Visualizing Graphs

Cole posted this link last year: http://www.webgraphviz.com/

You can copy the definition of a graph in the text box and generate a picture of it:

```plaintext
digraph G { 
a -> b
b -> c
c -> d
}
```
Algorithm Visualizations

https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

BFS
DFS
Kruskal’s MST
Prim’s MST
Fibonacci Heaps
etc.
Minimum Spanning Trees
CS 320, Fall 2017

Dr. Geri Georg, Instructor
georg@colostate.edu

Some materials adapted from Prof. Wim Bohm
Algorithmically....

Given a set of locations, with positive weights of some sort, we want to create a network \((T)\) that connects all nodes to each other with minimal sum of distances \((c_e, c \in T)\).

\[
G = (V, E)
\]

\[
\sum_{e \in T} c_e = 50
\]
An Example
An Example
An Example
Clustering
Biological Prediction

neuron

dendrites

dendrite length predicted using MSTs

real dendrites
MST predictions

12 days Reconstruction
12.5 days
13 days

Synthetic dendrites

25 μm
Greedy MST Algorithms

**Kruskal's algorithm.** Start with $T = \emptyset$. Consider edges in ascending order of cost. Add edge $e$ to $T$ unless doing so would create a cycle.

**Reverse-Delete algorithm.** Start with $T = E$. Consider edges in descending order of cost. Delete edge $e$ from $T$ unless doing so would disconnect $T$.

**Prim's algorithm.** Start with some node $s$ and greedily grow a tree $T$ from $s$. At each step, add the cheapest edge $e$ to $T$ that has exactly one endpoint in $T$, i.e. without creating a cycle.
Cut Property

Let $S$ be a subset of nodes, and let $e$ be the min cost edge with exactly one endpoint in $S$. Then the MST $T$ contains $e$.

Proof: $T$ is a spanning tree so it has to have a path from $v$ to $w$. Follow that path to $w$. It has to go somehow from $v$ to $v'$ (last node on $P$ in $S$), to $w'$ (1st node on $P$ in $V-S$), then somehow to $w$. 
Cut Property
Let $S$ be a subset of nodes, and let $e$ be the min cost edge with exactly one endpoint in $S$. Then the MST $T$ contains $e$.

Proof continued: Now exchange $e$ for $e'$ to get:
$T' = T - \{e'\} \cup \{e\}$
We claim $T'$ is a spanning tree. It is connected since any path can use $e$ instead of $e'$. It is acyclic since we substituted $e$ for $e'$. $e$ is the cheapest edge with just 1 node in $S$, so $c_e < c_{e'}$, and the cost of $T' < $ the cost of $T$
Kruskal’s Algorithm
Create sets for each tree in the forest – initially one for each \( u \in V \)
Sort edges from least cost to most
For each edge \((u,v)\) taken in increasing cost order, if \( u \) and \( v \) are in different trees, add the edge to the MST and merge the vertices of the 2 trees
Muddy City – Kruskal’s
For each edge \((u,v)\) taken in increasing cost order, if \(u\) and \(v\) are in different trees, add the edge to the MST and merge the vertices of the 2 trees.
Kruskal’s
Recall ....

What algorithm did we explore Fri?
How did we start it? What did we consider first – vertices or edges, and what did we do with them?
In what order did we consider vertices/edges?
What happened if we had multiple edges with the same weight and we tried taking them in different order?
Why is this cut-property thing important? (Why is the greedy choice safe?)

A is a subset of E included in some MST, T. (S, V-S) is a cut that respects A, and (u,v) is a light edge crossing the cut that is not in T. At least 1 edge in T has to be on the path from u to v, e.g. (x,y). Removing (x,y) breaks T into 2 parts. Adding (u,v) connects them to make a new ST, T'.

w(T') ≤ w(T) since (u,v) is a light edge, so T' is an MST and (u,v) is a safe edge.
Usefulness of MST?

- MSTs are used in finding the lowest cost interconnections in networks of various kinds (cable, utilities, etc.), finding clusters, predicting biological growth, and other applications.
Micro-Survey 3

Complexity?

Kruskal’s:

- Join trees in a forest into a single tree.
- Sort edges by weight: $O(E \lg E)$
- Joining sets uses union by rank: “add” the set with the lower rank to the other set. Next apply path compression: $O(m \alpha(n))$ where $m$ is the number of disjoint-set ops on $n$ elements.
- The sorting part dominates: $O(E \lg E)$.
- **BUT** $E \lg E \leq E \lg V^2$, which is $= 2E \log V$, and for $O()$ we throw away the constant and get $O(E \lg V)$ which is more aesthetically pleasing.
Micro-Survey 3.5

Complexity?

Prim’s:

• Linear: setup, bookkeeping
• Process all edges, potentially perform a heap op on each
• Heap ops dominate: $O(E \lg E)$.
• Same argument to get $O(E \lg V)$ which is more aesthetically pleasing.
• If a Fibonacci heap is used, the amortized heap ops can make the complexity $O(E + V \lg V)$
Micro-Survey 4

Why doesn’t MST solve the traveling salesman problem?

• TSP starts from one vertex, visits all others, and ends up at the first (so direction matters). The path is the shortest path. It is really looking for a Hamiltonian cycle.

• MST is looking for the lowest weight tree that connects all the vertices (direction doesn’t matter – the point is that there is a path to every vertex).

• MST can be used to approximate TSP by performing a pre-order DFS of the tree and constructing a list of vertices the first time we encounter them. This has been shown to be at most 2x the optimal path length.
Micro-Survey 5

Why was the Canvas graph quiz a different format?

• The Canvas graphics quiz and a new Canvas MST quiz are ‘reading’ quizzes, designed to help direct your reading of the Cormen book. They are meant to be taken while studying the book. We’ll do at least 2 quizzes this way, and see how it goes.
Prim’s Algorithm

Initialize $S = \text{any node}$. This will be the MST root.

Apply cut property to $S$: add min cost edge $(v, w)$ where $v$ is in $S$ and $w$ is in $V-S$, and add $w$ to $S$.

Repeat until $S = V$. 

![Diagram of Prim's Algorithm](image-url)
Prim(G,r)

foreach (u ∈ V)
    a[u] ← ∞; u.π = Ø
a[r]= 0

min priority queue Q = {}

foreach (u ∈ V) insert u onto Q (key: ‘a’ value)

set S ← {}

while (Q is not empty) {
    u ← extract min element from Q
    S ← S ∪ { u }

    foreach (edge e = (u, v) incident to u)
        if ((v ∉ S) and (c_e < a[v]))
            decrease priority a[v] to c_e
            v.π = u
Muddy City 2 – Prim’s
min priority queue \( Q = \{ \} \)

\[
\text{foreach} \ (u \in V) \ \text{insert} \ u \ \text{in} \ Q
\]

set \( S \leftarrow \{ \} \)

\[
\text{while} \ (Q \text{ is not empty}) \{
\quad u \leftarrow Q \text{ min element (delete)}
\quad S \leftarrow S \cup \{ u \}
\quad \text{foreach edge} \ e = (u, v)
\quad \quad \quad \text{if} \ ((v \notin S) \ \text{and} \ (c_e < a[v]))
\quad \quad \quad \quad a[v] = c_e
\quad \quad \quad v.\pi = u
\}
\]

\[
\begin{array}{|c|c|c|}
\hline
V & \text{adj} & a[u] & \pi \\
\hline
a & b,f,g & 0 & \emptyset \\
\hline
b & a,c,d,e,f & \infty & \emptyset \\
\hline
c & b,d & \infty & \emptyset \\
\hline
d & b,c,e,i & \infty & \emptyset \\
\hline
e & b,d,f,h,i & \infty & \emptyset \\
\hline
f & a,b,e,h,g & \infty & \emptyset \\
\hline
g & a,f,h,k & \infty & \emptyset \\
\hline
h & e,f,g,i,k & \infty & \emptyset \\
\hline
i & d,e,h,j & \infty & \emptyset \\
\hline
j & i,k & \infty & \emptyset \\
\hline
k & g,h,j & \infty & \emptyset \\
\hline
\end{array}
\]
Weight = 23

Prim's

\[ Q = \{ a, b, c, d, e, f, g, h, k \} \]

\[ S^2 = \{ a, f, e, b, c, d, i, h, k, g \} \]
Image Credits

muddyCity: http://www.utdallas.edu/~besp/teaching/mst-applications.pdf
clusteringMST2: https://www.researchgate.net/figure/236106483_fig2_Figure-2-Unrooted-minimum-spanning-tree-network-showing-genetic-relationship-among
dendritePP: http://www.pnas.org/content/109/27/11014.full.pdf
bean: http://clipart-library.com/bean-people.html