CSE 373: Data Structures and Algorithms

Lecture 22: Graphs VI
Minimum spanning tree

- **tree**: a connected, directed acyclic graph
- **spanning tree**: a subgraph of a graph, which meets the constraints to be a tree (connected, acyclic) and connects every vertex of the original graph
- **minimum spanning tree**: a spanning tree with weight less than or equal to any other spanning tree for the given graph
Min. span. tree applications

• Consider a cable TV company laying cable to a new neighborhood...
  – Can only bury the cable only along certain paths, then a graph could represent which points are connected by those paths.
  – Some of paths may be more expensive (i.e. longer, harder to install), so these paths could be represented by edges with larger weights.
  – A spanning tree for that graph would be a subset of those paths that has no cycles but still connects to every house.

• Similar situations: installing electrical wiring in a house, installing computer networks between cities, building roads between neighborhoods, etc.
Spanning Tree Problem

• Input: An undirected graph $G = (V, E)$. $G$ is connected.

• Output: $T$ subset of $E$ such that
  – $(V, T)$ is a connected graph
  – $(V, T)$ has no cycles
Spanning Tree Psuedocode

spanningTree():
   pick random vertex v.
   T := {}
   spanningTree(v, T)
   return T.

spanningTree(v, T):
   mark v as visited.
   for each neighbor v_i of v where there is an edge from v to v_i:
      if v_i is not visited
         add edge \{v, v_i\} to T.
         spanningTree(v_i, T)
   return T.
Example of Depth First Search

ST(1)
Example Step 2

\{1,2\}
Example Step 3

\{1,2\} \{2,7\}
Example Step 4

\{1,2\} \{2,7\} \{7,5\}
Example Step 5

\{1,2\} \{2,7\} \{7,5\} \{5,4\}
Example Step 6

{1,2} {2,7} {7,5} {5,4} {4,3}

ST(1)  
ST(2)  
ST(7)  
ST(5)  
ST(4)  
ST(3)
Example Step 7

{1,2} {2,7} {7,5} {5,4} {4,3}
Example Step 8

{1,2} {2,7} {7,5} {5,4} {4,3}

ST(1)  ST(2)  ST(7)  ST(5)  ST(4)
Example Step 9

{1,2} {2,7} {7,5} {5,4} {4,3}

ST(1)
ST(2)
ST(7)
ST(5)
ST(4)
Example Step 10

{1,2} {2,7} {7,5} {5,4} {4,3}
Example Step 11

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\} \{5,6\}
Example Step 12

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\} \{5,6\}
Example Step 13

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\} \{5,6\}
Example Step 14

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\} \{5,6\}
Example Step 15

{1,2} {2,7} {7,5} {5,4} {4,3} {5,6}
Example Step 16

\[
\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\} \{5,6\}
\]

ST(1)
Minimum Spanning Tree Problem

• Input: Undirected Graph $G = (V, E)$ and a cost function $C$ from $E$ to non-negative real numbers. $C(e)$ is the cost of edge $e$.

• Output: A spanning tree $T$ with minimum total cost. That is: $T$ that minimizes

$$C(T) = \sum_{e \in T} C(e)$$
Observations about Spanning Trees

• For any spanning tree $T$, inserting an edge $e_{new}$ not in $T$ creates a cycle

• But
  – Removing any edge $e_{old}$ from the cycle gives back a spanning tree
  – If $e_{new}$ has a lower cost than $e_{old}$ we have progressed!
Find the MST
Two Different Approaches

Prim’s Algorithm
Looks familiar!

Kruskals’s Algorithm
Completely different!
Prim’s algorithm

**Idea:** Grow a tree by adding an edge from the “known” vertices to the “unknown” vertices. Pick the edge with the smallest weight.
**Prim’s algorithm**

Starting from empty $T$, choose a vertex at random and initialize $V = \{A\}$, $T = \{\}$
Prim’s algorithm

Choose the vertex $u$ not in $V$ such that edge weight from $u$ to a vertex in $V$ is minimal (greedy!)

$V = \{A, C\}$

$T = \{ (A, C) \}$
Prim’s algorithm

Repeat until all vertices have been chosen

$V = \{A,C,D\}$

$T = \{(A,C), (C,D)\}$
Prims’s algorithm

\[ V = \{A, C, D, E\} \]

\[ T = \{ (A, C), (C, D), (D, E) \} \]
Prim’s algorithm

\[ V = \{A,C,D,E,B\} \]
\[ T = \{ (A,C), (C,D), (D,E), (E,B) \} \]
Prim’s algorithm

\[ V = \{ A, C, D, E, B, F \} \]

\[ T = \{ (A, C), (C, D), (D, E), (E, B), (B, F) \} \]
Prim’s algorithm

\[ V = \{A,C,D,E,B,F,G\} \]
\[ T = \{ (A,C), (C,D), (D,E), (E,B), (B,F), (E,G) \} \]
Prim’s algorithm

Final Cost: $1 + 3 + 4 + 1 + 1 + 6 = 16$
Prim's Algorithm Implementation

**Prim()**:  
for each vertex \( v \): // Initialization  
  \( v \)'s distance := infinity.  
  \( v \)'s previous := none.  
  mark \( v \) as unknown.  
choose random node \( v_1 \).  
\( v_1 \)'s distance := 0.  
List := \{all vertices\}.  
\( T := {} \).

while List is not empty:  
  \( v := \) remove List vertex with minimum distance.  
  add edge \{v, \( v \)'s previous\} to \( T \).  
  mark \( v \) as known.  
  for each unknown neighbor \( n \) of \( v \):  
    if distance(\( v, n \)) is smaller than \( n \)'s distance:  
      \( n \)'s distance := distance(\( v, n \)).  
      \( n \)'s previous := \( v \).

return \( T \).
Prim’s algorithm Analysis

• How is it different from Djikstra's algorithm?

• If the step that removes unknown vertex with minimum distance is done with binary heap the running time is:

$$O(|E|\log |V|)$$
Kruskal’s MST Algorithm

Idea: Grow a forest out of edges that do not create a cycle. Pick an edge with the smallest weight.

G=(V,E)
Example of Kruskal 1

{7,4} {2,1} {7,5} {5,6} {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 2

{7,4} {2,1} {7,5} {5,6} {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}
0 1 1 2 2 3 3 3 3 4
Example of Kruskal 2

\{7,4\} \{2,1\} \{7,5\} \{5,6\} \{5,4\} \{1,6\} \{2,7\} \{2,3\} \{3,4\} \{1,5\}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 3

{7,4} {2,1} {7,5} {5,6} {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 4

\{7,4\} \{2,1\} \{7,5\} \{5,6\} \{5,4\} \{1,6\} \{2,7\} \{2,3\} \{3,4\} \{1,5\}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 5

{7,4}  {2,1}  {7,5}  {5,6}  {5,4}  {1,6}  {2,7}  {2,3}  {3,4}  {1,5}
0    1    1    2    2    3    3    3    3    4
Example of Kruskal 6

\{7,4\} \{2,1\} \{7,5\} \{5,6\} \{5,4\} \{1,6\} \{2,7\} \{2,3\} \{3,4\} \{1,5\}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 7

\{7, 4\} {2, 1} {7, 5} {5, 6} {5, 4} {1, 6} {2, 7} {2, 3} {3, 4} {1, 5}
Example of Kruskal 7

{7,4} {2,1} {7,5} {5,6} {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 8,9

\begin{itemize}
\item \{7,4\}
\item \{2,1\}
\item \{7,5\}
\item \{5,6\}
\item \{5,4\}
\item \{1,6\}
\item \{2,7\}
\item \{2,3\}
\item \{3,4\}
\item \{1,5\}
\end{itemize}

\begin{center}
\begin{tabular}{ccccccccccc}
0 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 3 & 4 \\
\end{tabular}
\end{center}
Kruskal's Algorithm Implementation

Kruskals():
    sort edges in increasing order of length \((e_1, e_2, e_3, ..., e_m)\).

    \[ T := \{ \}. \]

    \[ \text{for } i = 1 \text{ to } m \]
    \[ \quad \text{if } e_i \text{ does not add a cycle:} \]
    \[ \quad \quad \text{add } e_i \text{ to } T. \]

    \[ \text{return } T. \]

• But how can we determine that adding \(e_i\) to \(T\) won't add a cycle?