

CSE 100

Disjoint Set, Union Find

Naïve Implementation of Kruskal's algorithm:

1. Sort edges in increasing order of cost
2. Set of edges in MST, $T = \{\}$
3. For $i = 1$ to $|E|$

If $T \cup \{e_i\}$ has no cycles

Add e_i to T

Towards a fast implementation of Kruskal's Algorithm

Q: Which of the following algorithms can be used to check if adding an edge (v, w) to an existing Graph creates a cycle?

- A. DFS
- B. BFS
- C. Either A or B
- D. None of the above

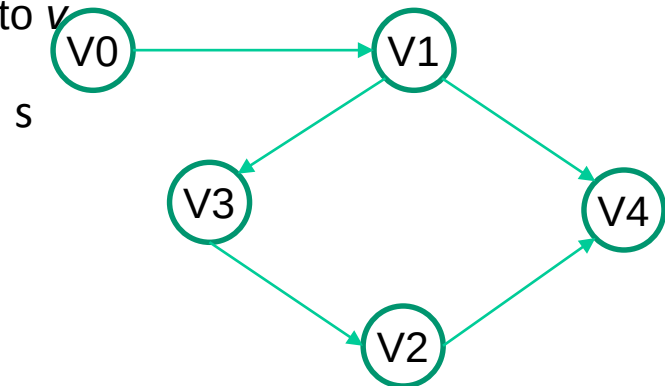
BFS: Running Time

The basic idea is a breadth-first search of the graph, starting at source vertex s

- Initially, give all vertices in the graph a distance of INFINITY
- Start at s ; give s distance = 0
- Enqueue s into a queue
- While the queue is not empty:
 - Dequeue the vertex v from the head of the queue
 - For each of v 's adjacent nodes that has not yet been visited:
 - Mark its distance as 1 + the distance to v

What is the time complexity (in terms of $|V|$ and $|E|$) of this algorithm?

- A. $O(|V|)$
- B. $O(|V| |E|)$
- C. $O(|V| + |E|)$
- D. $O(|V|^2)$
- E. Other





Running Time of Naïve Implementation of Kruskal's algorithm using BFS for cycle checks:

Running Time

1. Sort edges in increasing order of cost
2. Set of edges in MST, $T = \{\}$
3. For $i = 1$ to $|E|$

If $T \cup \{e_i\}$ has no cycles

Add e_i to T

Towards a fast implementation for Kruskal's Algorithm

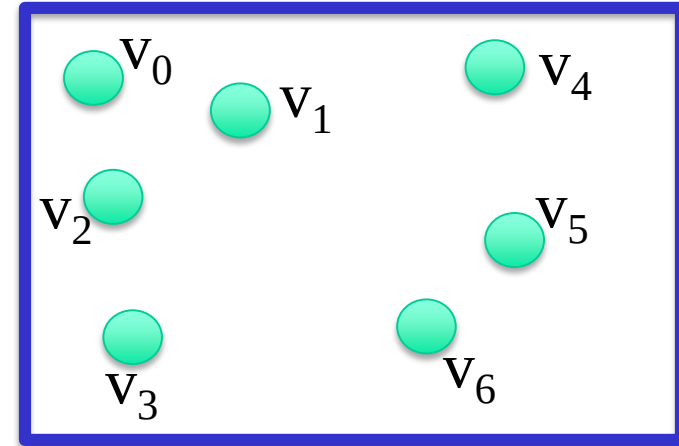
- What is the work that we are repeatedly doing with Kruskal's Algo?

Towards a fast implementation for Kruskal's Algorithm

- What is the work that we are repeatedly doing in Kruskal's Algo?
 - Checking for cycles: Linear with BFS, DFS
 - Union-Find Data structure allows us to do this in nearly constant time!

The Union-Find Data Structure

- Efficient way of maintaining partitions
- Supports only two operations
 - Union
 - Find



Equivalence Relations

- An equivalence relation $E(x,y)$ over a domain S is a boolean function that satisfies these properties for every x,y,z in S
 - $E(x,x)$ is true (reflexivity)
 - If $E(x,y)$ is true, then $E(y,x)$ is true (symmetry)
 - If $E(x,y)$ and $E(y,z)$ are true, then $E(x,z)$ is true (transitivity)
- Example 1:
 - $E(x,y)$: Are the integers x and y equal?
 - Then $E()$ is an equivalence relation over integers
- Example 2: Given vertices x and y in a Graph G
 - $E(x,y)$: Are x and y connected?

Equivalence Classes

- An equivalence relation $E()$ over a set S defines a system of *equivalence classes* within S
 - The equivalence class of some element $x \in S$ is that set of all $y \in S$ such that $E(x,y)$ is true
 - Note that every equivalence class defined this way is a subset of S
 - The equivalence classes are disjoint subsets: no element of S is in two different equivalence classes
 - The equivalence classes are exhaustive: every element of S is in some equivalence class
- Example 1:
 - $E(x,y)$: Are the integers x and y equal?
 - Then $E()$ is an equivalence relation over integers
 - The equivalence classes in this case is:

Equivalence Classes for Kruskal's

For Kruskal's algo we will partition all vertices of the graph into disjoint sets, based on the equivalence relation: Are two vertices connected?

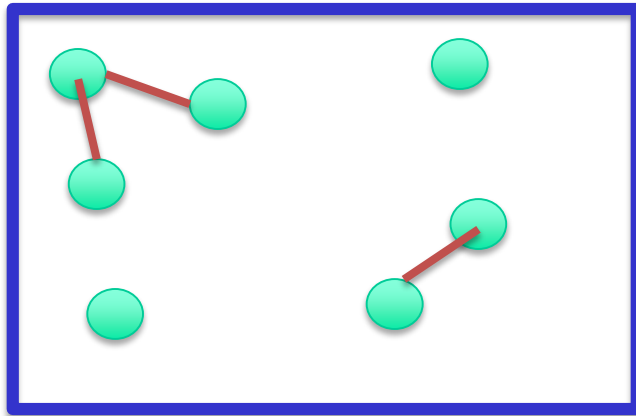
Q: The above equivalence relation partitions the graph into which of the following equivalence classes?

- A. Connected subgraphs
- B. Fully-connected (Complete) subgraphs

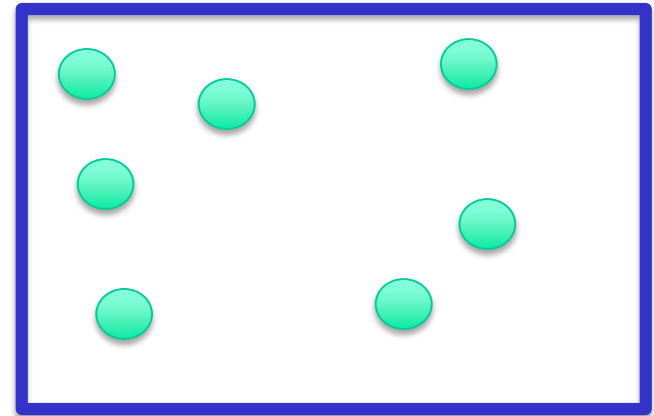
Application of Union-Find to Kruskal's MST

- Vertices that form a connected subgraph will be in the same group
- Connected subgraphs that are disconnected from each other will be in different groups

The graph data structure



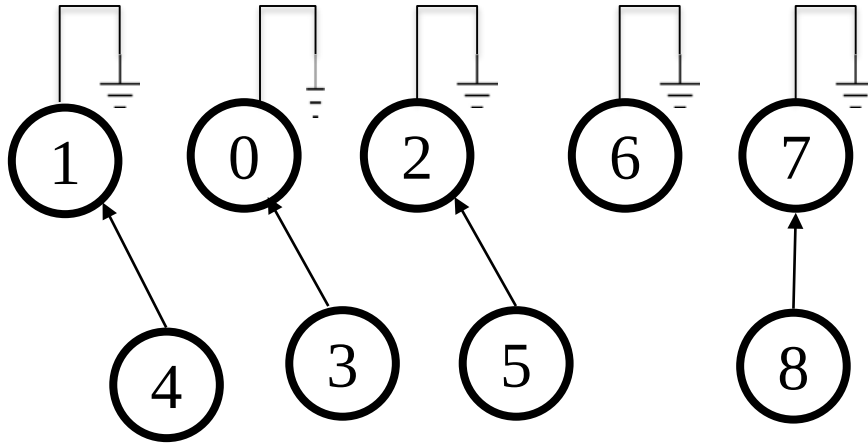
The union-find data structure



Q1: How can we check if adding an edge (v, w) to the graph creates a cycle using the operations supported by union-find?

Q2: In Kruskal's algo what would we like to do if adding the edge does not create a cycle?

Union-find using up-trees



- Each subtree represents a disjoint set
- The root node represents the set that any node belongs to

Perform these operations:

Find(4) =

Find(3) =

Union(1, 0)=

Find(4)=



Running Time of Kruskal's algorithm with union find data structure:

1. Sort edges in increasing order of cost
2. Set of edges in MST, $T = \{\}$
3. For $i = 1$ to $|E|$
 if ($\text{find}(u) \neq \text{find}(v)$) { //If $T \cup \{e_i = u, v\}$ has no cycles
 Add e_i to T
 union($\text{find}(u)$, $\text{find}(v)$)
 }