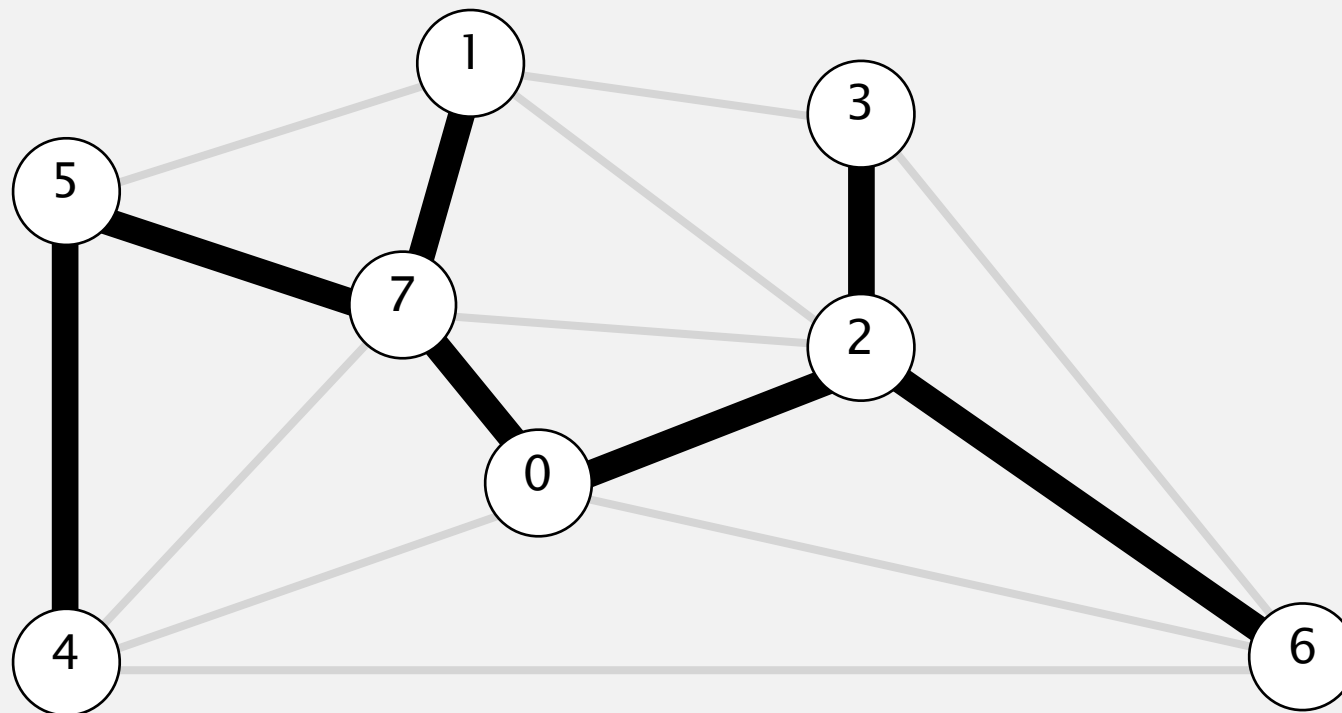
**MST edges**

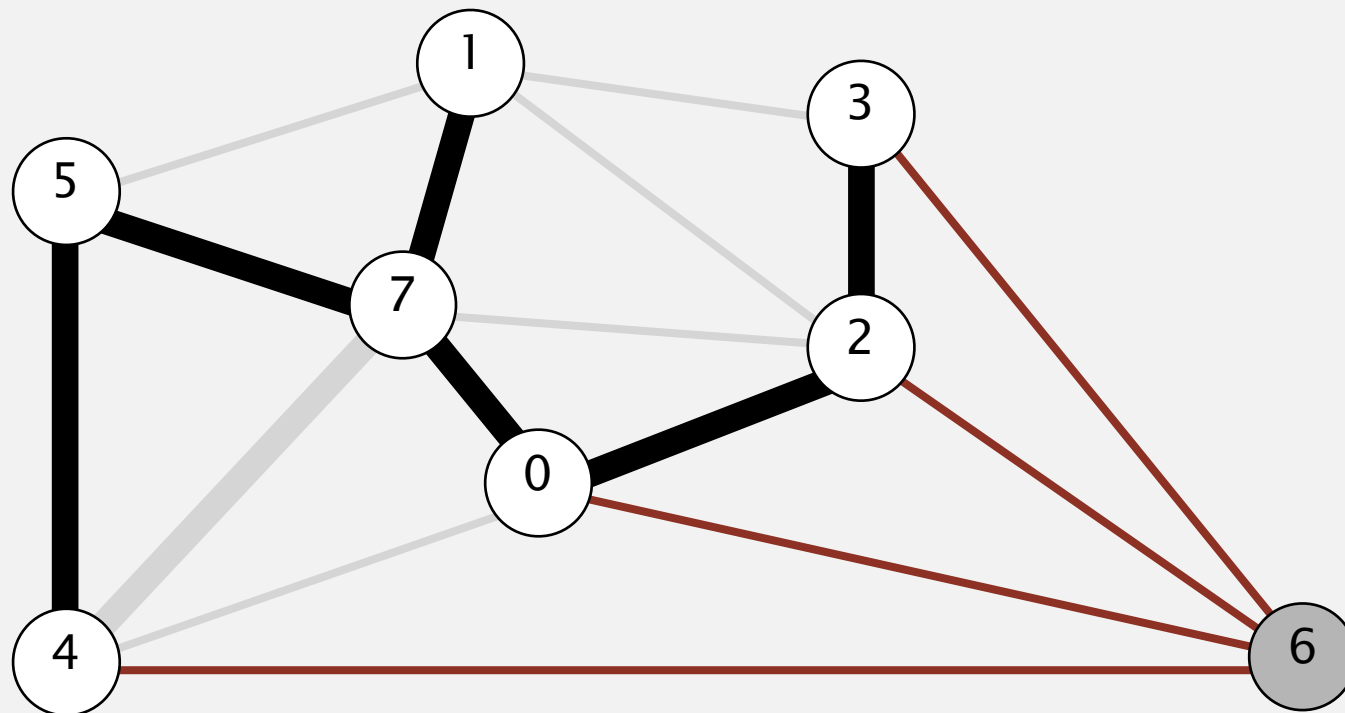0–7   1–7   0–2   2–3   5–7   4–5   6–2

# 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**
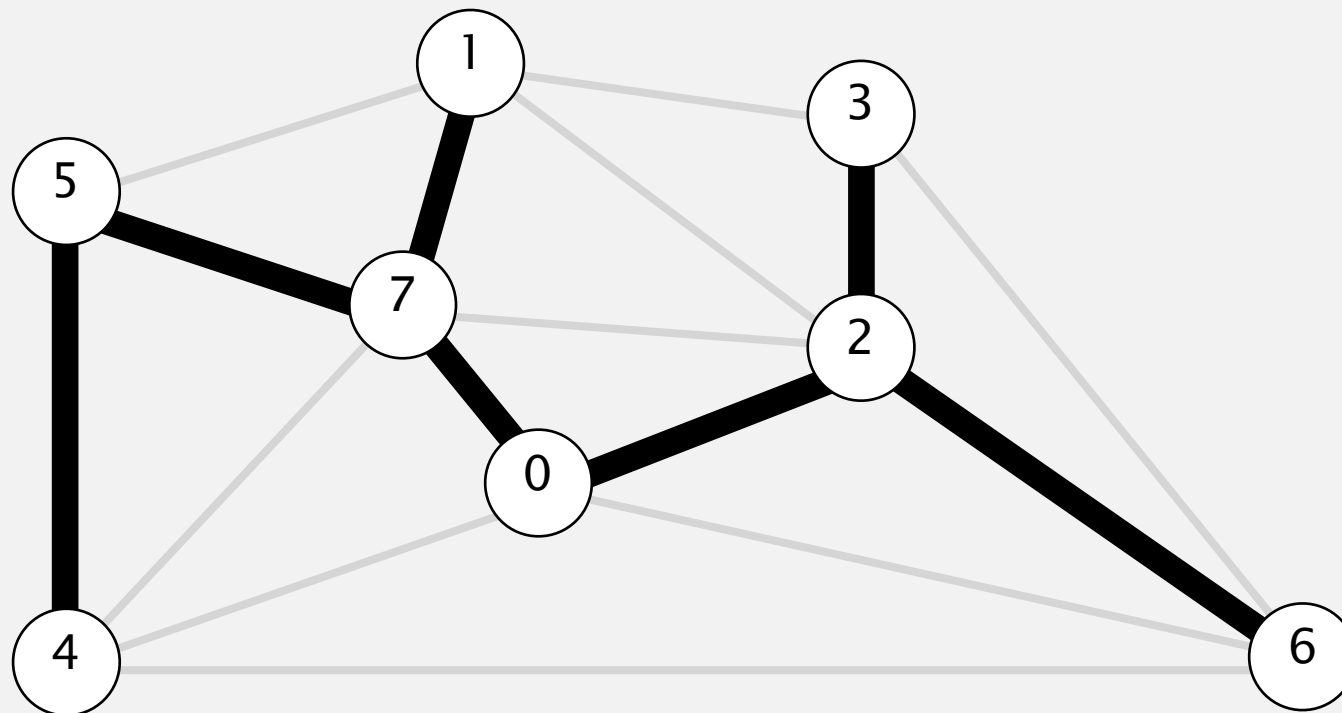


edges on PQ
(sorted by weight)

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

- 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

```java
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);                            ← assume G is connected

         while (!pq.isEmpty() && mst.size() < G.V() - 1)
         {
             Edge e = pq.delMin();               ← repeatedly delete the
                                                   min weight edge e = v–w from PQ
             int v = e.either(), w = e.other(v);
             if (marked[v] && marked[w]) continue;   ← ignore if both endpoints in T
             mst.enqueue(e);                     ← add edge e to tree
             if (!marked[v]) visit(G, v);
             if (!marked[w]) visit(G, w);        ← 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

# Prim's algorithm:  visualization
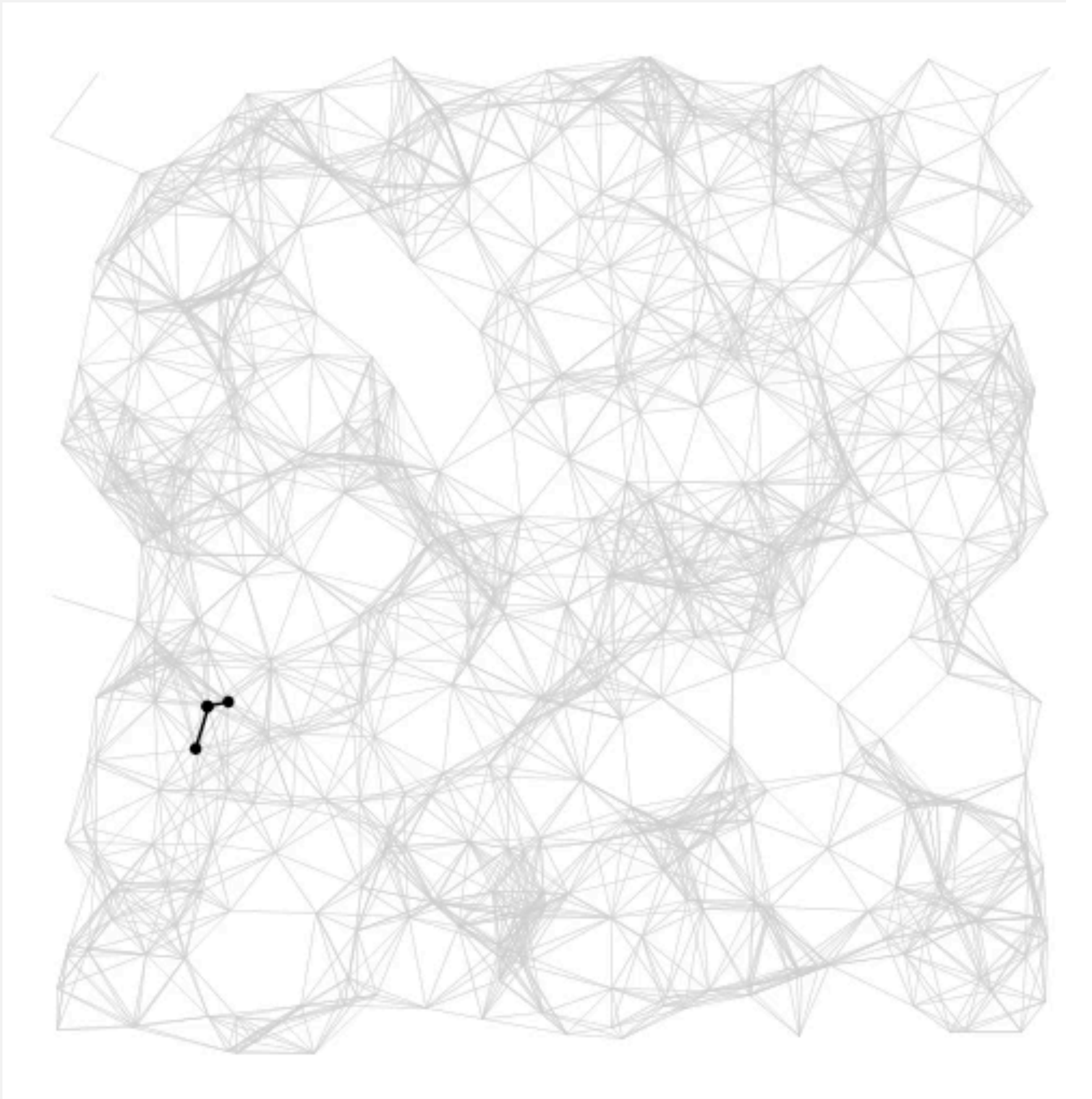
# Lazy Prim's algorithm:  running time

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** | $E$ | $\log E$ |

# Prim's algorithm: eager implementation

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

Observation. For each vertex $v$, need only shortest edge connecting $v$ to $T$.

- MST includes at most one edge connecting $v$ to $T$. Why?
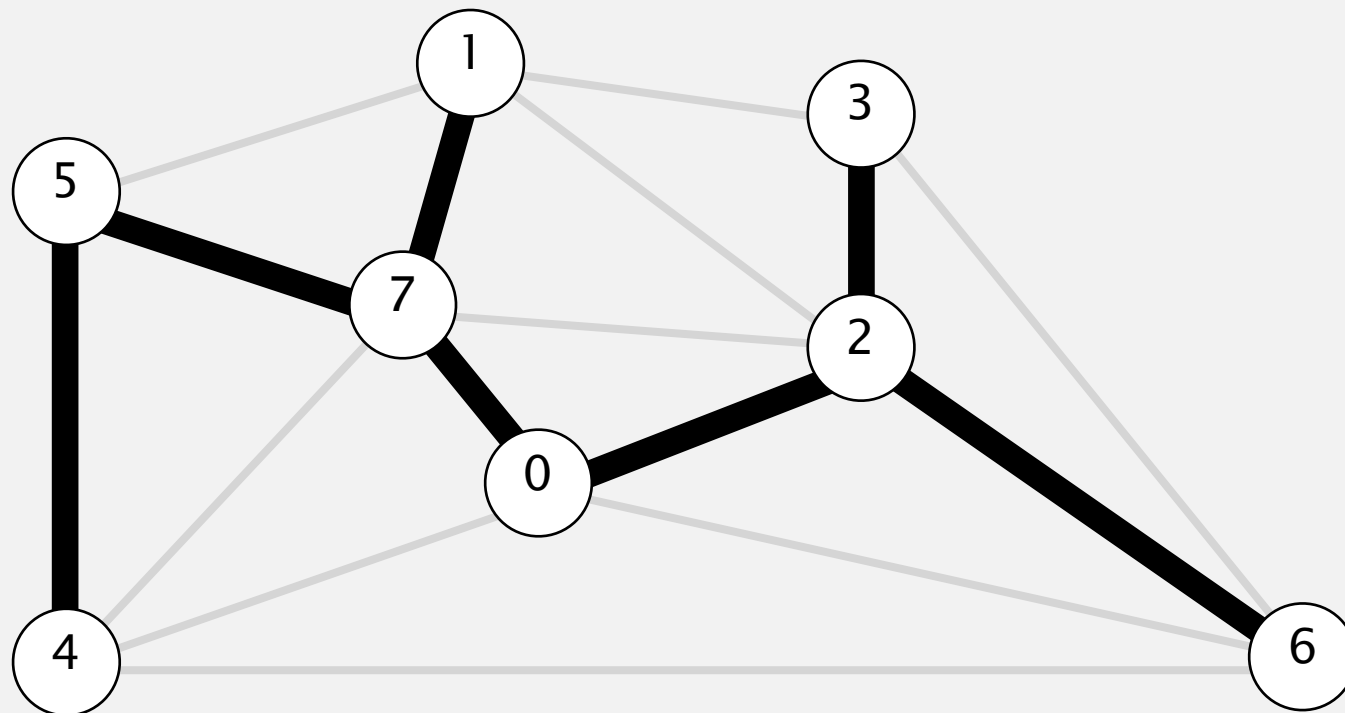
# Prim's algorithm:  eager implementation demo

- 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: eager 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.
- 

```
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  0.39
0-4  0.38
6-2  0.40
3-6  0.52
6-0  0.58
6-4  0.93
```



| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | –        | –        |

# Prim's algorithm: eager 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.
-

```
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   0.39
0-4   0.38
6-2   0.40
3-6   0.52
6-0   0.58
6-4   0.93
```



| v | edgeTo[] | distTo[] |
|---|----------|----------|
| → 0 | – | – |
| 7 | 0–7 | 0.16 |
| 2 | 0–2 | 0.26 |
| 4 | 0–4 | 0.38 |
| 6 | 6–0 | 0.58 |

vertices on PQ
(sorted by weight)

add vertices 7, 2, 4, and 6 to PQ

- 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.
-

```
0-7  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
```



| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | –        | –        |
| 7 | 0–7      | 0.16     |
| 2 | 0–2      | 0.26     |
| 4 | 0–4      | 0.38     |
| 6 | 6–0      | 0.58     |

vertices on PQ
(sorted by weight)

- 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.
-

```
0-7  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
```



| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | –        | –        |
| 7 | 0–7      | 0.16     |
|   |          |          |
| 2 | 0–2      | 0.26     |
| 4 | 0–4      | 0.38     |
| 6 | 6–0      | 0.58     |

**MST edges**

**0–7**

- 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 vertex 1 to PQ

add vertex 5
to PQ

```
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   0.39
0-4   0.38
6-2   0.40
3-6   0.52
6-0   0.58
6-4   0.93
```
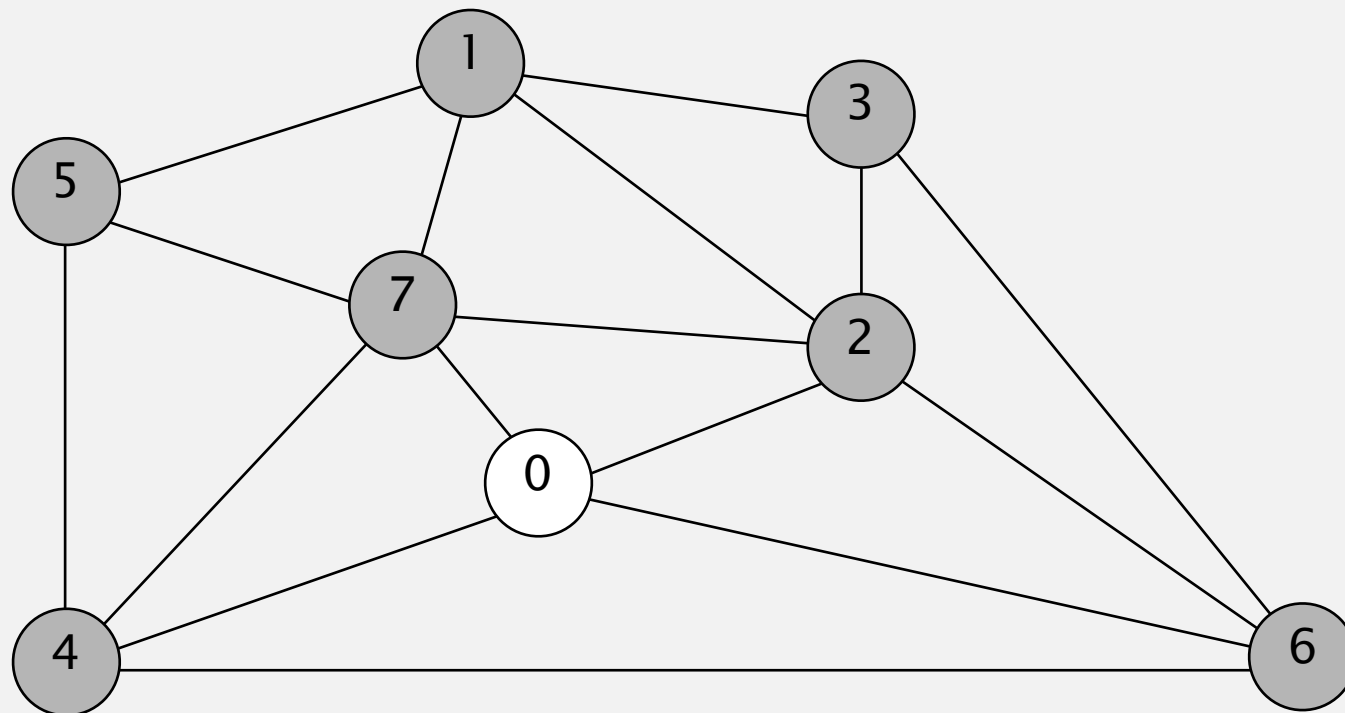
| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | –        | –        |
| 7 | 0–7      | 0.16     |
| 1 | 1–7      | 0.19     |
| 2 | 0–2      | 0.26     |
| 5 | 5–7      | 0.28     |
| 4 | 0–4      | 0.38     |
| 6 | 6–0      | 0.58     |

vertices on PQ
(sorted by weight)

already a better connection
to 2 and 4 (discard)

**MST edges**

**0–7**

- 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.
-

```
0-7   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   0.39
0-4   0.38
6-2   0.40
3-6   0.52
6-0   0.58
6-4   0.93
```
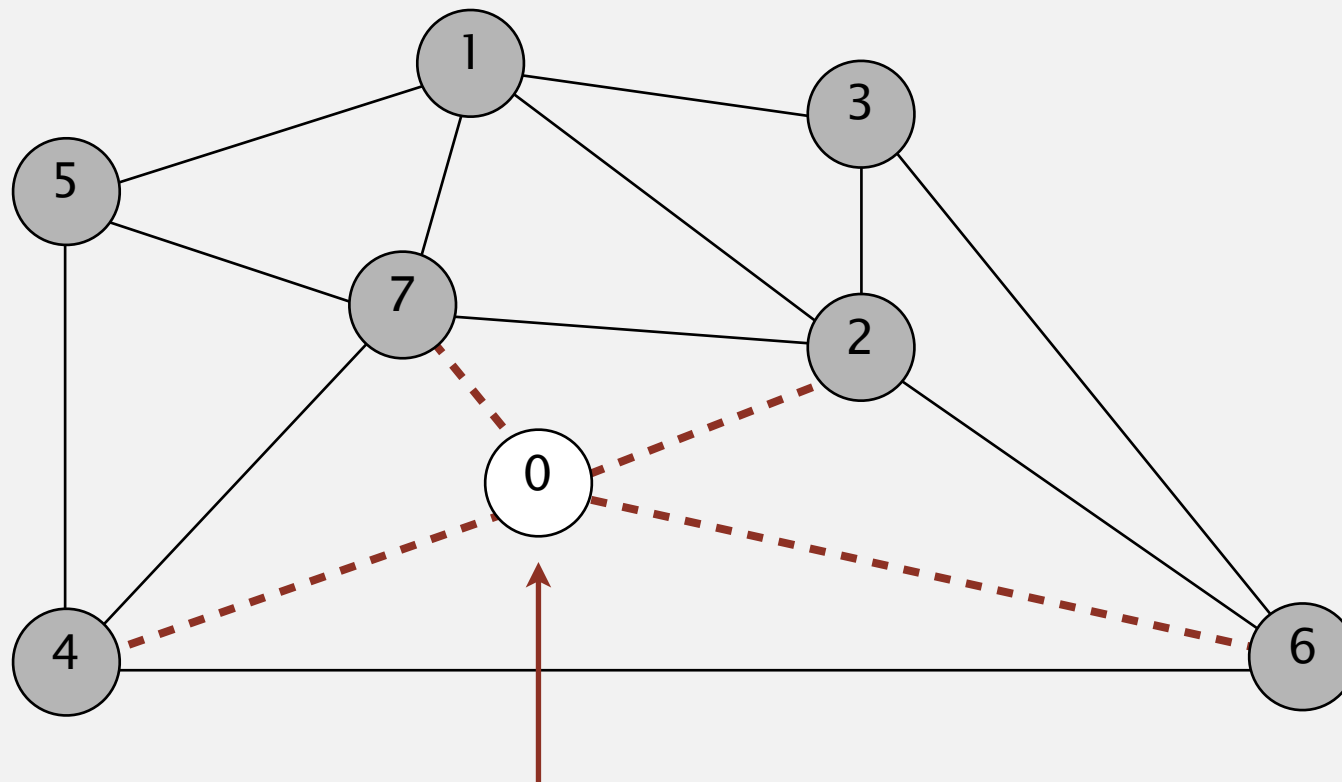


| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | –        | –        |
| 7 | 0–7      | 0.16     |
| 1 | 1–7      | 0.19     |
| 2 | 0–2      | 0.26     |
| 5 | 5–7      | 0.28     |
| 4 | 0–4      | 0.38     |
| 6 | 6–0      | 0.58     |

vertices on PQ
(sorted by weight)

**MST edges**

**0-7   1-7**

105
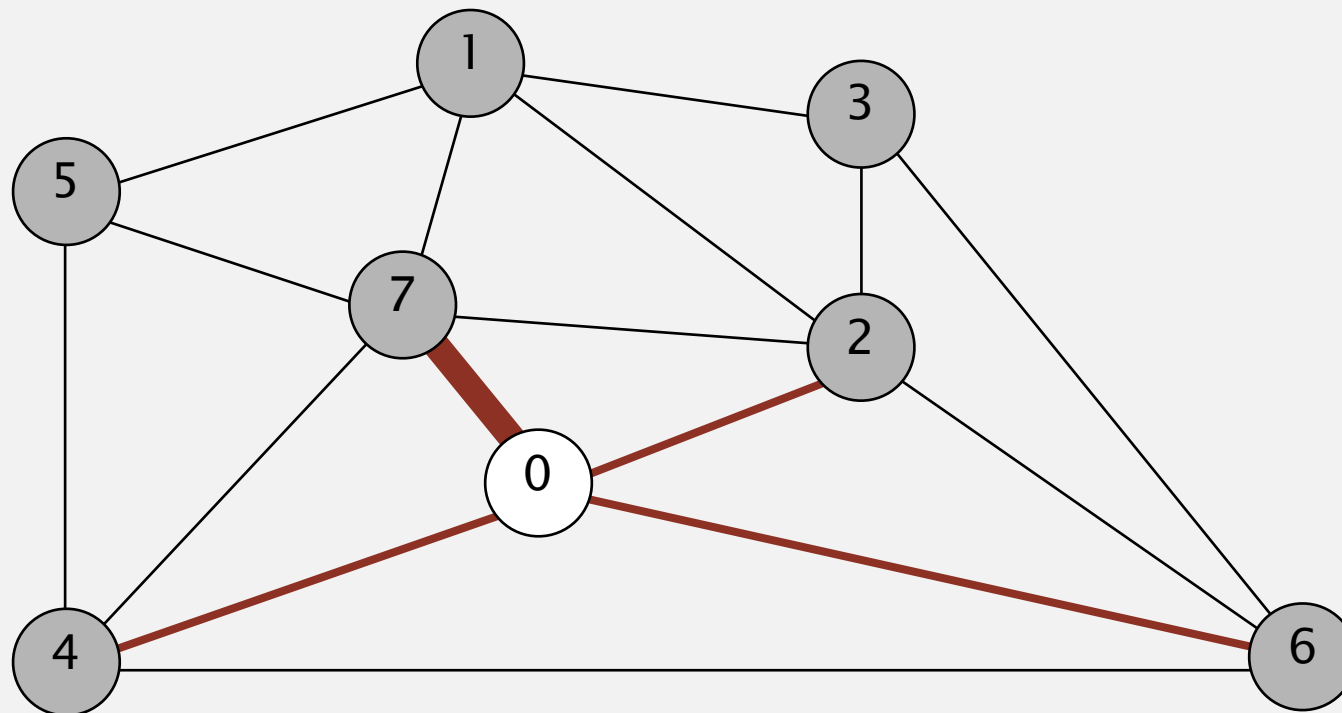
# Prim's algorithm:  eager implementation demo

- 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.
- 

```
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
```
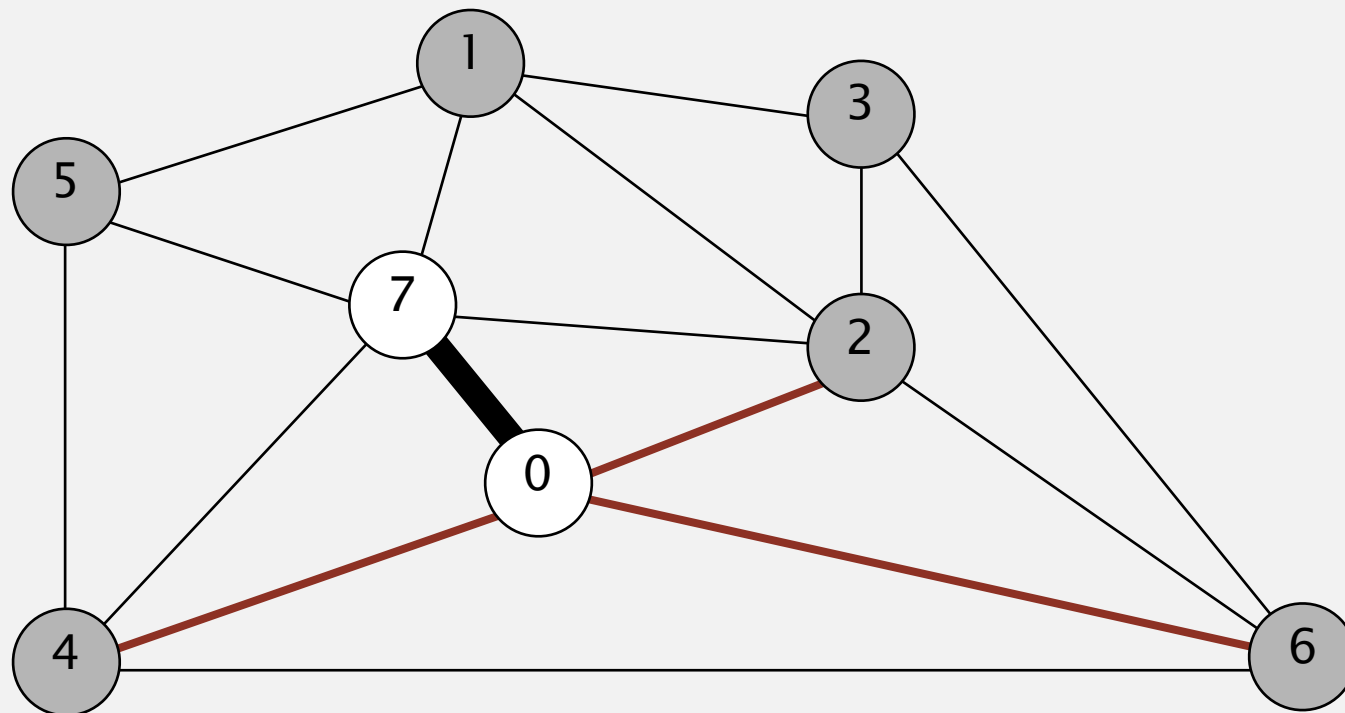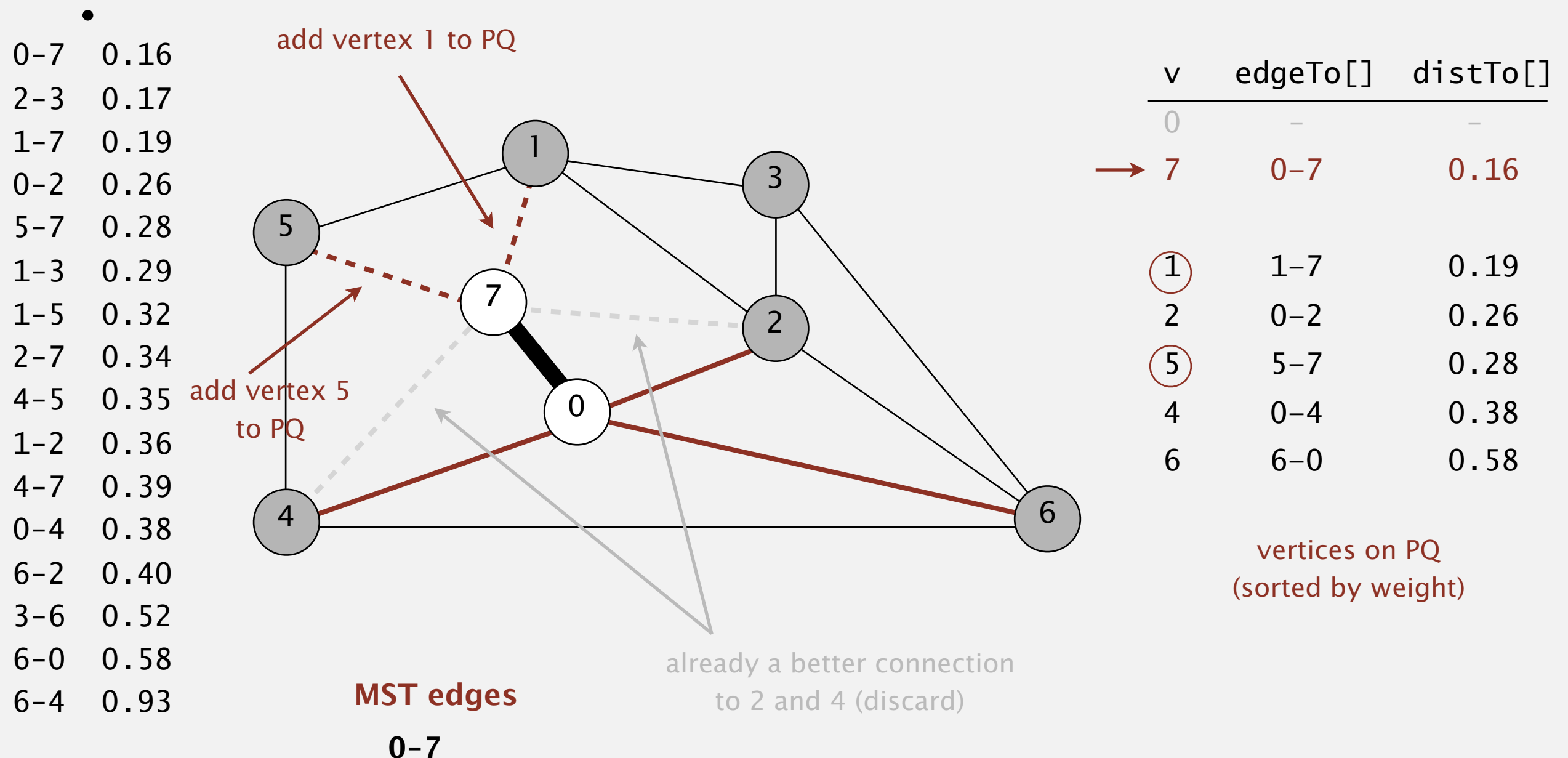
| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | –        | –        |
| 7 | 0–7      | 0.16     |
| 1 | 1–7      | 0.19     |
| 2 | 0–2      | 0.26     |
| 5 | 5–7      | 0.28     |
| 4 | 0–4      | 0.38     |
| 6 | 6–0      | 0.58     |

vertices on PQ
(sorted by weight)

**MST edges**

0-7   1-7

# Prim's algorithm:  eager 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.

-



| 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 |

add vertex 3 to PQ

| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 5 | 5–7 | 0.28 |
| ③ | 1–3 | 0.29 |
| 4 | 0–4 | 0.38 |
| 6 | 6–0 | 0.58 |

**MST edges**

**0-7    1-7**
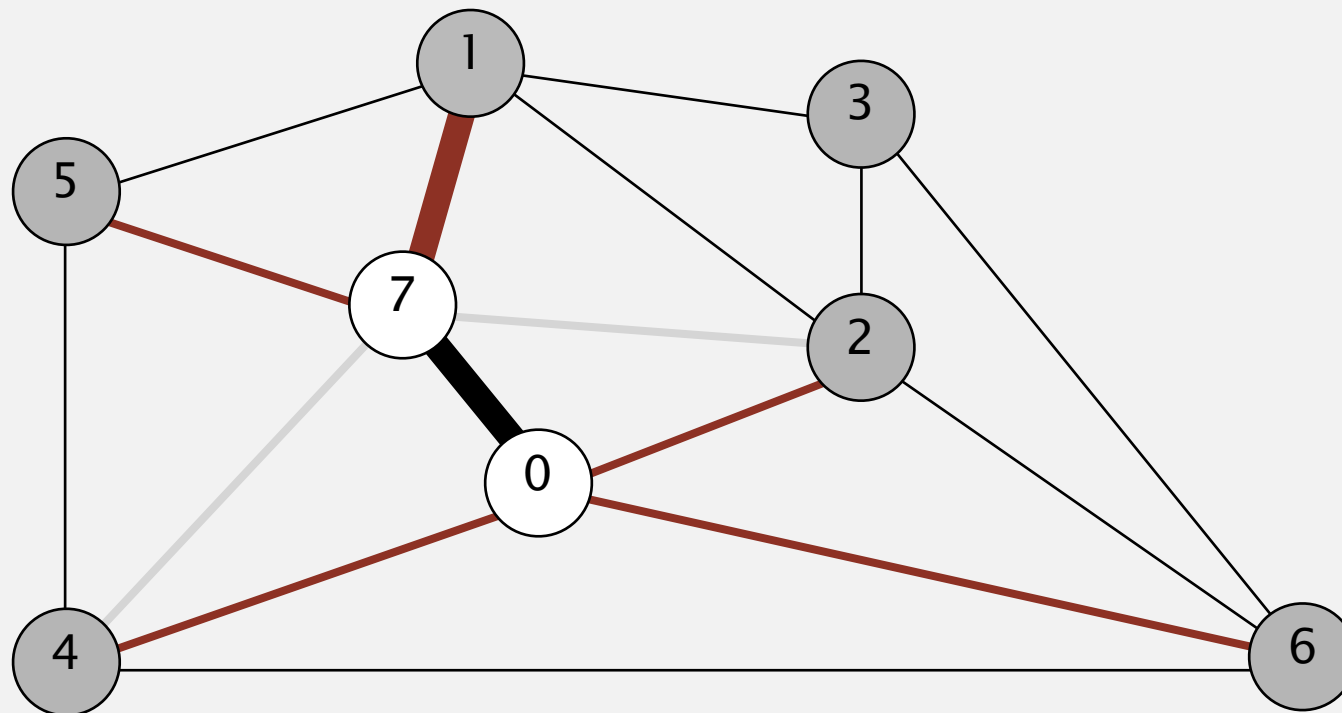
already a better connection
to 5 and 7 (discard)
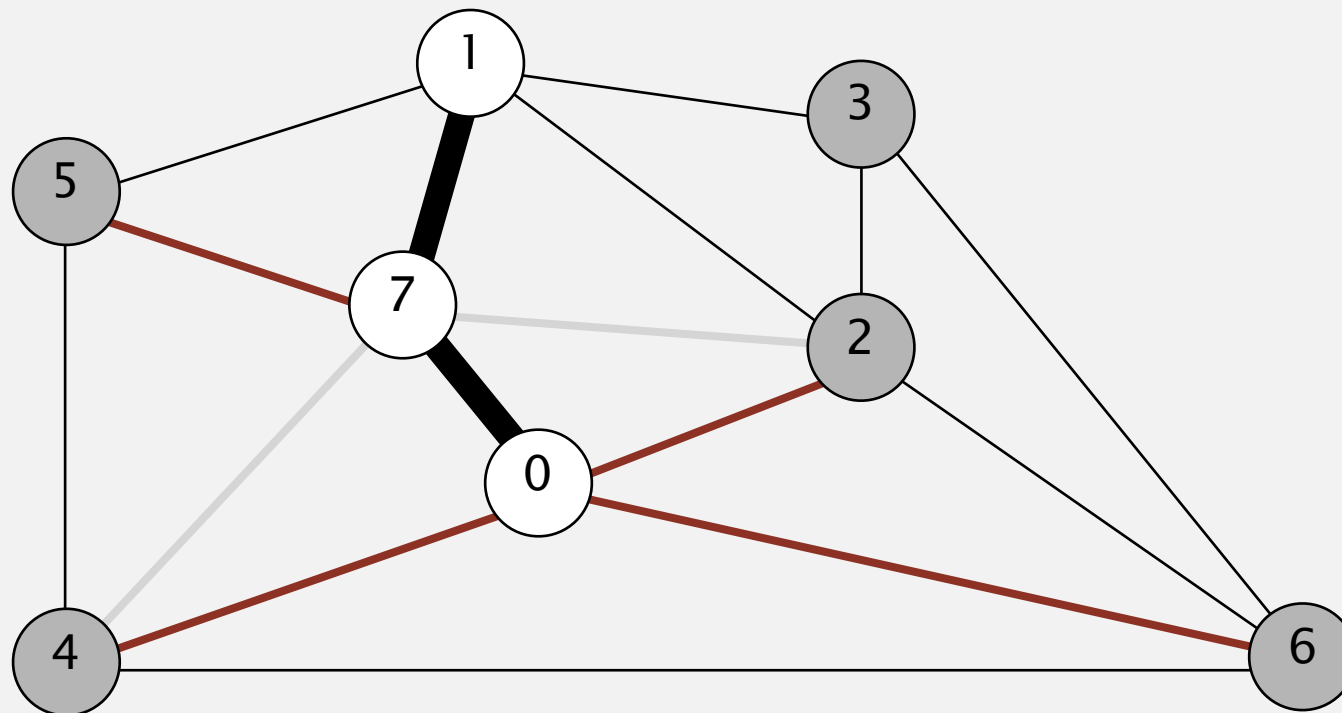
# Prim's algorithm:  eager implementation demo

- 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.
-

```
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
```



| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | –        | –        |
| 7 | 0–7      | 0.16     |
| 1 | 1–7      | 0.19     |
| 2 | 0–2      | 0.26     |
| 5 | 5–7      | 0.28     |
| 3 | 1–3      | 0.29     |
| 4 | 0–4      | 0.38     |
| 6 | 6–0      | 0.58     |

**MST edges**

**0-7   1-7**
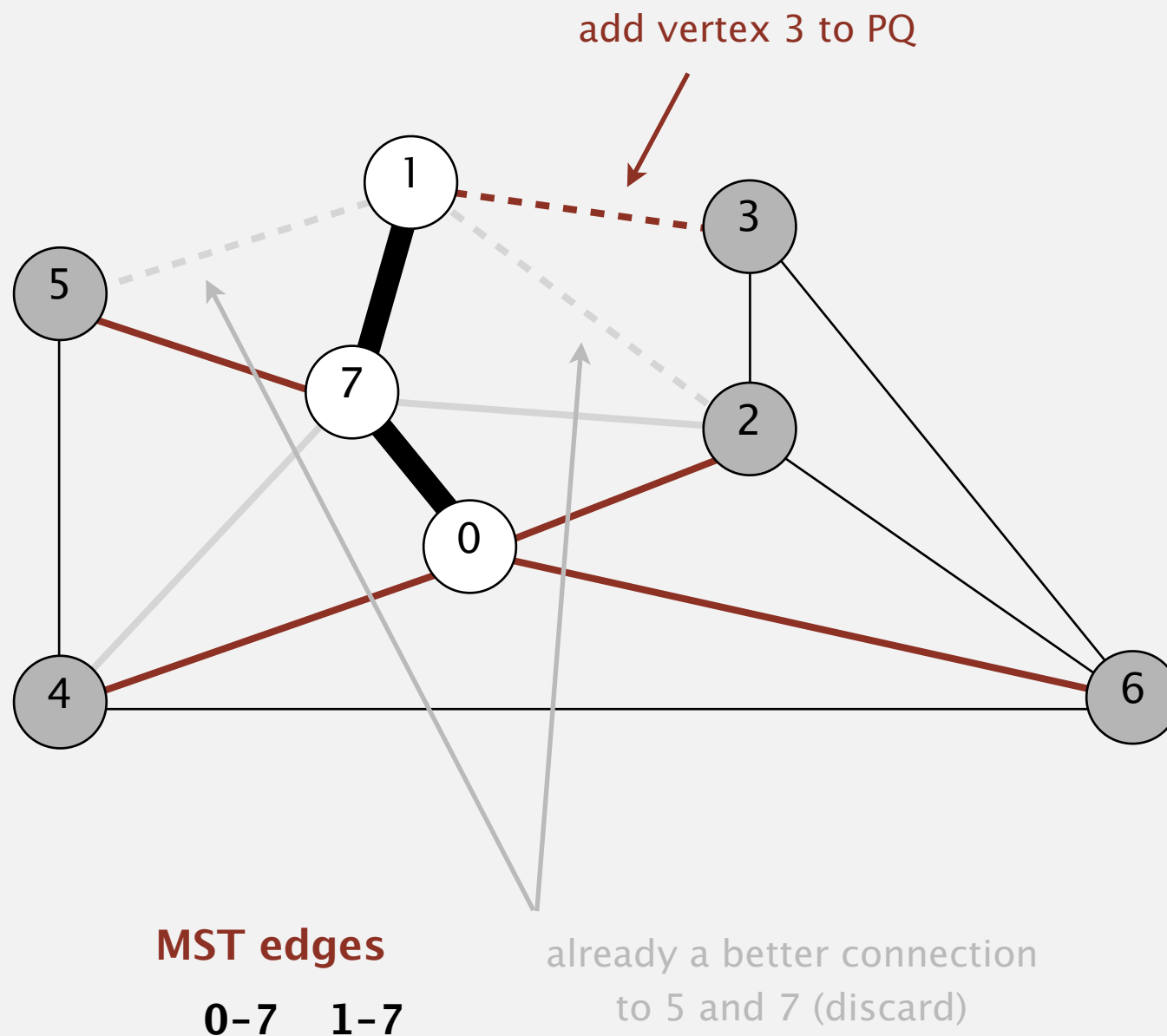
# Prim's algorithm: eager implementation demo

- 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.



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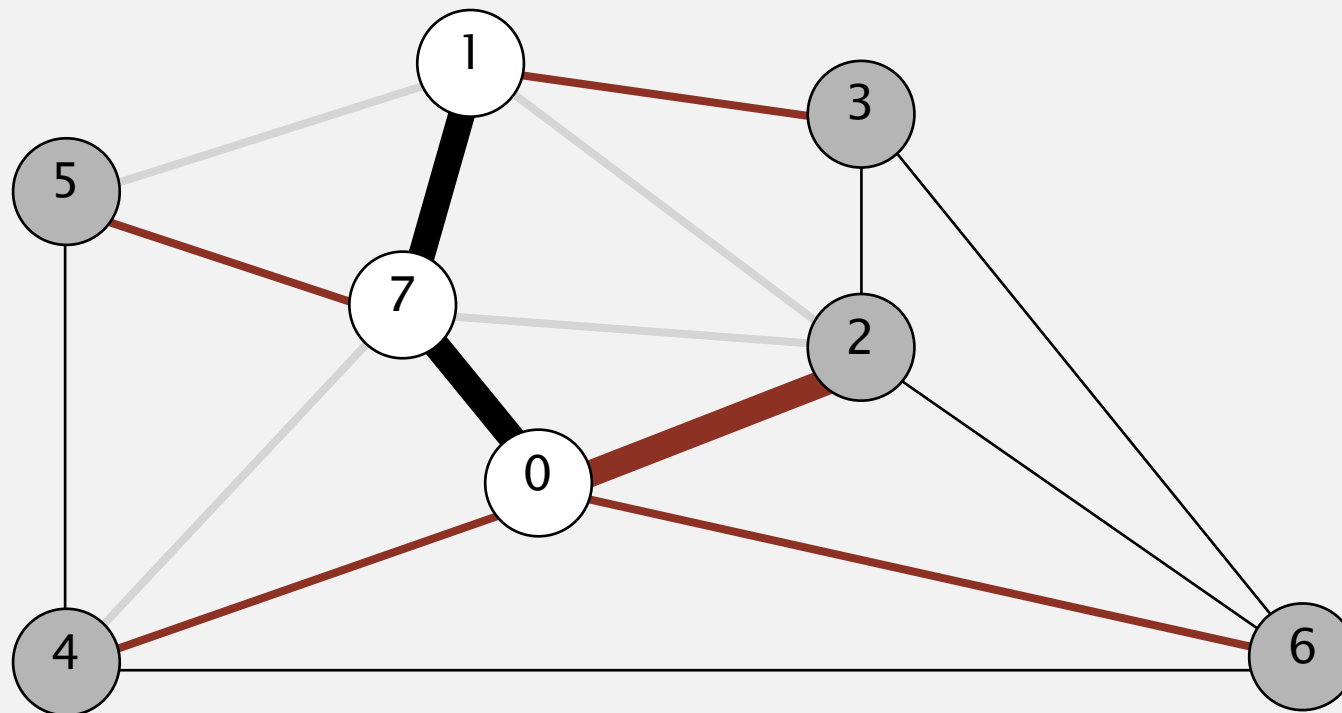4-7  0.37
0-4  0.38
6-2  0.40
3-6  0.52
6-0  0.58
6-4  0.93

| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | –        | –        |
| 7 | 0–7      | 0.16     |
| 1 | 1–7      | 0.19     |
| 2 | 0–2      | 0.26     |
| 5 | 5–7      | 0.28     |
| 3 | 1–3      | 0.29     |
| 4 | 0–4      | 0.38     |
| 6 | 6–0      | 0.58     |

**MST edges**

**0-7   1-7   0-2**

- 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.
- 



decrease key of vertex 3
from 0.29 to 0.17

| 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 |

| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 3 | 1–3  2–3 | 0.29  0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 0–4 | 0.38 |
| 6 | 6–0  6–2 | 0.58  0.40 |

now better connections
to 0 and 1 (discard)

decrease key of vertex 6
from 0.58 to 0.40

**MST edges**

**0-7   1-7   0-2**

- 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.
-



```
0-7   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
```
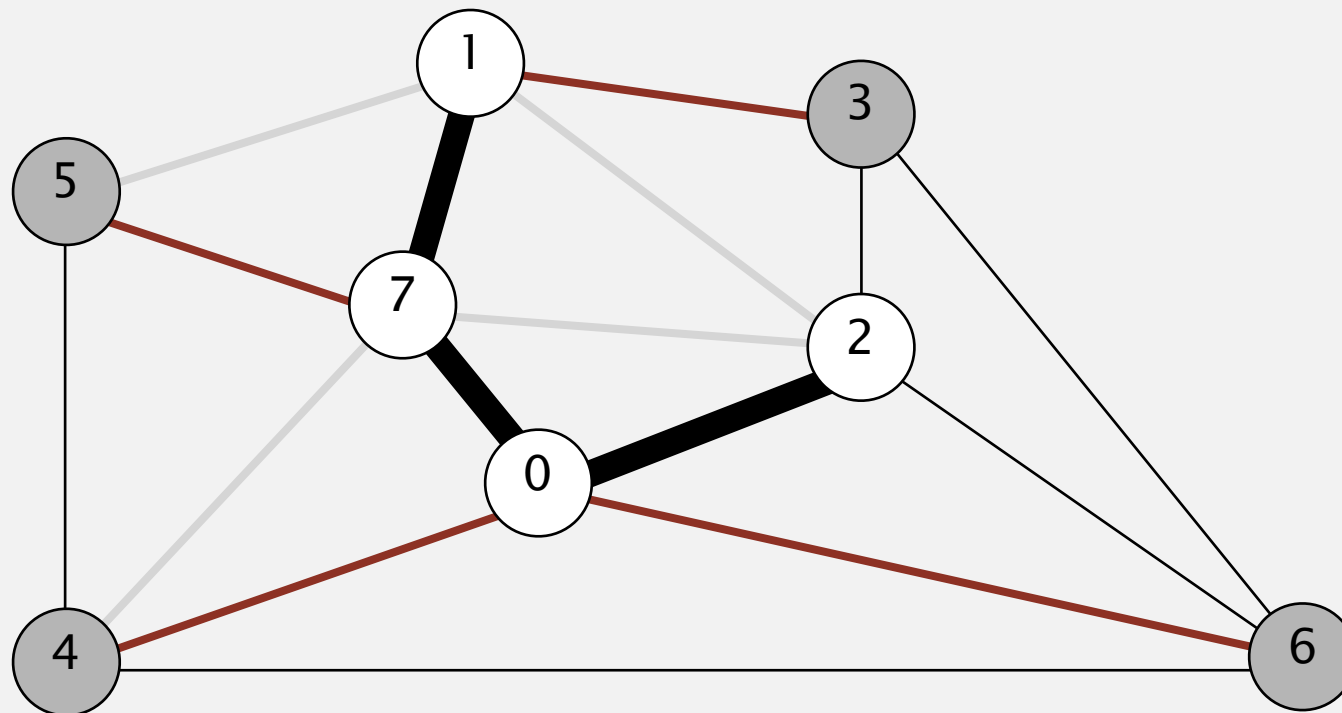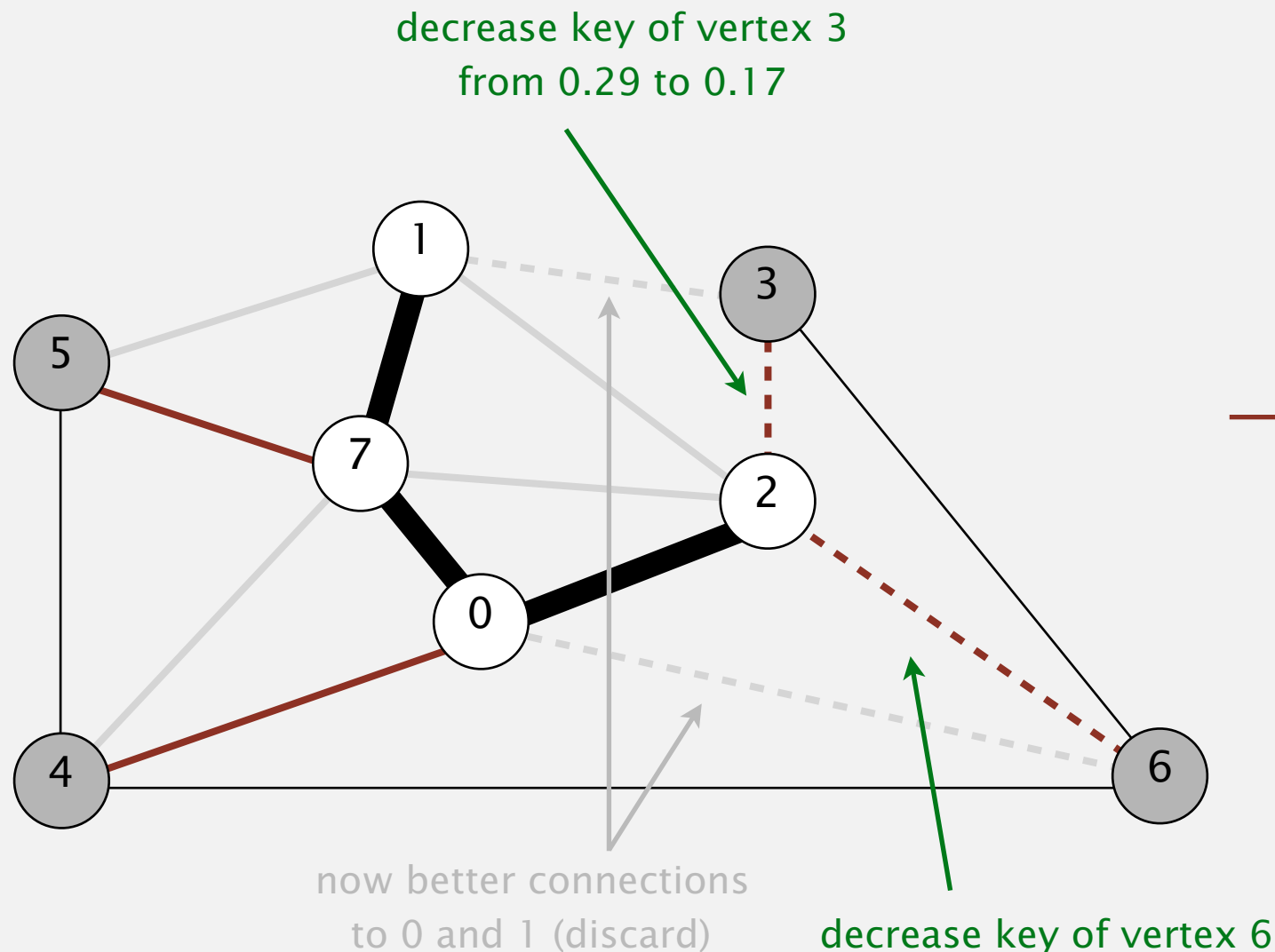
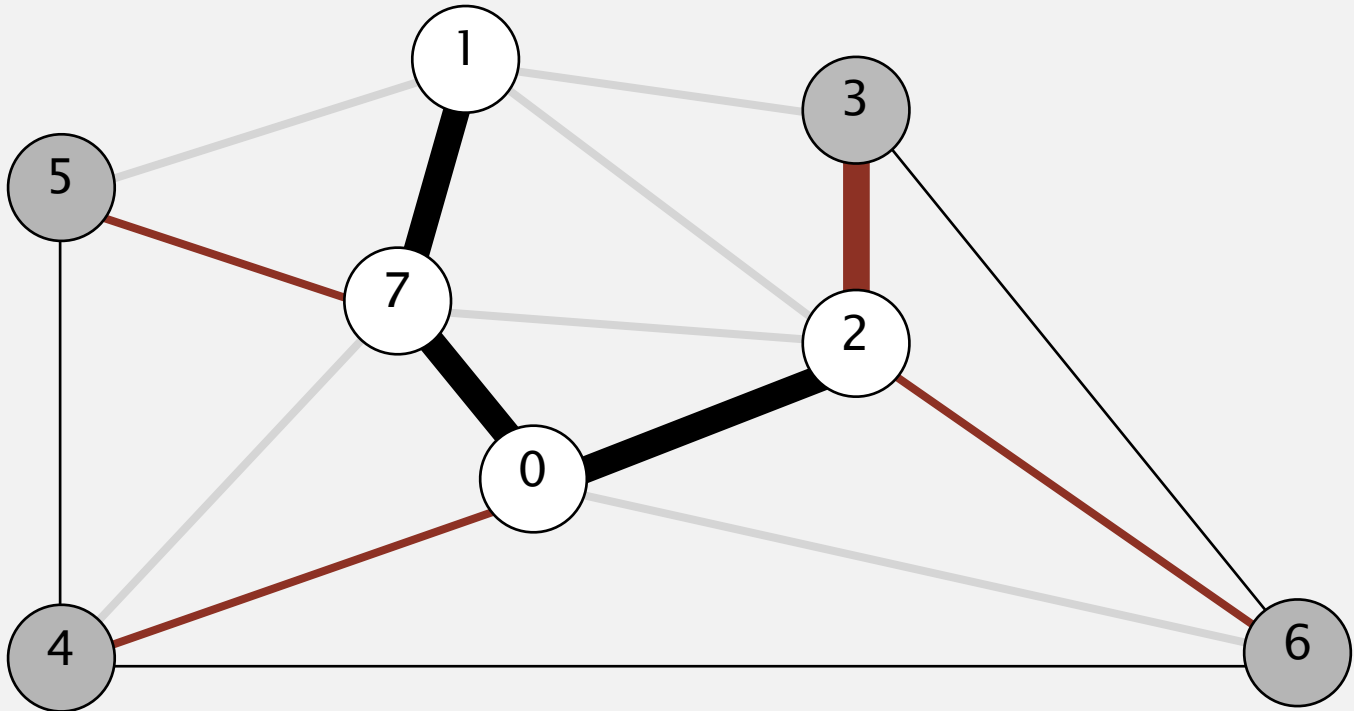| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 0–4 | 0.38 |
| 6 | 6–2 | 0.40 |

**MST edges**

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

111

- 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.
- 

| 0-7 | 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 |

| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| → 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 0–4 | 0.38 |
| 6 | 6–2 | 0.40 |

**MST edges**

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

# Prim's algorithm: eager 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.

-

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
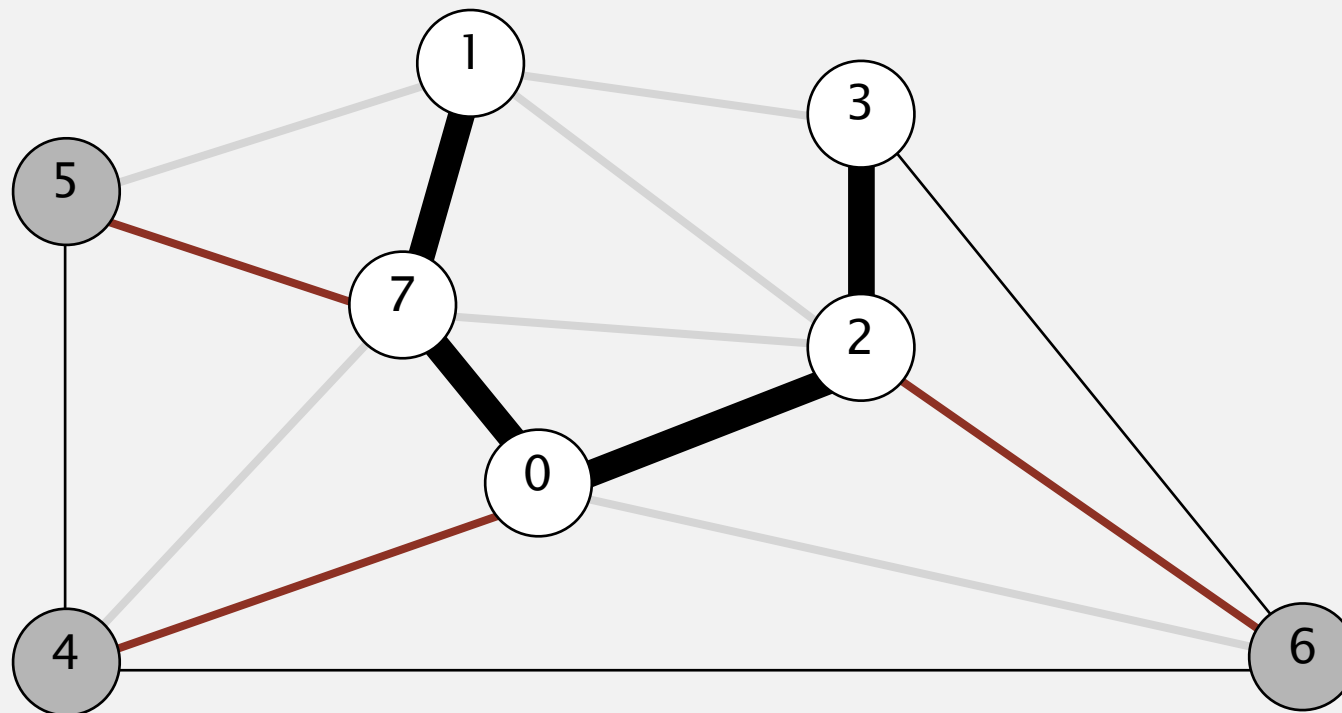6-4  0.93



| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| → 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 0–4 | 0.38 |
| 6 | 6–2 | 0.40 |

already a better connection
to 6 (discard)

**MST edges**

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

113

- 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.
- 



```
0-7  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
```
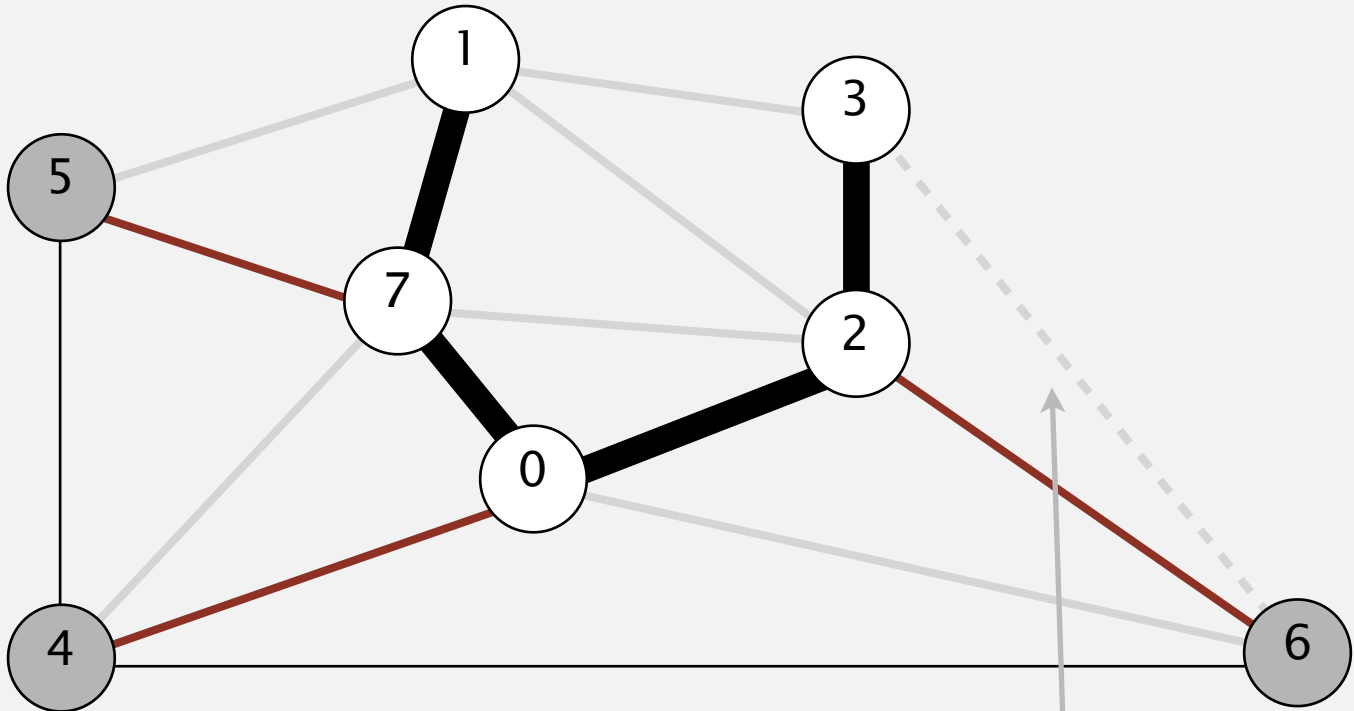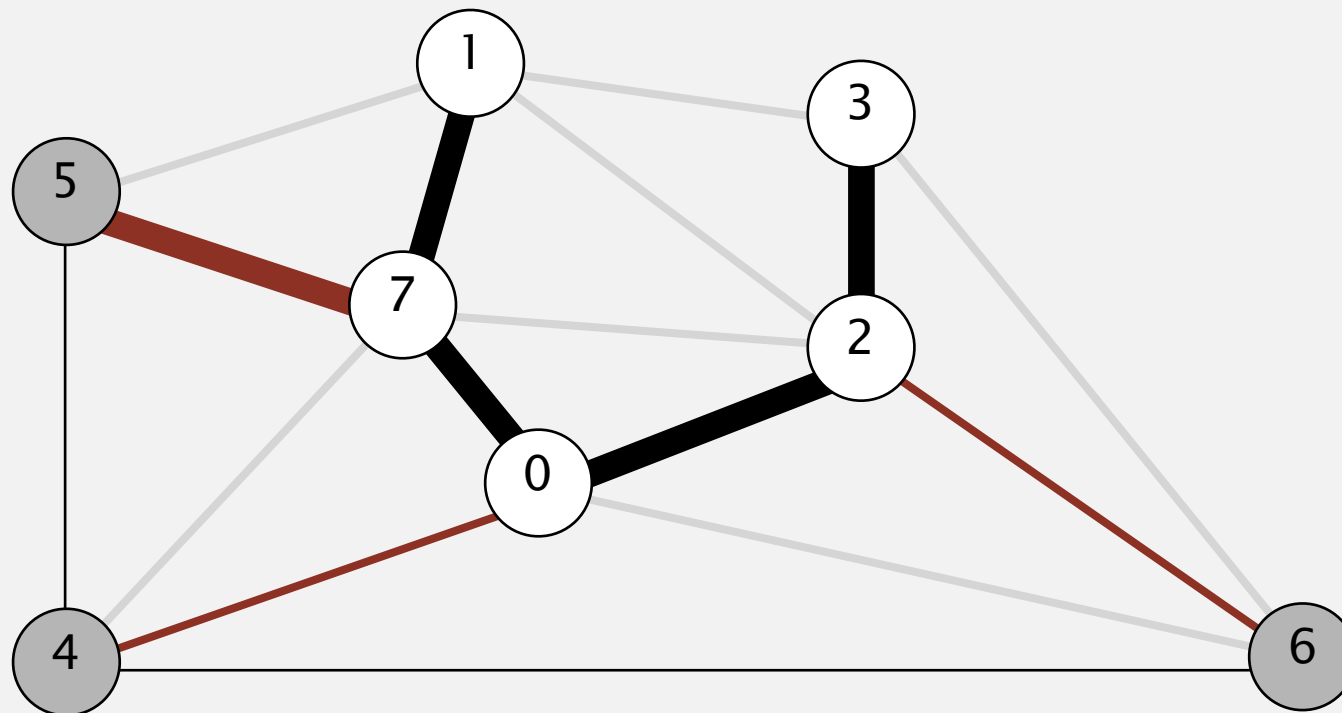
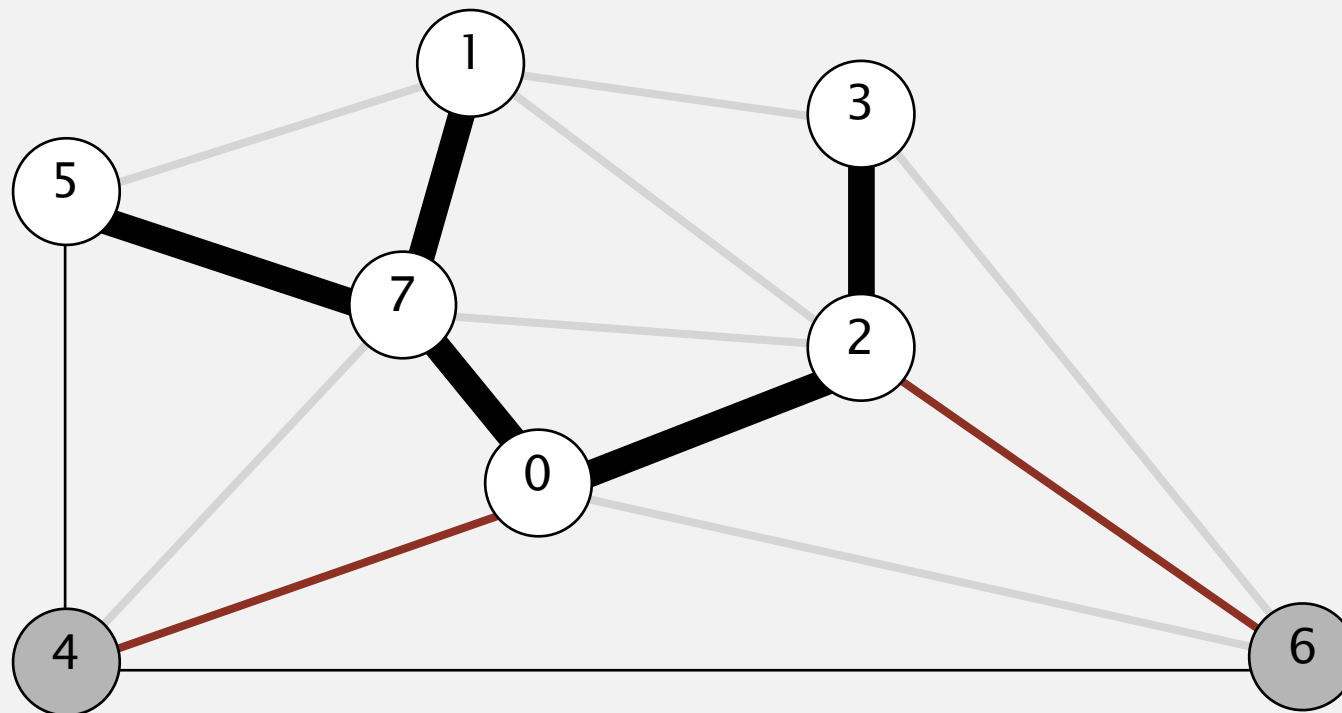| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 0–4 | 0.38 |
| 6 | 6–2 | 0.40 |

**MST edges**

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

114

- 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.

- 

| | |
|---|---|
| 0-7 | 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 |



| v | edgeTo[] | distTo[] |
|---|---|---|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 0–4 | 0.38 |
| 6 | 6–2 | 0.40 |

**MST edges**

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

# Prim's algorithm:  eager 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.
-

decrease key of 4
from 0.38 to 0.35



now a better connection
to 0 (discard)
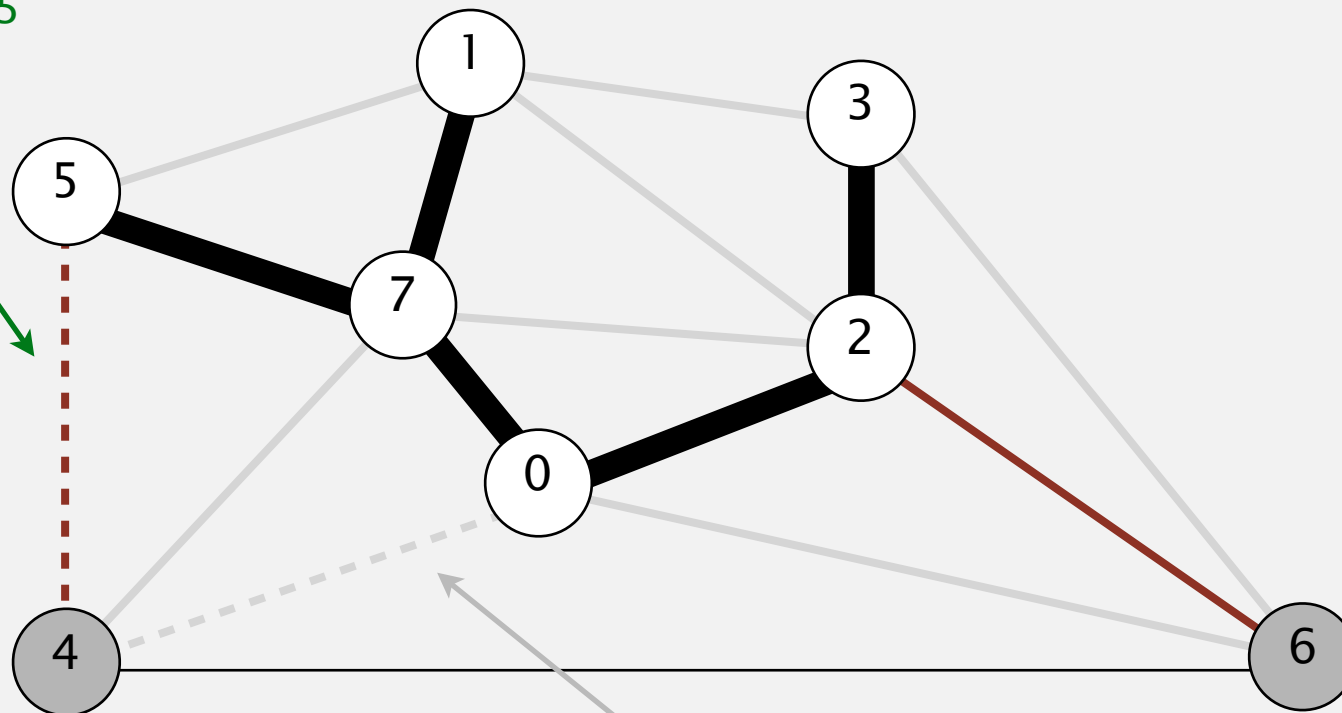
| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 0–4  4–5 | 0.38  0.35 |
| 6 | 6–2 | 0.40 |

**MST edges**

0–7   1–7   0–2   2–3   5–7

# Prim's algorithm:  eager 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.
- 



| v | edgeTo[] | distTo[] |
|---|---|---|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 4–5 | 0.35 |
| 6 | 6–2 | 0.40 |

**MST edges**

**0–7   1–7   0–2   2–3   5–7**

- 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.

- 



| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 4–5 | 0.35 |
| 6 | 6–2 | 0.40 |

**MST edges**

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

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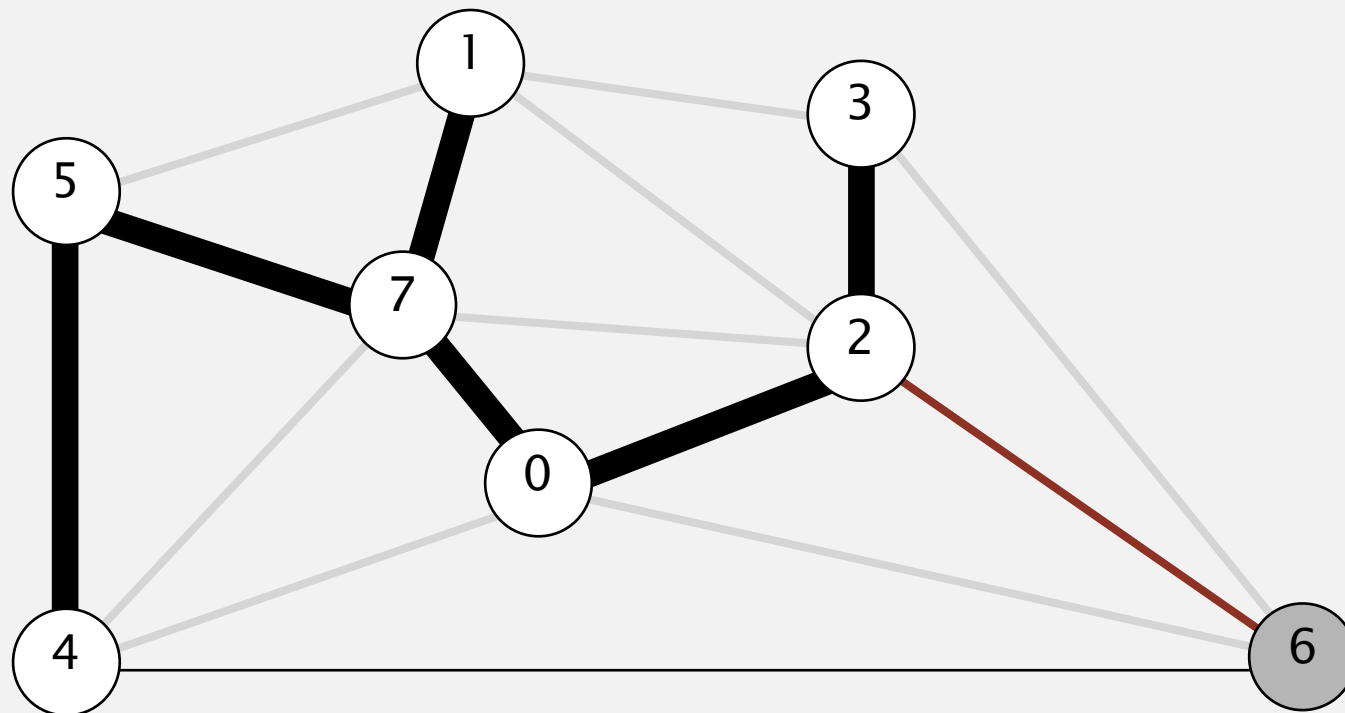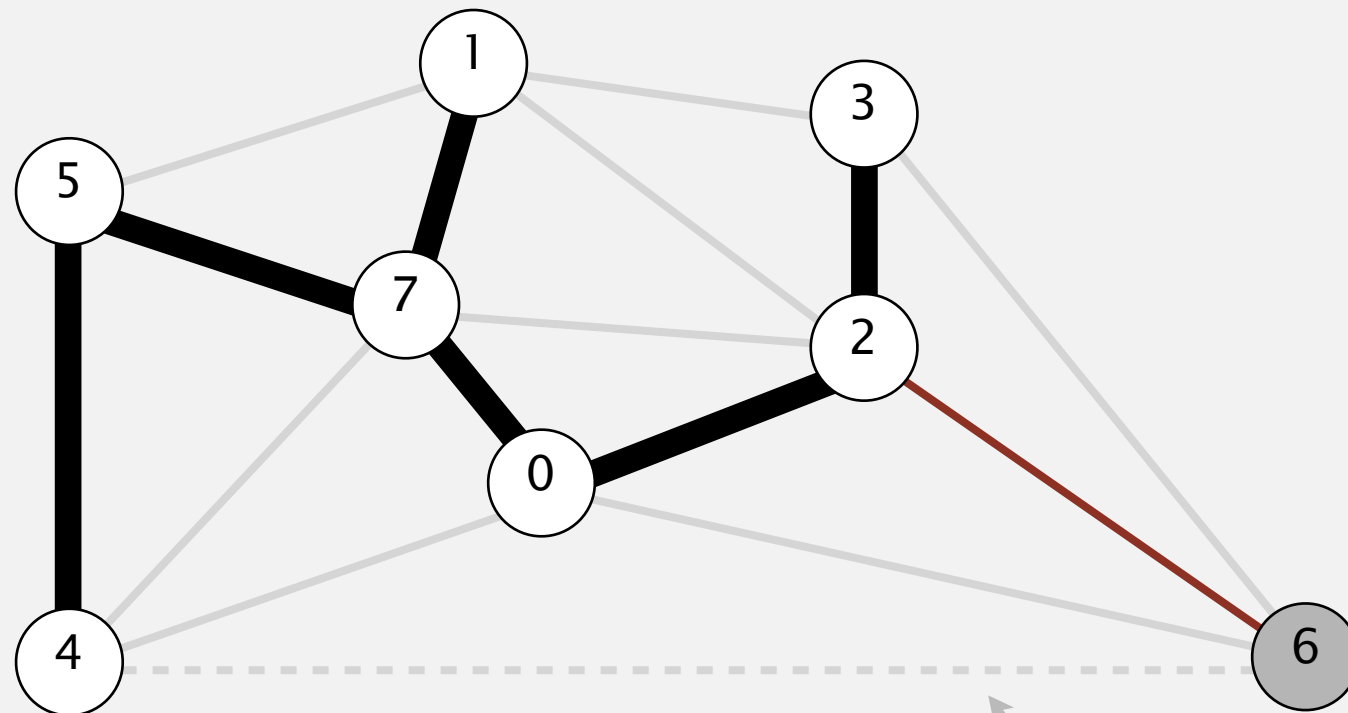- Repeat until $V - 1$ edges.
- 



| v | edgeTo[] | distTo[] |
|---|---|---|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 4–5 | 0.35 |
| 6 | 6–2 | 0.40 |

already a better connection
to 6 (discard)

**MST edges**

**0–7   1–7   0–2   2–3   5–7   4–5**

# Prim's algorithm:  eager 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.

-



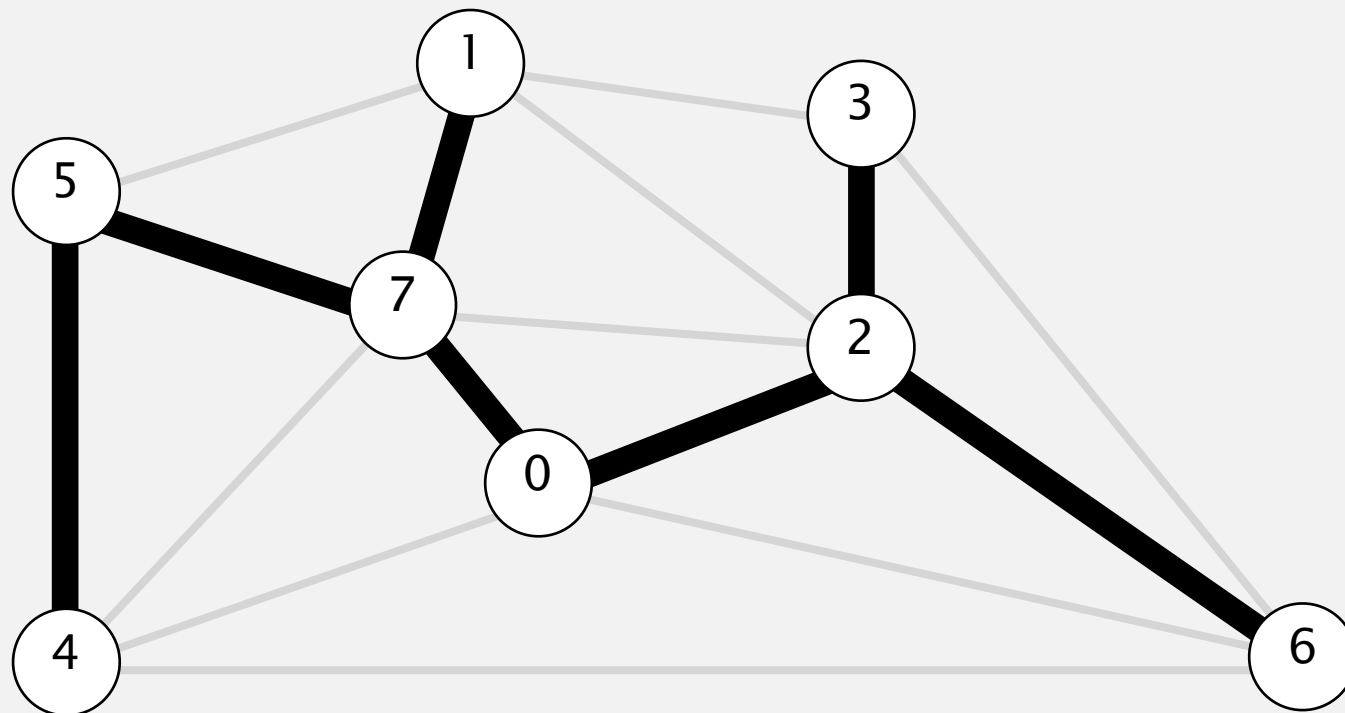| v | edgeTo[] | distTo[] |
|---|---|---|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 4–5 | 0.35 |
| 6 | 6–2 | 0.40 |

**MST edges**

0–7   1–7   0–2   2–3   5–7   4–5

# Prim's algorithm: eager 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.
-



| v | edgeTo[] | distTo[] |
|---|---|---|
| 0 | – | – |
| 7 | 0–7 | 0.16 |
| 1 | 1–7 | 0.19 |
| 2 | 0–2 | 0.26 |
| 3 | 2–3 | 0.17 |
| 5 | 5–7 | 0.28 |
| 4 | 4–5 | 0.35 |
| 6 | 6–2 | 0.40 |

**MST edges**

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

- 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.

- 



| v | edgeTo[] | distTo[] |
|---|----------|----------|
| 0 | –        | –        |
| 7 | 0–7      | 0.16     |
| 1 | 1–7      | 0.19     |
| 2 | 0–2      | 0.26     |
| 3 | 2–3      | 0.17     |
| 5 | 5–7      | 0.28     |
| 4 | 4–5      | 0.35     |
| 6 | 6–2      | 0.40     |

**MST edges**

**0–7   1–7   0–2   2–3   5–7   4–5   6–2**

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

↙ pq has at most one entry per vertex

Eager solution.  Maintain a PQ of vertices connected by an edge to $T$, where priority of vertex $v$ = weight of shortest edge connecting $v$ to $T$.
- Delete min vertex $v$ and add its associated edge $e = v–w$ to $T$.
- Update PQ by considering all edges $e = v–x$ incident to $v$
– ignore if $x$ is already in $T$
– add $x$ to PQ if not already on it
– decrease priority of $x$ if $v–x$ becomes shortest edge connecting $x$ to $T$
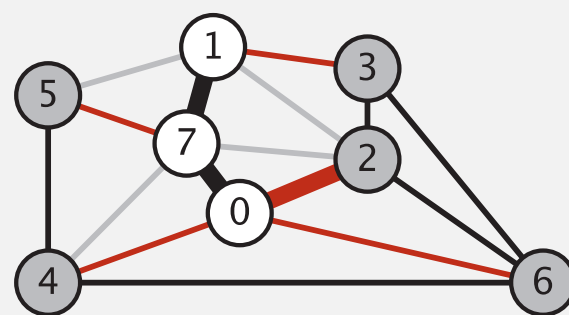
```
0
1  1-7  0.19
2  0-2  0.26       ←——— red:  on PQ
3  1-3  0.29
4  0-4  0.38
5  5-7  0.28
6  6-0  0.58
7  0-7  0.16
```

black:  on MST

# Prim's algorithm:  eager implementation

```java
public class PrimMST
{
   private Edge[] edgeTo;              // shortest edge from tree vertex
   private double[] distTo;           // distTo[w] = edgeTo[w].weight()
   private boolean[] marked;          // true if v on tree
   private IndexMinPQ<Double> pq;     // eligible crossing edges

   public PrimMST(EdgeWeightedGraph G)
   {
      edgeTo = new Edge[G.V()];
      distTo = new double[G.V()];
      marked = new boolean[G.V()];
      for (int v = 0; v < G.V(); v++)
         distTo[v] = Double.POSITIVE_INFINITY;
      pq = new IndexMinPQ<Double>(G.V());

      distTo[0] = 0.0;
      pq.insert(0, 0.0);                // Initialize pq with 0, weight 0.
      while (!pq.isEmpty())
         visit(G, pq.delMin());         // Add closest vertex to tree.
   }

   private void visit(EdgeWeightedGraph G, int v)
   {  // Add v to tree; update data structures.
      marked[v] = true;
      for (Edge e : G.adj(v))
      {
         int w = e.other(v);
         if (marked[w]) continue;       // v-w is ineligible.
         if (e.weight() < distTo[w])
         {  // Edge e is new best connection from tree to w.
            edgeTo[w] = e;
            distTo[w] = e.weight();
            if (pq.contains(w)) pq.change(w, distTo[w]);
            else                pq.insert(w, distTo[w]);
         }
      }
   }

   public Iterable<Edge> edges()      // See Exercise 4.3.21.
   public double weight()             // See Exercise 4.3.31.
}
```

# Eager Version's Prim's algorithm

The eager version of Prim's algorithm uses extra space proportional to *V* and time proportional to *E* log *V* (in the worst case) to compute the MST of a connected edge- weighted graph with *E* edges and *V* vertices.

| | | |
|---|---|---|
| *lazy Prim* | $E$ | $E \log E$ |
| *eager Prim* | $V$ | $E \log V$ |
| *Kruskal* | $E$ | $E \log E$ |