# Lecture 8: Kruskal's MST Algorithm

CLRS Chapter 23

**Main Topics of This Lecture**

- Kruskal's algorithm
  Another, but different, greedy MST algorithm

- Introduction to UNION-FIND data structure.
  Used in Kruskal's algorithm
  Will see implementation in next lecture.

## Idea of Kruskal's Algorithm

The Kruskal's Algorithm is based directly on the generic algorithm. Unlike Prim's algorithm, we make a different choices of cuts.

Initially, trees of the forest are the vertices (no edges).

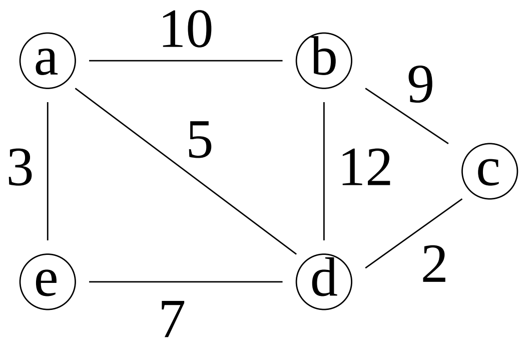
In each step add the cheapest edge that does not create a cycle.

Observe that unlike Prim's algorithm, which only grows one tree, Kruskal's algorithm grows a collection of trees (a forest).

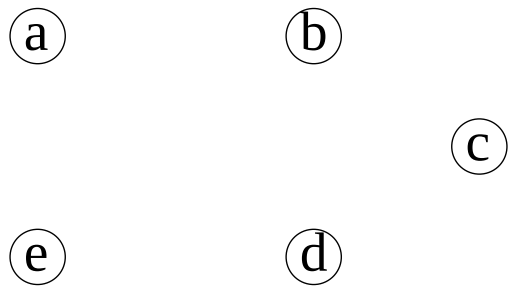
Continue until the forest 'merge to' a single tree.
(Why is a single tree created?)

This is a *minimum* spanning tree
(we must prove this).

# Outline by Example




original graph




forest $\longrightarrow$ MST

E

| edge | weight |
|---|---|
| {d, c} | 2 |
| {a, e} | 3 |
| {a, d} | 5 |
| {e, d} | 7 |
| {b, c} | 9 |
| {a, b} | 10 |
| {b, d} | 12 |
| | |

Forest (V, A)

A={ }

## Outline of Kruskal's Algorithm

**Step 0:** Set $A = \emptyset$ and $F = E$, the set of all edges.

**Step 1:** Choose an edge $e$ in $F$ of minimum weight, and check whether adding $e$ to $A$ creates a cycle.

- If "yes", remove $e$ from $F$.
- If "no", move $e$ from $F$ to $A$.

**Step 2:** If $F = \emptyset$, stop and output the minimal spanning tree $(V, A)$. Otherwise go to Step 1.

**Remark:** Will see later, after each step, $(V, A)$ is a subgraph of a MST.

## Outline of Kruskal's Algorithm

**Implementation Questions:**

- How does algorithm choose edge $e \in F$ with minimum weight?

- How does algorithm check whether adding $e$ to $A$ creates a cycle?

## How to Choose the Edge of Least Weight

**Question:**
How does algorithm choose edge $e \in F$ with minimum weight?

**Answer:** Start by sorting edges in $E$ in order of increasing weight.
Walk through the edges in this order.
(Once edge $e$ causes a cycle it will always cause a cycle so it can be thrown away.)

## How to Check for Cycles

**Observation:** At each step of the outlined algorithm, $(V, A)$ is acyclic so it is a forest.

If $u$ and $v$ are in the same tree, then adding edge $\{u, v\}$ to $A$ creates a cycle.

If $u$ and $v$ are not in the same tree, then adding edge $\{u, v\}$ to $A$ does not create a cycle.

**Question:** How to test whether $u$ and $v$ are in the same tree?

**High-Level Answer:** Use a disjoint-set data structure
Vertices in a tree are considered to be in same set.
Test if Find-Set($u$) = Find-Set($v$)?

**Low -Level Answer:**
The UNION-FIND data structure implements this:

# The UNION-FIND Data Structure

UNION-FIND supports three operations on collections of **disjoint sets**: Let $n$ be the size of the universe.

**Create-Set($u$):** $O(1)$

Create a set containing the single element $u$.

**Find-Set($u$):** $O(\log n)$

Find the set containing the element $u$.

**Union($u, v$):** $O(\log n)$

Merge the sets respectively containing $u$ and $v$ into a common set.

For now we treat UNION-FIND as a black box.
Will see implementation in next lecture.

## Kruskal's Algorithm: the Details

Sort $E$ in increasing order by weight $w$; $O(|E| \log |E|)$

/* After sorting $E = \langle \{u_1, v_1\}, \{u_2, v_2\}, \ldots, \{u_{|E|}, v_{|E|}\} \rangle$ */

$A = \{\}$;
for (each $u$ in $V$) CREATE-SET($u$); $O(|V|)$

for $e_i = (u_i, v_i)$ from 1 to $|E|$ do $O(|E| \log |V|)$
    if (FIND-SET($u_i$) != FIND-SET($v_i$) )
    { add $\{u_i, v_i\}$ to $A$;
        UNION($u_i, v_i$);
    }
return(A);

**Remark:** With a proper implementation of UNION-FIND, Kruskal's algorithm has running time $O(|E| \log |E|)$.

## Why Kruskal's Algorithm is correct?

Let $A$ be the edge set which has been selected by Kruskal's Algorithm, and $(u, v)$ be the edge to be added next. It suffices to show there is a cut which respects $A$, and $(u, v)$ is the light edge crossing that cut.

1. Let $A^{'} = (V^{'}, E^{'})$ denote the tree of the forest $A$ that contains $u$. Consider the cut $(V^{'}, V - V^{'})$.

2. Observe that there is no edge in $A$ crosses this cut, so the cut respects $A$.

3. Since adding $(u, v)$ to $A^{'}$ does not induce a cycle, $(u, v)$ crosses the cut. Moreover, since $(u, v)$ is currently the smallest edge, $(u, v)$ is the light edge crossing the cut. This completes the correctness proof of Kruskal's Algorithm.

# Why Kruskal's Algorithm is correct?