Minimum Spanning Tree

Problem: Given a graph \( G = (V, E) \) with weights \( w : E \rightarrow \mathbb{R}^+ \) on the edges, find a subset of the edges that connects all the vertices and has minimum weight.

E.g.:

- Weight 8
- Weight 7
- Weight 4

The edge subset will be a tree, why?

called the minimum spanning tree

Greedy algorithms will find min. spanning trees you've seen some of this in MATH 239.

In fact, there are several possible correct greedy approaches, with different implementation challenges.

e.g. - add cheapest edge first, never build a cycle

Kruskal's alg.

- grow connected graph from one vertex

Prim's alg.

- throw away expensive edges, never disconnect
Kruskal's Algorithm
Order edges by weight \( e_1 \ldots e_m \)
\[ w(e_i) \leq w(e_{i+1}) \]
\( T \leftarrow \emptyset \)
for \( i = 1 \ldots m \)
if \( e_i \) does not make a cycle with \( T \) then
\[ T \leftarrow T \cup \{ e_i \} \]
end

General situation

Connected components

\( e \) makes a cycle with \( T \) iff \( e \) joins vertices in same connected component.

e.g. edge \( e \) makes a cycle \( \Rightarrow \) throw it out

edge \( f \) does not \( \Rightarrow \) add \( f \) to \( T \)

Correctness - an exchange proof
Let \( T \) have edges \( t_1 \ldots t_{n-1} \)
Prove by induction on \( i \) that there is a MST matching \( T \) on the first \( i \) edges

basis case: \( i = 0 \)
Assume by induction that there is a MST $M$ matching $T$ on the first $i-1$ edges.

Let $t_i = (a, b)$ and let $C$ be the connected component of $T$ containing $a$. Note: when the algorithm considers $t_i$ all edges of weight $< w(t_i)$ have been considered, and none of them go from $C$ to $V - C$.

Look at a path in $M$ from $a$ to $b$. It must cross from $C$ to $V - C$, say on edge $e'$. Then $w(e) \leq w(e')$ by Note above, so $e'$ is later in ordering.

Exchange: Let $M' = (M - e')\cup e''$

Claim $M'$ is a MST.

Then we're done, since $M'$ matches $T$ on $i$ edges.

Pf: $M'$ is a spanning tree because it connects all vertices (replace $e'$ by blue path from $a'$ to $a$, in $M'$, edge $e$; from $b$ to $b'$ in $M$).

and has same no. of edges,

$w(M') = w(M) - w(e') + w(e) \leq w(M)$

so $M'$ is a min. spanning tree.
Implementing and analyzing Kruskal’s Algorithm

Graph \( G = (V,E) \) \( |V| = n \) \( |E| = m \)

\( O(m \log m) \) to sort edges = \( O(m \log n) \)

because \( m \leq n^2 \) so \( \log m \) is \( O(\log n) \)

Then we need to maintain connected components as we add edges. Also test if edge \((a, b)\) has 

\( a, b \) in same component, or different components 

(don’t add edge) \hspace{1cm} (do add edge)

Union-Find Problem

Maintain a collection of disjoint sets

Operations

- \( \text{Find}(x) \) — which set contains element \( x \)?
- \( \text{Union} \) — unite two sets

In our case the elements are vertices and the sets are connected components of \( T \), the tree so far

This Abstract Data Structure has a very simple implementation that gives \( O(m \log n) \) for Kruskal.

There is a fancier implementation — CS 466

[Alg. is pretty simple, analysis is hard and true runtime involves Ackerman’s fn, very slow growing]
Simple implementation of Union Find:
Keep array \( S \) \([1 \ldots n]\), \( S[i] = \) component of element \( i \)
and keep linked list of elements in each set

\[
\begin{align*}
C_1 &= 1, 3, 5, 6 \\
C_2 &= 2, 4 \\
C_3 &= 7
\end{align*}
\]
\[
S = \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
C_1 & C_2 & C_1 & C_2 & C_1 & C_1 & C_3
\end{array}
\]

Find is \( O(1) \), join 2 linked lists \( O(1) \) and union - must rename one of the two sets
so \( O(n) \) in worst case

But renaming smaller set does better!

\[
\text{e.g. to unite } C_1 \text{ and } C_2 \text{ do } C_1 \leftarrow C_1 \cup C_2
\]

must fix \( S(2) \leftarrow 1, S(4) \leftarrow 1 \).

If an element's set number changes, then its set (more than) doubles
This happens \( \leq \log n \) times
Therefore total renaming work is \( O(n \log n) \)

Total run time
\[
O(m \log n) + O(m) + O(n \log n)
\]

\[
\frac{\text{sort}}{\text{Finds}} \frac{\text{unions}}{
\text{so } O(m \log n) \text{ assuming G is connected}}
\]

\[
\text{so } m \geq n - 1
\]