Minimum Spanning Tree: Undirected Graphs

Data Structures and Algorithms

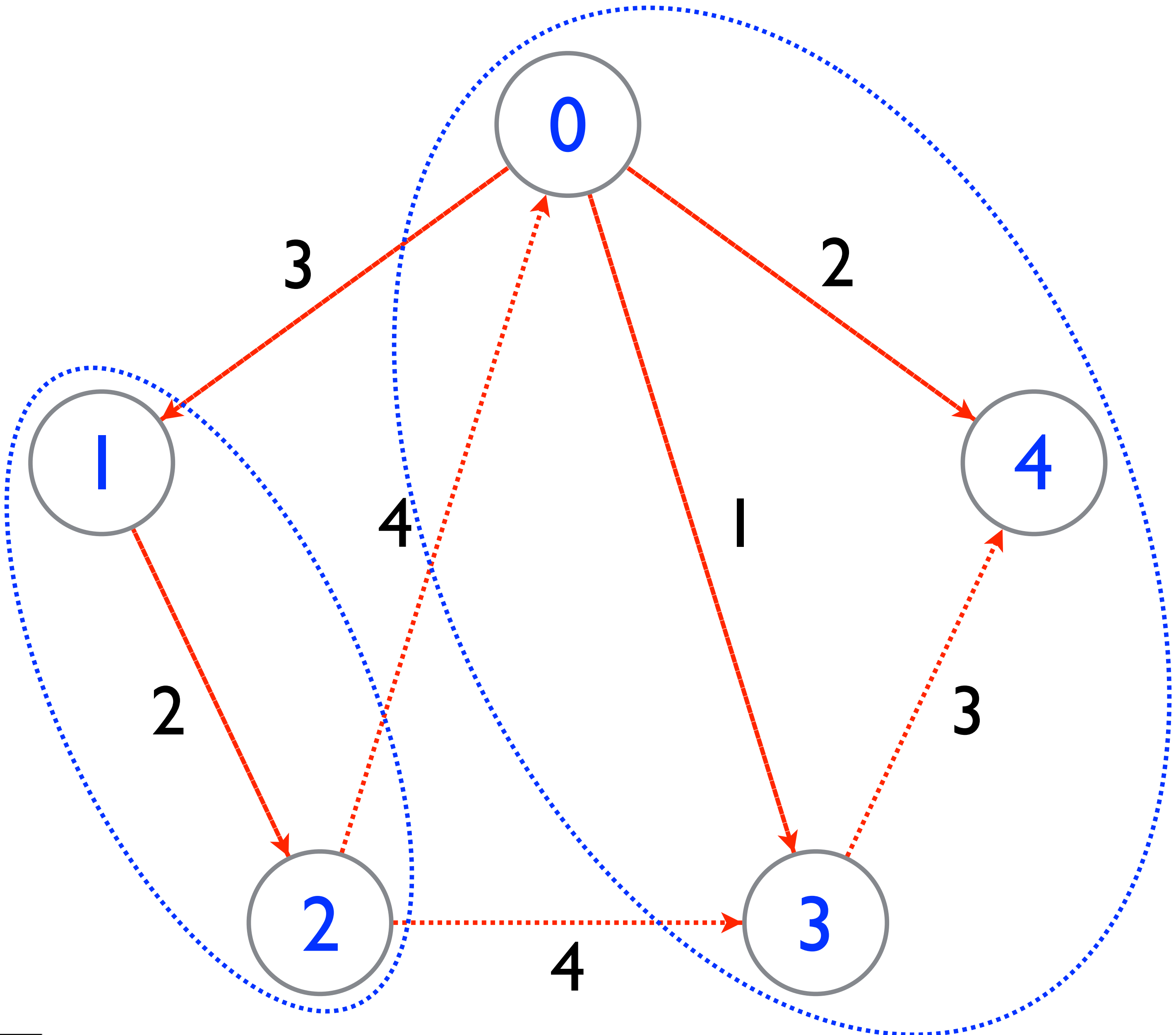
Emory University

Jinho D. Choi

EMORY UNIVERSITY

Types of Graphs

G = (V , E)



Undirected

Weighted

Directed

Acyclic

Tree

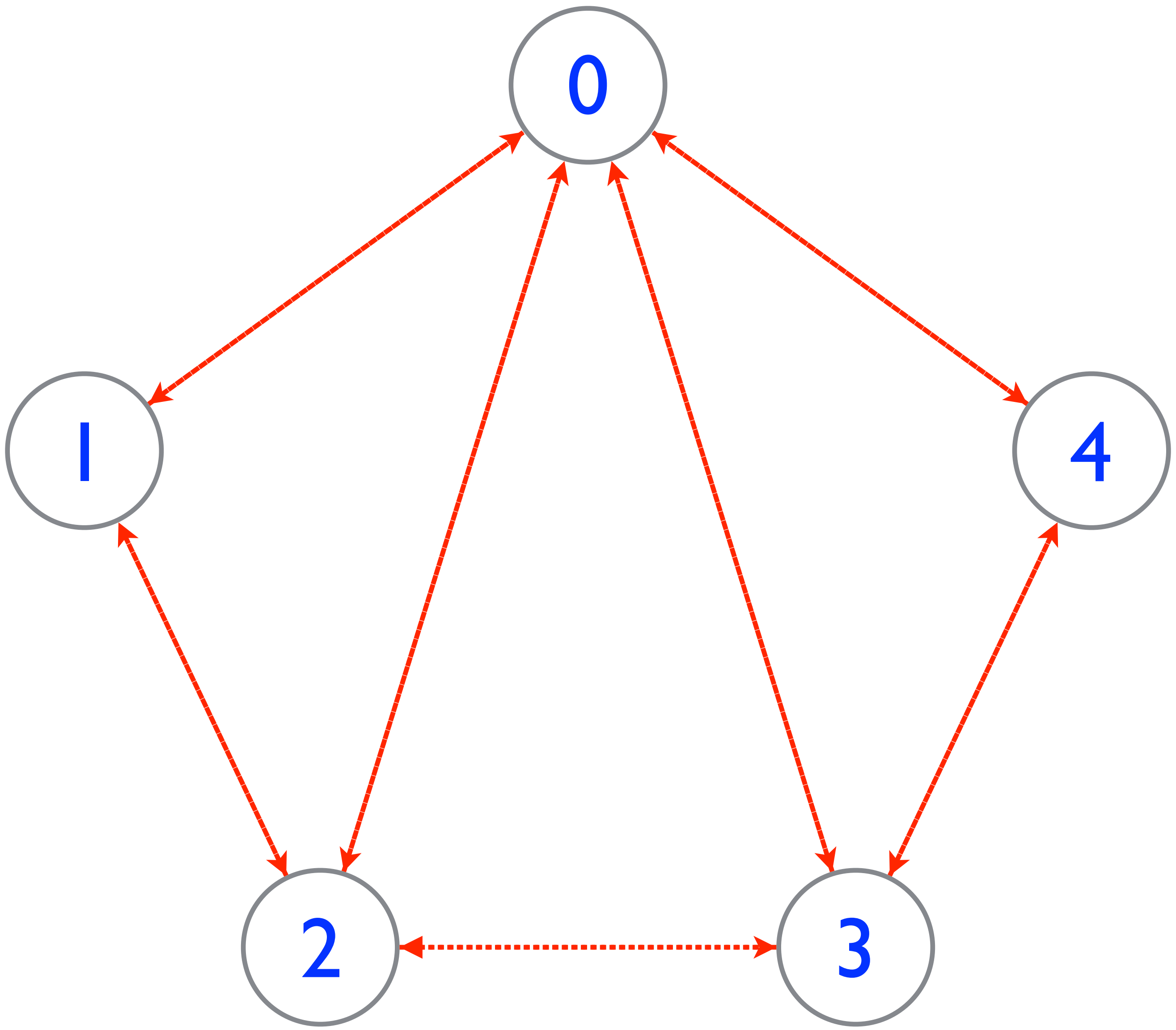
Forest

Every node except for the root must have exactly one incoming edge.

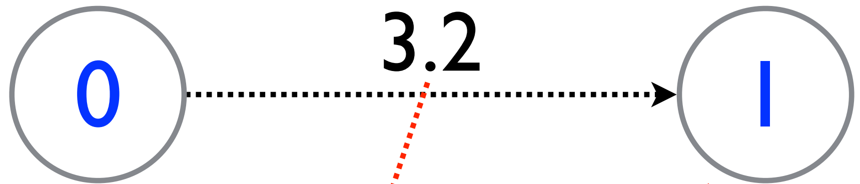
Every node must be reachable from the root.

Undirected vs. Directed Graph

Can we represent undirected graphs
using directed graphs?



Edge



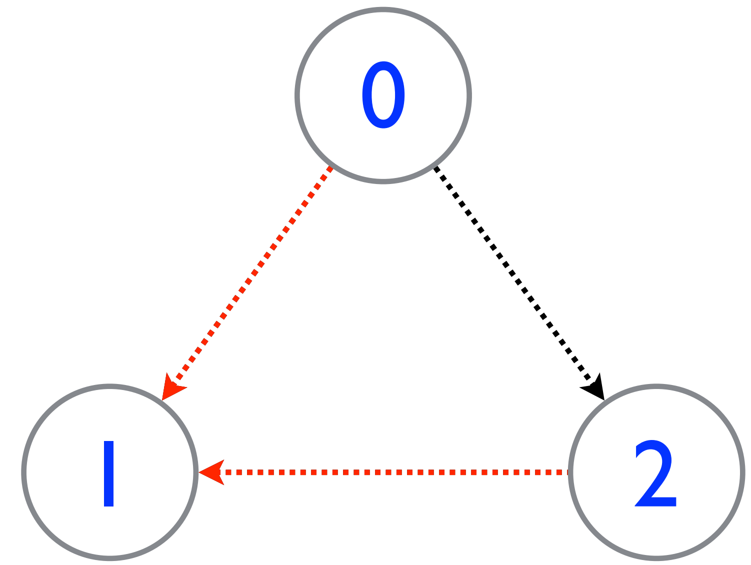

```
public class Edge implements Comparable<Edge>
{
  private int    source;
  private int    target;
  private double weight;

  public Edge(int source, int target, double weight)
  {
    setSource(source);
    setTarget(target);
    setWeight(weight);
  }
}
```

by weight

```
public int compareTo(Edge edge)
{
  double diff = weight - edge.weight;
  if      (diff > 0)  return  1;
  else if (diff < 0)  return -1;
  else                return  0;
}
```

Graph

```
incoming_edges[0] = {};
incoming_edges[1] = {1 <- 0, 1 <- 2};
incoming_edges[2] = {2 <- 0};
```




```
getIncomingEdges(1);
```

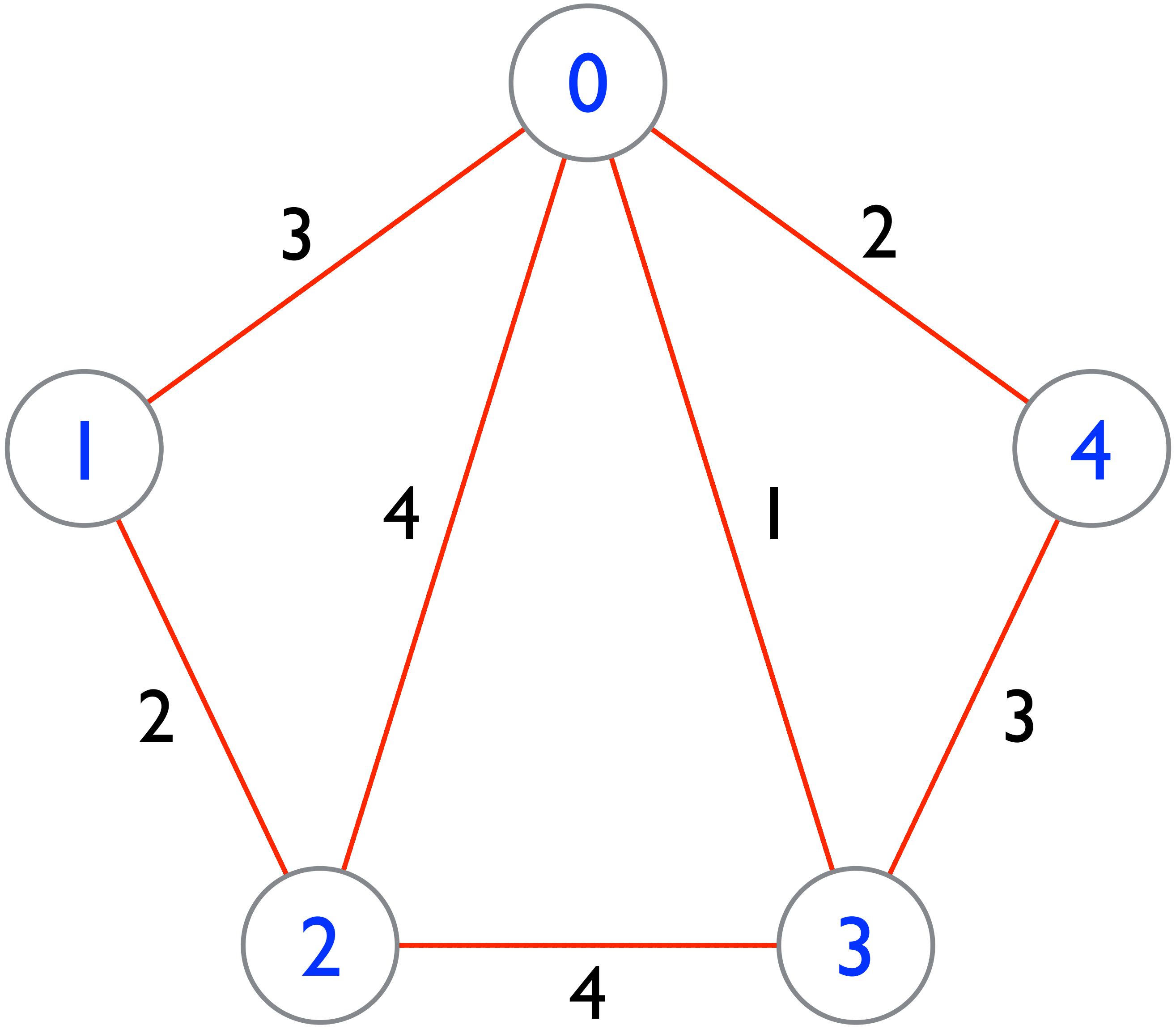
```
public class Graph
{
  private List<Edge>[] incoming_edges;

  public Graph(int size)
  {
    incoming_edges = (List<Edge>[])DSUtils.createEmptyListArray(size);
  }

  public List<Edge> getIncomingEdges(int target)
  {
    return incoming_edges[target];
  }
}
```

EMORY UNIVERSITY

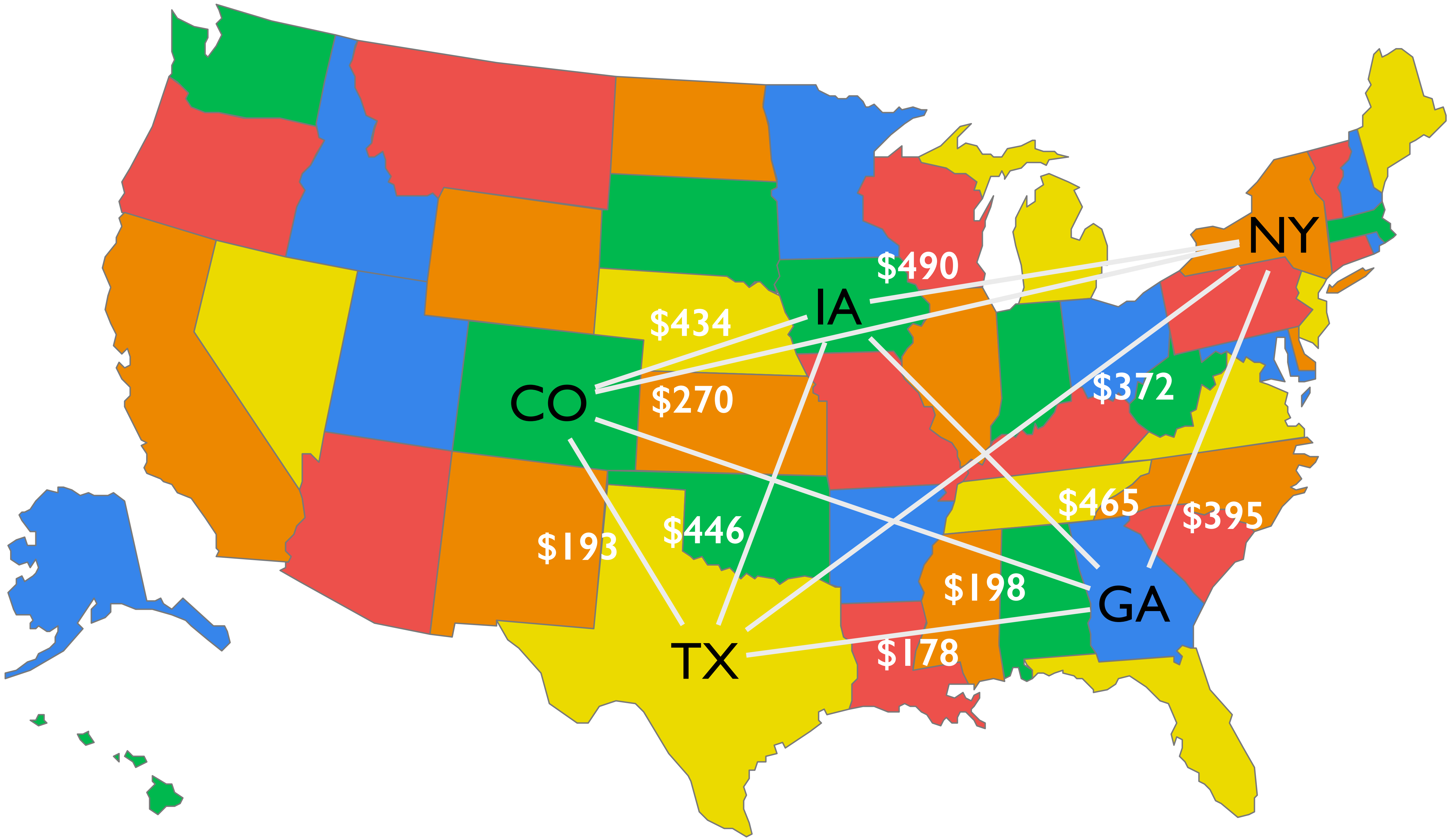
Graph


```
public void setDirectedEdge(int source, int target, double weight)
{
  List<Edge> edges = getIncomingEdges(target);
  edges.add(new Edge(source, target, weight));
}

public void setUndirectedEdge(int source, int target, double weight)
{
  setDirectedEdge(source, target, weight);
  setDirectedEdge(target, source, weight);
}
```

Minimum Spanning Tree



Minimum Spanning Tree

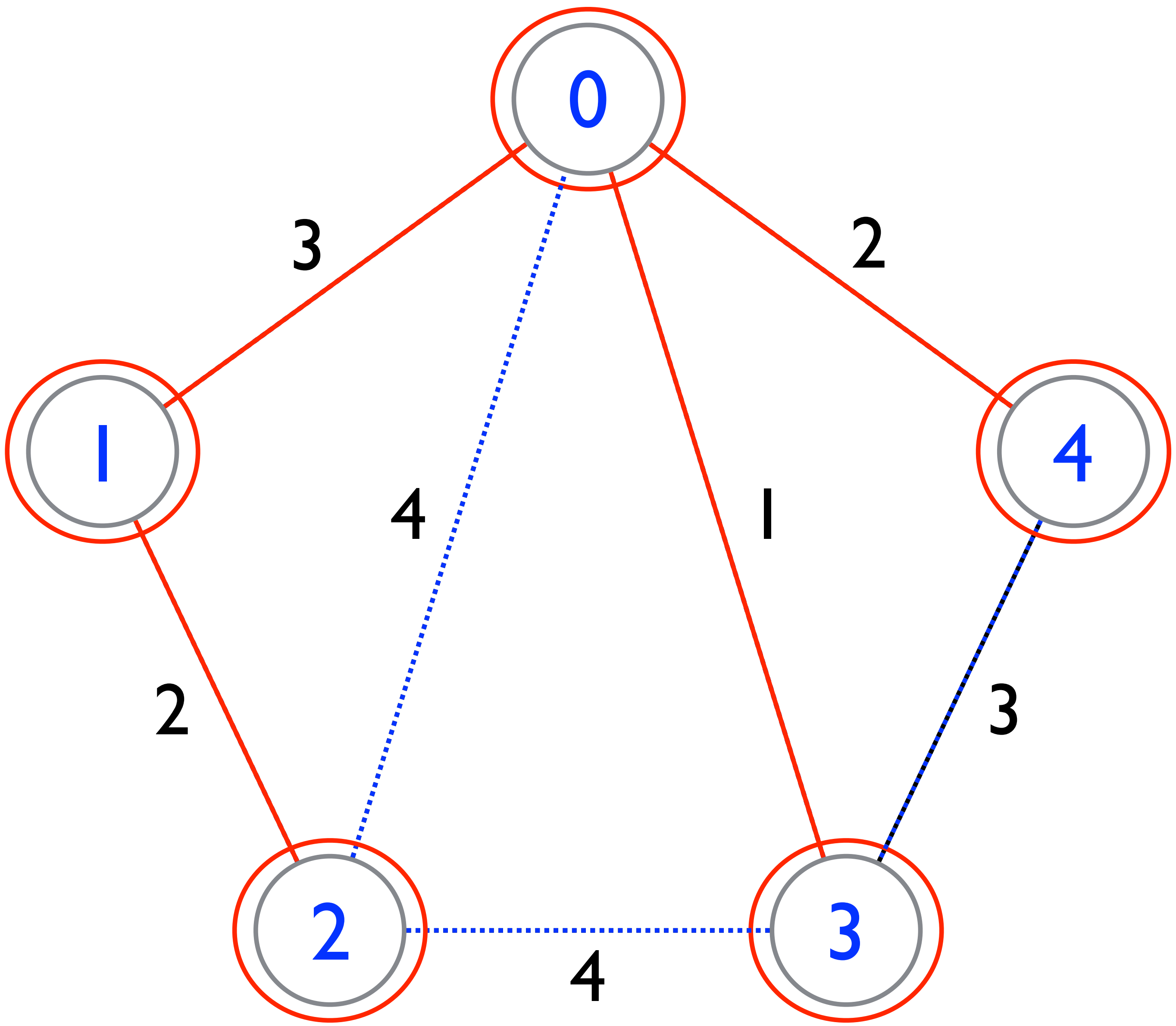


Spanning Tree

```
public class SpanningTree implements Comparable<SpanningTree>
{
  private List<Edge> edges;
  private double total_weight;

  public SpanningTree()
  {
    edges = new ArrayList<>();
  }

  public void addEdge(Edge edge)
  {
    edges.add(edge);
    total_weight += edge.getWeight();
  }
}
```

```
public int size()
{
  return edges.size();
}

public double getTotalWeight()
{
  return total_weight;
}

public int compareTo(SpanningTree tree)
{
  double diff = total_weight
              - tree.total_weight;
  if      (diff > 0) return 1;
  else if (diff < 0) return -1;
  else               return 0;
}
```

Prim's Algorithm




```
0 <- 3
0 <- 4
0 <- 1
1 <- 2
```

A spanning tree is found!

EMORY UNIVERSITY

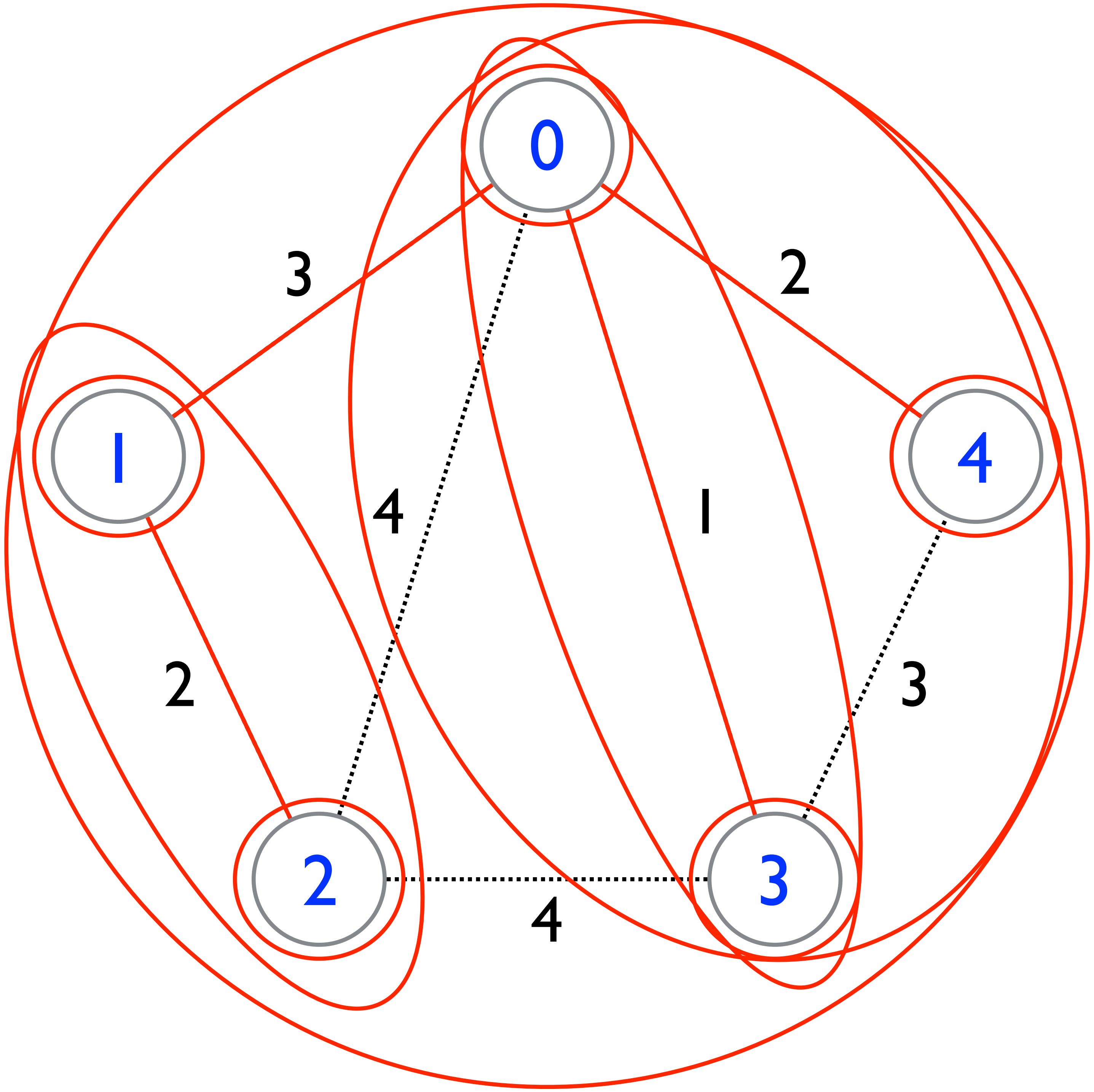
Prim's Algorithm

```java
public SpanningTree getMinimumSpanningTree(Graph graph)
{
  PriorityQueue<Edge> queue = new PriorityQueue<>();
  SpanningTree tree = new SpanningTree();
  Set<Integer> set = new HashSet<>();
  Edge edge;

  add(queue, set, graph, 0);

  while (!queue.isEmpty())
  {
    edge = queue.poll();

    if (!set.contains(edge.getSource()))
    {
      tree.addEdge(edge);
      if (tree.size()+1 == graph.size()) break;
      add(queue, set, graph, edge.getSource());
    }
  }

  return tree;
}
```

Complexity?

EMORY UNIVERSITY

Kruskal's Algorithm




```
0 <- 3
1 <- 2
0 <- 4
0 <- 1
```

Kruskal's Algorithm

```
public SpanningTree getMinimumSpanningTree(Graph graph)
{
  DisjointSet forest = new DisjointSet(graph.size());
  PriorityQueue<Edge> queue = createEdgePQ(graph);
  SpanningTree tree = new SpanningTree();
 Edge edge;

  while (!queue.isEmpty())
  {
    edge = queue.poll();

    if (!forest.inSameSet(edge.getTarget(), edge.getSource()))
    {
      tree.addEdge(edge);
      if (tree.size()+1 == graph.size()) break;
      forest.union(edge.getTarget(), edge.getSource());
    }
  }

  return tree;
}
```

```
Set<Integer>[] forest = DSUtils.createEmptySetArray(size);
for (int i=0; i<size; i++) forest[i].add(i);
return forest;
```

Proof by Induction

$$\sum_{n=0}^{k} = \frac{k(k+1)}{2}$$

Proof by Induction

Base

$$\sum_{n=0}^{0} = \frac{0 \cdot 1}{2} = 0$$

Induction

Assume $\displaystyle\sum_{n=0}^{k} = \frac{k(k+1)}{2}$ Then,

$$\sum_{n=0}^{k} + (k+1) = \frac{k(k+1)}{2} + (k+1) = \frac{K^2 + 3k + 2}{2} = \frac{(k+1)(k+2)}{2}$$

EMORY UNIVERSITY

Proof by Contradiction

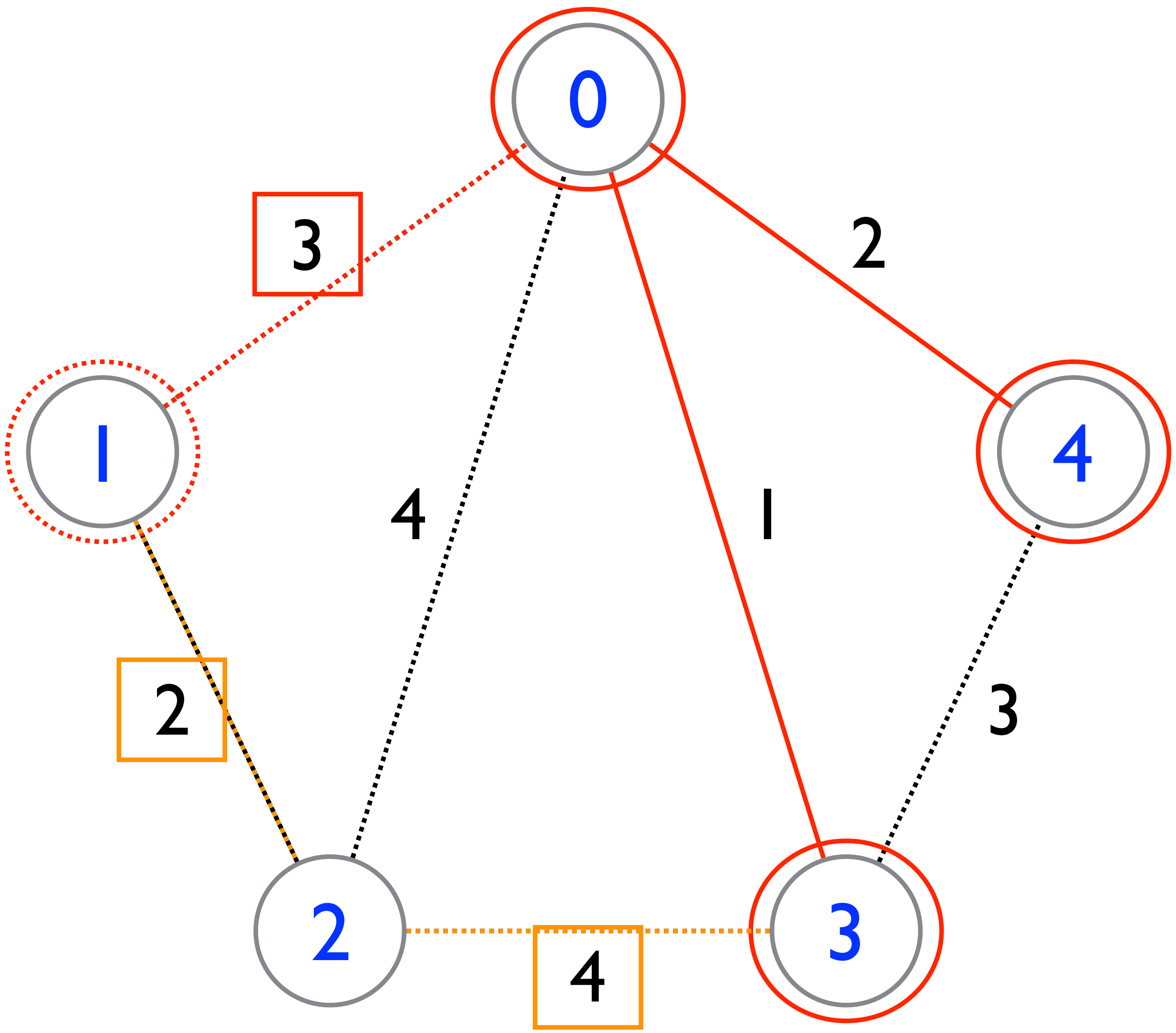
$$n^2 = n \cdot n = 2k \cdot 2k = \boxed{2(2 \cdot 2k^2)}$$

Contradiction

EMORY UNIVERSITY

Correctness of Prim's Algorithm

Correctness of Prim's Algorithm



Prim's algorithm finds e.

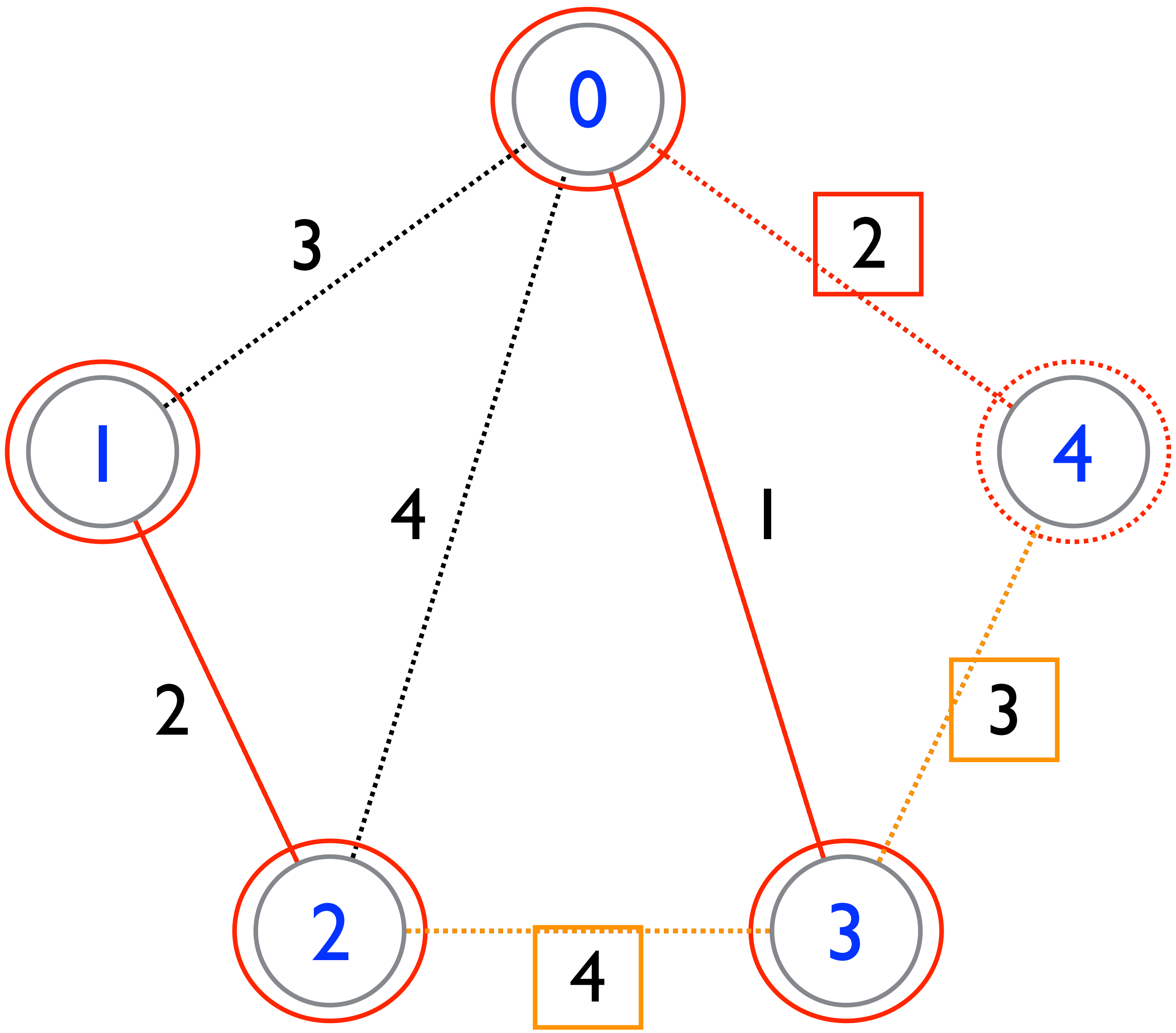
Suppose that there exists another path p from T to 1 whose total weight < e.

Then, all edges in p must have a weight < e.

Proof by contradiction!

Correctness of Kruskal's Algorithm

Correctness of Kruskal's Algorithm



Kruskal's algorithm finds e.

Suppose that there exists another path p from T_s to T_t whose total weight < e.

Then, all edges in p must have a weight < e.

Proof by contradiction!