Prim's MST Algorithm

$$T = \{3\}$$
make sure that $$T$$ always remain a tree

$$T = (V_T, E_T)$$

A-C
C-D
A-B
C-F
E-F

Greedy method

$$T = \{3\}$$
add "safe" edge

light edge

$$\left(V_T, V - V_T \right)$$
we always pick a lighter edge that crosses this cut.
while PQ not empty:
  \(\times = \text{delete min}(PQ)\)
  for all \((x, y) \in E:\)
    \[\text{if } c_{xy} \geq w(x, y) :\]
    \[\begin{align*}
    \text{Cost}(y) &= w(x, y) \\
    \text{decrease key}(PQ, y)
    \end{align*}\]

1. building PQ: \(O(|V|^2)\)
2. \(|V|\), delete min: \(|V| \log |V|\)
3. \(|E|\), decrease key: \(|E| \log |V|\)

\(O((|V| + |E|) \log (|V|))\)

---

Huffman Coding

\(\Rightarrow\) binary code
mississippi:

\[
\begin{align*}
  i & = 00 \\
  m & = 01 \\
  p & = 10 \\
  s & = 11 \\
\end{align*}
\]

fixed length coding

\[
\Xi = \{i, m, p, s\}
\]

\[
0100111100111100101000
\]

length < 22 bits

full tree

\[
\begin{align*}
  i & = 000 \\
  m & = 001 \\
  p & = 010 \\
  s & = 011 \\
  x & = 100 \\
\end{align*}
\]

length: 12 x 3 = 36 bits
allowing
Variable length coding

encoded (mississippi) = 34 bits

Mississippi

\[ \text{encoded} \]

\[ 001 \ 000 \ 1 \ 011 \ 011 \ 000 \]

decode

\[ \text{prefix free code} \]

no code is a prefix of another

walk from the root to a leaf
output the leaf character

unique decoding

fixed length

Mississippi

\[ \text{freq} \]

\[ \frac{i \ - \ 0 \ - \ m \ - \ 0 \ - \ p \ - \ s \ - \ x}{0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0} \]
Implementation: PQ. \mid s \mid = m
Input: string $S$

1. $\Sigma \leftarrow$ extract unique characters from $S$
2. Compute $f(x) = \text{freq}(x) \quad \forall x \in \Sigma$

Create a PQ with $(x, f(x))$

while

$a = \text{delete min}(\text{PQ})$

$b = \text{delete min}(\text{PQ})$

Create a new node $z$ with $a$ & $b$ as children

$f(z) = f(a) + f(b)$

Insert $(z, f(z))$ in PQ.

After each merge, the number of nodes decreases by 1.

Total # of merge = $n$.

$2$ delete mins $\quad \text{per merge step}$

$1$ insert

$\text{delete min} \quad O(\log n)$

$\text{insert} \quad n = |\Sigma|$

lecture16 Page 6
deleting \( n = |X| \) \( n = |Y| \)

\[ \text{total cost is } n \log n \]

Proof that greedy strategy works.

Pick the two smallest freq nodes and merge them.

Outline:

Assume that \( T \) is an optimal prefix-free code.

Let \( f(x) \leq f(y) \), with \( f(x) \leq f(y) \) are the 2 least frequency nodes.

Let \( T \) be optimal.

\( a \leq b \) are at the largest depth in \( T \).

Show that \( X \) \( T' \) such that \( x \) \( y \) are neighbors in \( T' \) instead of \( a \leq b \).

\( T' \) is also optimal.
Sure, I can help you with that. Let's break it down step by step:

1. We want to show that the new tree $\Gamma''$ is also optimal.

2. Since $\Gamma$ is optimal, we have:
   
   \[ \text{Cost}(\Gamma'') \leq \text{Cost}(\Gamma') \leq \text{Cost}(\Gamma) \]

3. To prove $\Gamma''$ is optimal, we need to show:
   
   \[ \text{Cost}(\Gamma'') = \text{Cost}(\Gamma') \]

4. Define $d_{\Gamma}(x)$ as the height of node $x$ in tree $\Gamma$.

5. For tree $\Gamma'$, we have:
   
   \[ \text{Cost}(\Gamma') = f(x) \cdot d_{\Gamma}(x) + f(a) \cdot d_{\Gamma}(a) - f(x) \cdot d_{\Gamma'}(x) - f(a) \cdot d_{\Gamma'}(a) \]

6. For tree $\Gamma''$, we have:
   
   \[ \text{Cost}(\Gamma'') = f(x) \cdot d_{\Gamma}(x) + f(a) \cdot d_{\Gamma}(a) - f(x) \cdot d_{\Gamma''}(x) - f(a) \cdot d_{\Gamma''}(a) \]

7. Subtracting the two equations, we get:
   
   \[ \text{Cost}(\Gamma) - \text{Cost}(\Gamma') = [f(a) - f(x)] \cdot [d_{\Gamma}(a) - d_{\Gamma}(x)] \]

8. Since $f(x)$ and $d_{\Gamma}(a)$ are non-negative, the expression inside the brackets is non-negative.

Therefore, $\Gamma''$ is also optimal.
because \( d_G(c) \) is the largest possible depth in \( G \)

\[
\Rightarrow \quad \text{Cost}(T) - \text{Cost}(T') \geq 0
\]

\[
\Rightarrow \quad \text{Cost}(T) \geq \text{Cost}(T')
\]

\[
\Rightarrow \quad \text{Cost}(T) = \text{Cost}(T')
\]

---

Set Cover

1. make a graph
   - each edge \((u, v)\) \(\Rightarrow\)
   - \(d(u, v) \leq 50\)

2. place hospitals
3. no one should have to travel more than 50 mile
   - minimize the # of hospitals

Optimal = 3

Set cover problem is NP-complete

\(\Rightarrow\) Unlikely to be solvable in polynomial time

"Greedy" is: pick a city that will cover the largest # of uncovered cities

Not optimal!
Q: How bad can the solution be?

If $k$ is optimal # of hospitals, then greedy solution is at most $k \cdot \log n$ away.

$n = |V|$ in $G$. Approximation factor.

We cannot do better than $\log n$ in poly time.