Lecture 11

Instructor: Karteek Sreenivasaiah

25th September 2018

Disjoint Set

Maintain a collection $\mathcal{F} = \{S_1, S_2, \ldots, S_k\}$ of disjoint sets.
One element from each set serves as a 'representative' for that set.

Disjoint Set supports the following procedures:

- $\textsc{MakeSet}(x)$ – Creates a singleton set with element x.
- $\textsc{Union}(x, y)$ – Performs union on sets containing x and y.
- $\textsc{FindSet}(x)$ – Find the set containing x.

MakeSet(x)

Creates a singleton set containing x.

We assume that x is not an element of any other set in $\mathcal{F}$.
We assign x as the representative for the set just created.

Union

Union(x, y)

Performs union on sets containing x and y.

Let $S, T \in \mathcal{F}$ such that $x \in S$ and $y \in T$.
Create a new set $U = S \cup T$.
Choose and assign a representative for U.
Remove S and T from $\mathcal{F}$.

FindSet

FindSet(x)
Find the set containing x.

Let $S \in \mathcal{F}$ such that $x \in S$. (Note: exactly one set contains x.)
Return a pointer to the representative element of S.

Implementation

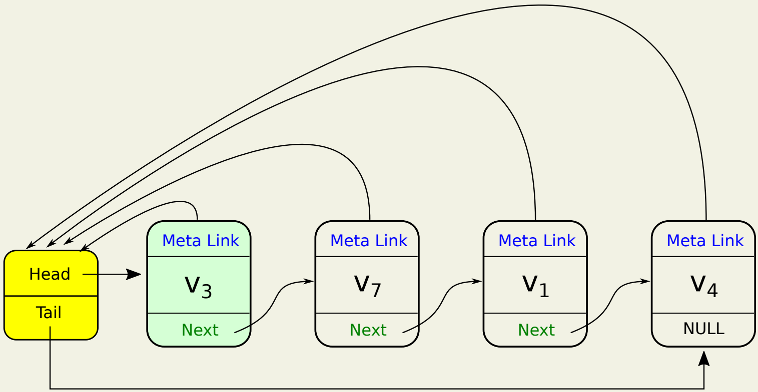
Disjoint Set using linked lists:

- For each set S, maintain:
  - a node with metadata
  - a linked list L_S with the objects in the set.
- The "Metadata Node" stores head and tail pointers to the linked list.
- Each node in the linked list consists of:
  - The value of the element.
  - A pointer to the next element.
  - A pointer to the Metadata Node.

The head of L_S is the representative of S.

Implementation

Linked list for set $\{v_1, v_3, v_4, v_7\}$.



Implementation

$\textsc{MakeSet}(x)$

Creates a singleton set with element x

- Create a new node for metadata
- Create a linked list containing just x.
- Node x is the head and tail of the list.
- Representative for this set is x itself.

Implementation

FindSet(x)

Find the set containing node x.

- Return a pointer to the representative.

Implementation

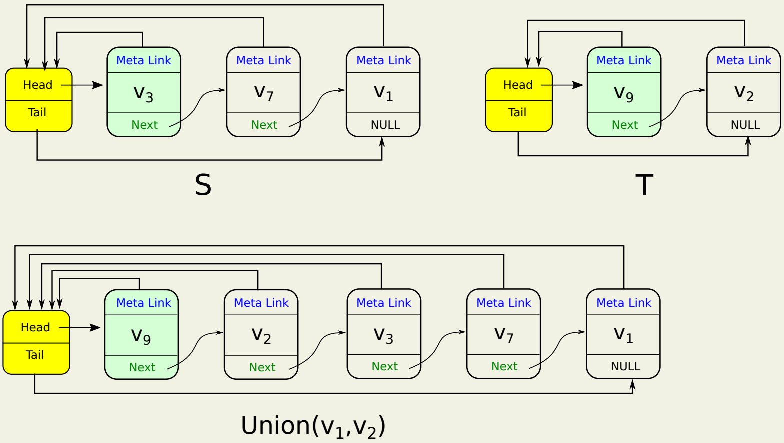
UNION(X,Y)

Union of sets containing x and y.

- Append linked list of set S containing x to set T containing y.
- Representative of new set is same as representative of T.
- Update meta pointers of nodes in S to the correct metadata node.
- Update tail pointer in metadata node of T.

Implementation

Union of sets $S = \{v_1, v_3, v_7\}$ and $T = \{v_2, v_9\}$.



Analysis

Running time under Linked List implementation:

- MakeSet(x) – $O(1)$
- FindSet(x) – $O(1)$
- Union(x, y) – ?

Analysis

Union(x, y) –

- $S \leftarrow$ FindSet(x) and $T \leftarrow$ FindSet(y) – $O(1)$ time.
- Appending linked list of S to tail end of T – $O(1)$ time.
- Updating the new metadata (tail) – $O(1)$ time.
- Updating the backward pointers of nodes in S takes $O(n)$ time.

We can show a case where after $O(n)$ operations, time taken would be $O(n^2)$.

Spanning Tree

A graph $T = (V, E')$ is a spanning tree of an undirected connected graph $G = (V, E)$ if:

- $E' \subseteq E$.
- T is a *tree*. i.e., there are no cycles in T.
- T is connected.

Informally: A spanning tree for G is a tree that can be found inside G which *spans* all vertices of G.

Recap: Minimum Spanning Tree Problem

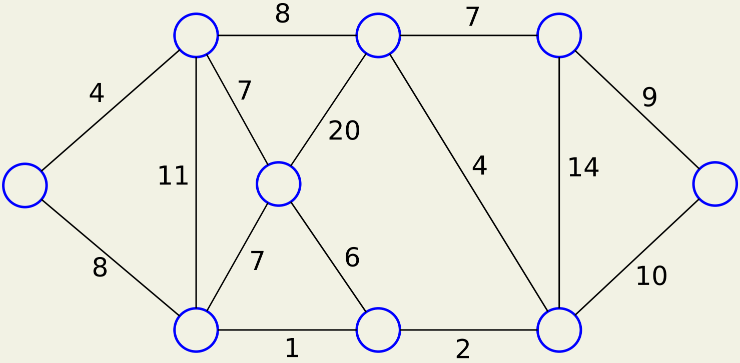
Input

- Undirected connected graph $G = (V, E)$
- Weight function $w : E \to \mathbb{Z}^+$

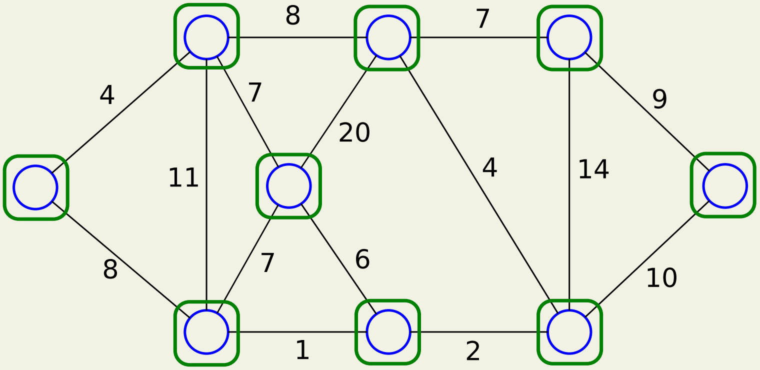
Goal

Compute a spanning tree for G with minimum total weight.

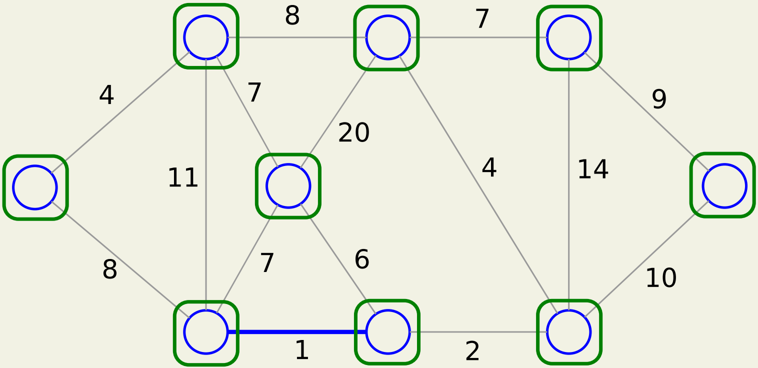
Kruskal’s algorthm example



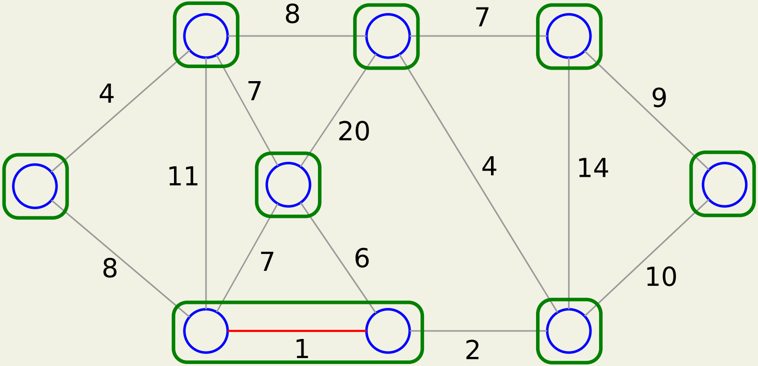
Kruskal's algorthm example



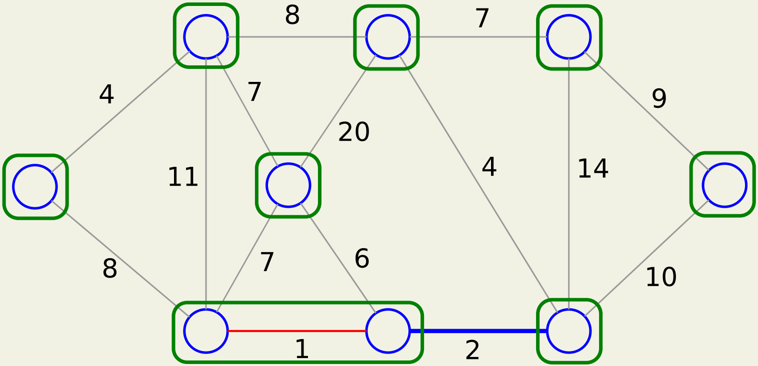
Kruskal's algorthm example



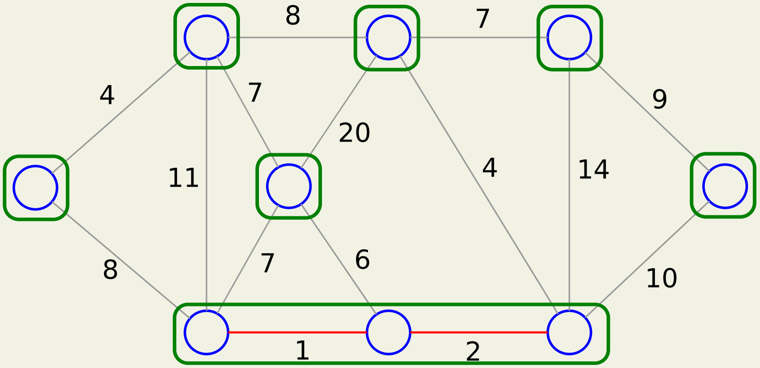
Kruskal’s algorthm example



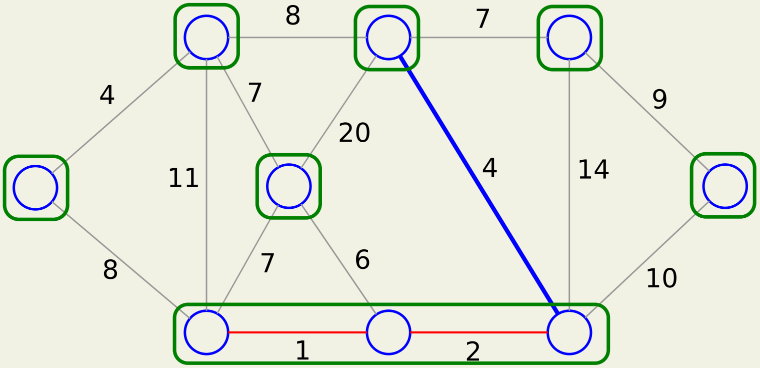
Kruskal’s algorthm example



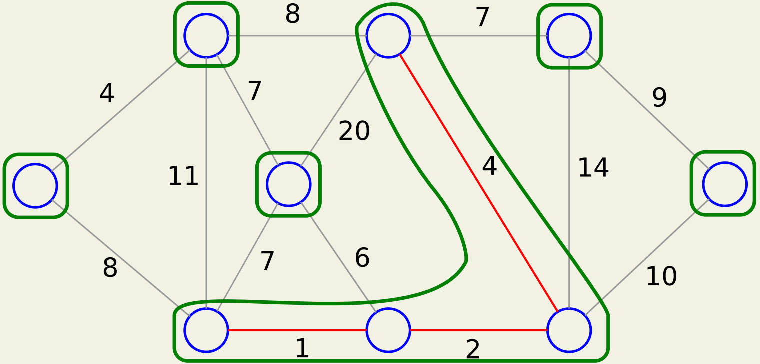
Kruskal’s algorthm example



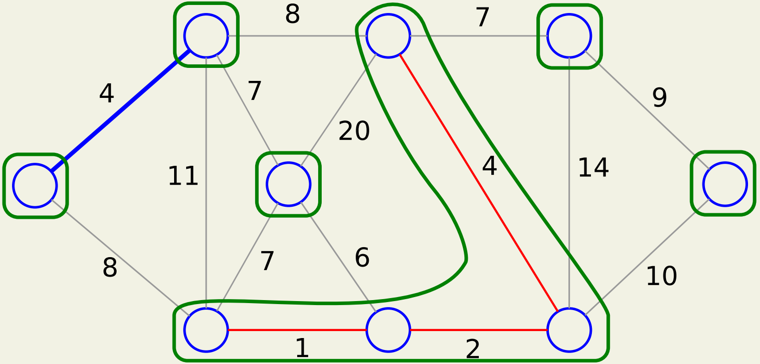
Kruskal’s algorthm example



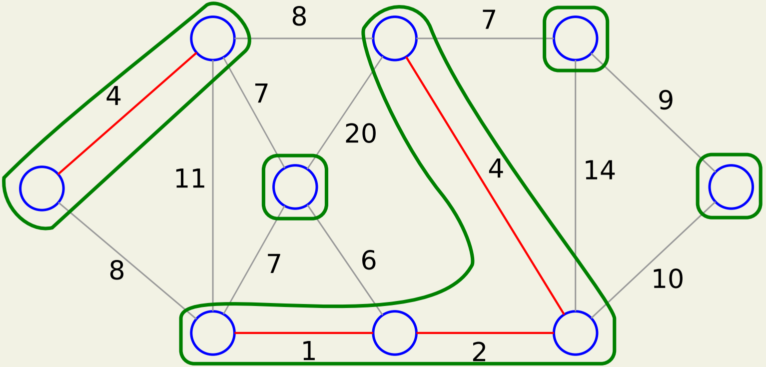
Kruskal’s algorthm example



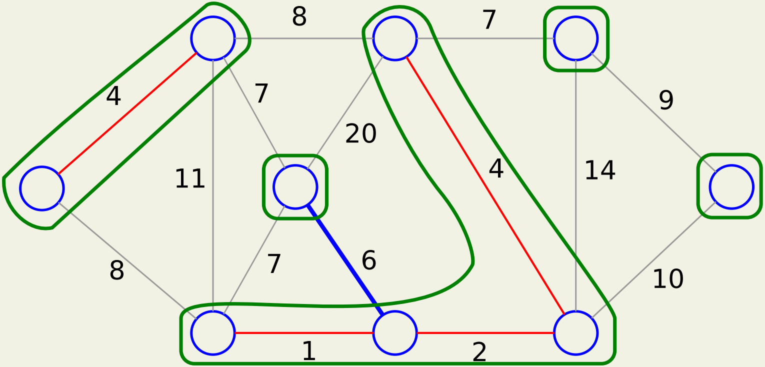
Kruskal's algorthm example



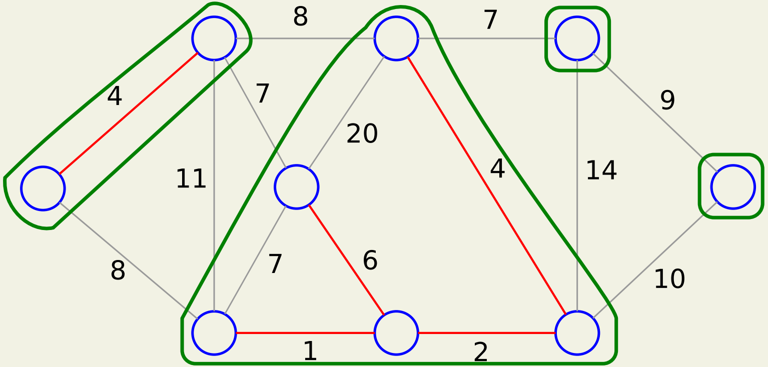
Kruskal’s algorthm example



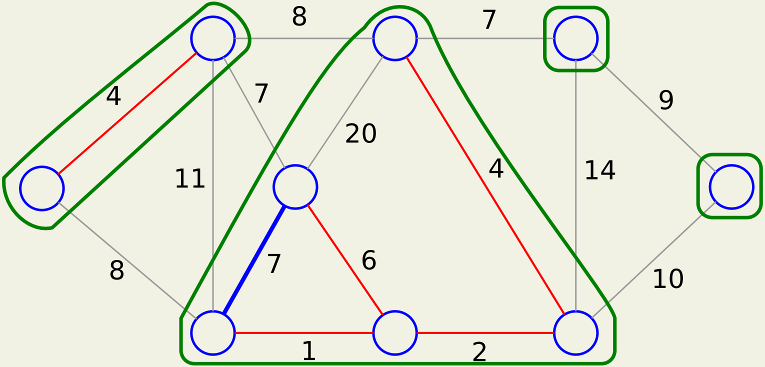
Kruskal's algorthm example



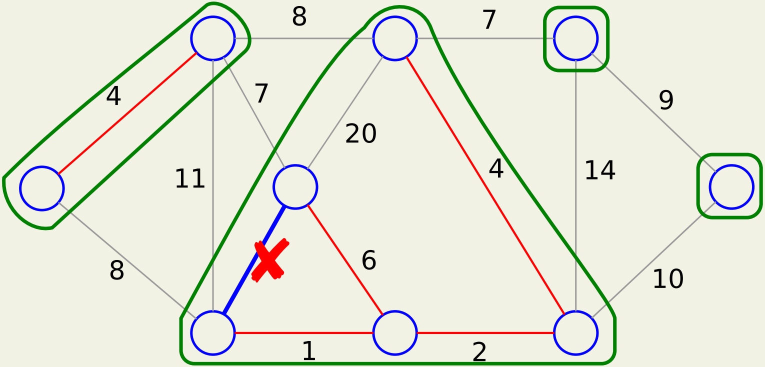
Kruskal's algorthm example



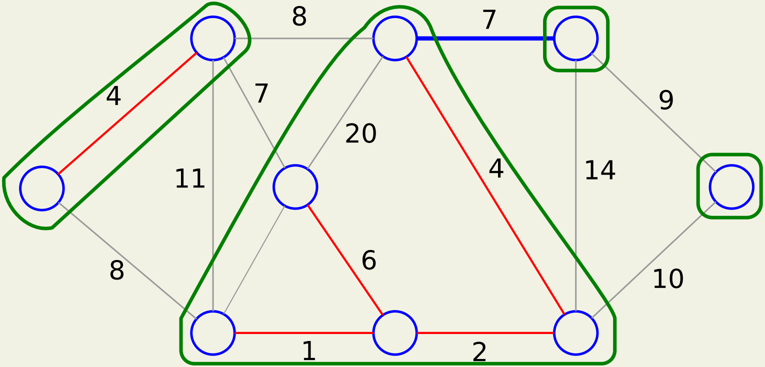
Kruskal’s algorthm example



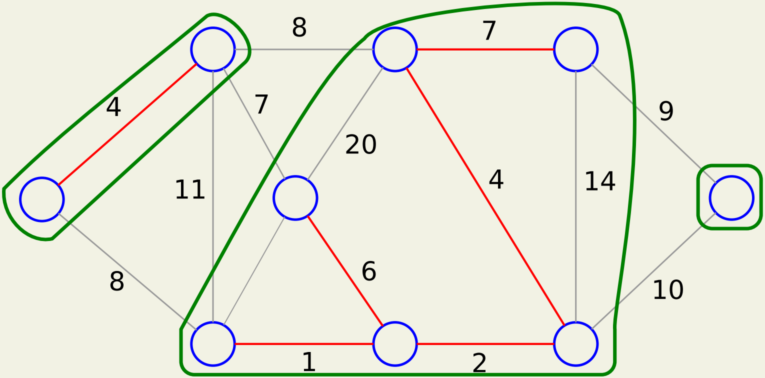
Kruskal’s algorthm example



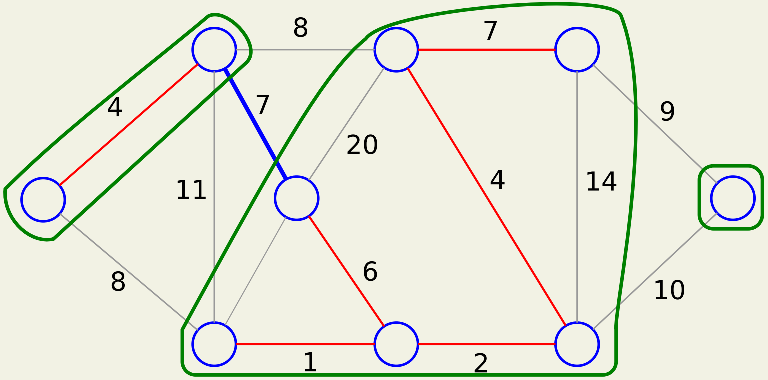
Kruskal's algorthm example



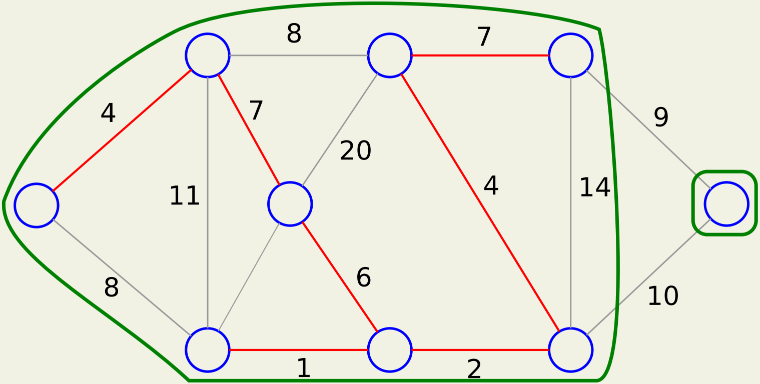
Kruskal's algorthm example



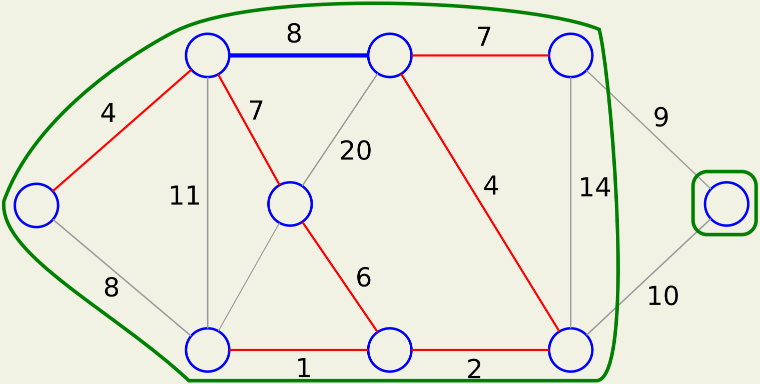
Kruskal’s algorthm example



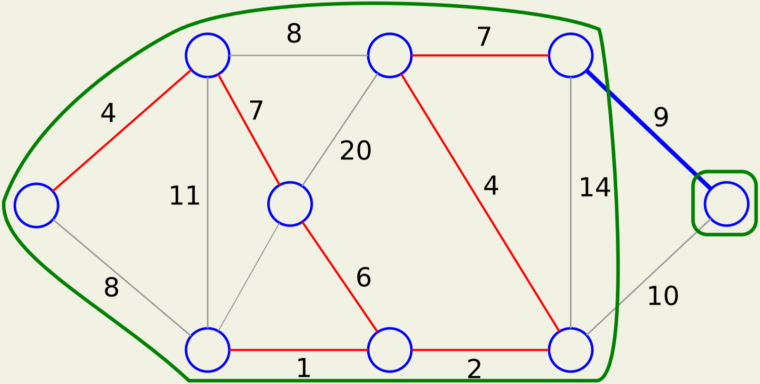
Kruskal's algorthm example



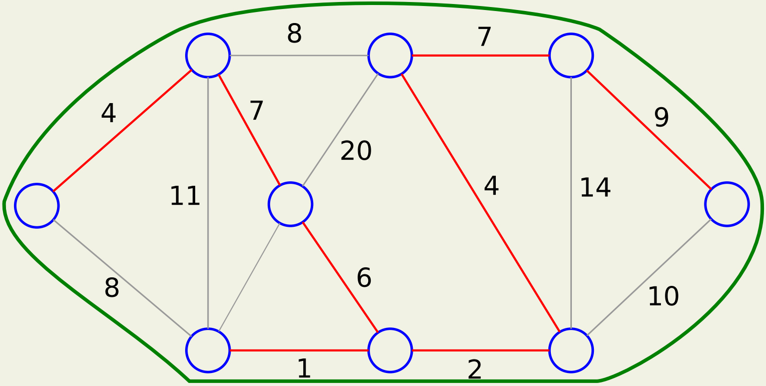
Kruskal’s algorthm example



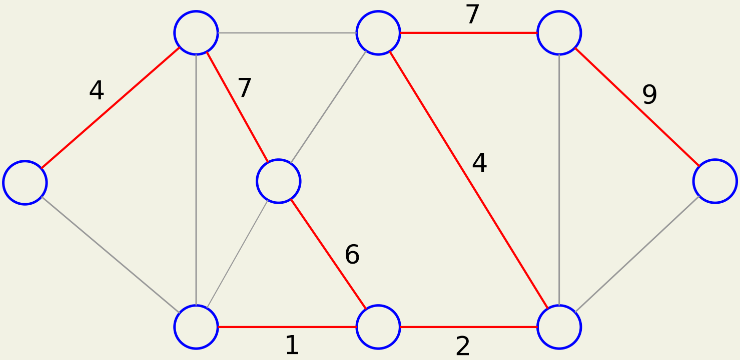
Kruskal's algorthm example



Kruskal's algorthm example



Kruskal’s algorthm example



Pseudocode

Algorithm 1 Input: Undirected connected graph $G = (V, E)$

```
T ← ∅
for each v ∈ V do
    Create a node for v
    MakeSet(v)
end for
while T ≠ V do
    Choose smallest weight edge e = {u, v} ∈ E
    if FindSet(u) ≠ FindSet(v) then
        T ← T ∪ {e}
        Union(u, v)
    end if
end while
```