Review:

Single Source Shortest Path (SSSP) Algorithms

1) BFS  \( O(N + E) \)
2) Dijkstra’s Algorithm  \( O((N + E) \log N) \)
3) Bellman-Ford Algorithm  \( O(N \times E) \)
4) DAG-SSSP-Algorithm  \( O(N + E) \)

Greedy Algorithms:

1) Scheduling:

Thm: First finish time is optimal

Proof: Exchange Argument

2) Compression (Huffman Codes)

Goal: Encode text with \( T \) letters from an alphabet \( \Gamma \)

with \( n \) letters and frequencies \( \{ f_i : i \in \Gamma \} \)

Ex: \( \Gamma = \{A, B, C, D\} \quad T = 100 \)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Frequency</th>
<th>Code1</th>
<th>Code2</th>
<th>Code3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>80</td>
<td>00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>01</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>11</td>
<td>11</td>
<td>101</td>
</tr>
</tbody>
</table>

Cost: 200 110 130

Prefix Problem: In Code 2, how to decode

10 = BA or C
- B and C have same prefix

**Prefix-Free Property:**
No codeword can be prefix of another

**Tree Representation**

**Binary tree:**
- 0 in \(i^{th}\) position \(\iff\) go left in level \(i\)

**Codewords on leaves \(\iff\) prefix-free**

![Full binary tree](image)

- Every node has 0 or 2 children
- Why full? Not optimal

Q: Is the code \(\{0, 11, 100, 101\}\) optimal?

How do we find optimal codes in general?

\[
\text{Cost}(T) = \sum_{i=1}^{n} f_i \times \text{depth of } i
\]

**Greedy:** Smallest \(f_i\) should be the leaves with largest depth, say \(f_A\) and \(f_B\) \(\Rightarrow\) Build tree bottom up

\[
T \quad \text{How to continue?}
\]

- New alphabet with \(n-1\) letters: \(A' = A \text{ or } B\)
\[
T \quad \rightarrow \quad T'
\]
\[
l_0 \quad \rightarrow \quad l_0 + l_0
\]
\[
\text{cost}(T) = \text{cost}(T') + \delta A + \delta B
\]

**Huffman Code**

- Start with \( F = f_1, \ldots, f_n \)
- Find lowest two, \( f_i \) and \( f_j \)
- Remove \( f_i, f_j \) from \( F \)
- Add \( f_i + f_j \) to \( F \)
- Iterate

**Example**

\( f_1 = 80, f_2 = 10, f_3 = 5, f_4 = 5 \)

\[
\begin{align*}
\text{original} & \quad \rightarrow \quad \{ f_1 = 80, f_2 = 10, f_3' = 10 \} \\
\text{iteration } 1 & \quad \rightarrow \quad \{ f_1 = 80, f_3 = 20 \} \\
\text{iteration } 2 & \quad \rightarrow \quad \{ f_1 = 80, f_3'' = 20 \}
\end{align*}
\]
Huffman (f)

Input: f[1;...;n] frequencies
Output: encoding tree with n leaves

H = priority queue
For i = 1;...; n : insert (i, f[i])
For k = n+1;...; 2n-1
    u = Delete Min, v = Delete Min
    add (k, n) and (k, v) to E
    f[k] = f[u] + f[v]
    insert (k, f[k])

Correctness Proof:
Claim: Order f_1 ≤ f_2 ≤ ... . Then ∃ optimal encoding tree s.t. f_i and f_j are assigned to two siblings which are leaves of maximal depth

Proof: Choose optimal encoding tree
Choose leaf j of maximal depth
Full tree ⇒ must have sibling j
depth(i) maximal ⇒ j is leaf

\[ \begin{array}{c}
  \text{exchange } f_i, f_j \\
  \text{with } i, j \\
  \text{if needed}
  \end{array} \]
⇒ cost can only go down ⇒ new optimal tree with properly from claim
**Lemma:** Huffman finds optimal tree

*By induction:

**Base:** $n = 2$ trivial

**PF:** $n ightarrow n+1$

$T_{n+1}$ optimal. By Claim 1

$\exists T_{n+1}$ s.t. $T_{n+1}$ is optimal and 1 and 2 are leaves

Define $T_n$ by pruning $T_{n+1}$

$\triangle T_{n+1} \rightarrow \triangle T_n$

$\begin{align*}
\text{cost}(H_{n+1}) &= f_1 + f_2 + \text{cost}(H_n) \\
\text{cost}(T_{n+1}) &= f_1 + f_2 + \text{cost}(T_n)
\end{align*}$

*By induction

$\text{cost}(H_n) \leq \text{cost}(T_n)$

$\Rightarrow \text{cost}(H_{n+1}) \leq \text{cost}(T_{n+1})$

$\Rightarrow H_{n+1}$ is optimal.

---

**Kruskal and Prim**

Given Graph

$G = (V, E)$

$\begin{align*}
\text{Goal:} \text{Find edge subset } T \subseteq E \text{ with minimal total weight } W(T) &= \sum_{e \in T} W_e \text{ which keeps } G
\end{align*}$
Rem: We can choose T to have no cycles, i.e., to be a tree
⇒ Problem become to find minimum spanning tree (MST)

Trees

Def: An undirected connected graph without cycles

Claim 1: If G is connected and contains a cycle, removing any edge on a cycle can't disconnect it

By Picture:

\[ \text{cycle} \quad \xrightarrow{\text{removed edge}} \quad \text{want to prove} \quad \begin{array}{c}
\text{still connected in } G \\
\text{path } u \rightarrow w \text{ in } G \\
\end{array} \]

Claim 2: A tree T with n nodes has n-1 edges

By: Remove all edges, add them back one by one.

Each time, we reduce # of connected components (cc) by 1
Note: we can never add an edge within a cc, since that would create a cycle.

Claim 3: If G is connected with n nodes and
n-1 edges ⇒ G is a tree

Proof: Follows from Claim 1+2

Greedy Algorithm (Kruskal)
- Choose edges in order of weights
- Skip an edge if it creates cycle

Prove it is optimal
- Final data structure to implement it

The Cut Property:
- Suppose X∈E is part of a MST of G
- Let S∈V n.th. X has no edge from S to V\S
- Let e be the lightest edge from S to V\S
- Then: X∪e is part of some MST
PF: Let $T$ be the MST s.t. $X \subseteq T$
let $e' \in T$ s.t. it connects $S$ to $V \setminus S$
add $e$ to $T \Rightarrow$ creates cycle
remove $e' \Rightarrow$ still connected
$n-1 + 1 - 1$ edges $\Rightarrow$ new tree $T'$
$W(T') = W(T) - w_{e'} + w_e \leq W(T)$

Remark: The partition $S, \bar{S} = V \setminus S$ is called a cut

Prove Kruskal's algorithm MST

Recall: Adds lightest edge $e$ not creating cycle

$X$ edges at time $t \Rightarrow$

$S \cup \{X\} \cup V \setminus S$
⇒ If \( X \) is part of MST, so is \( X_u \) for \( u \in S \)

**Proof (K Finds MST)**

By induction

**Base Case** \( X = \emptyset \)

\(|X| = k \rightarrow |X| = k + 1 \) Cut property

**Implementation of Kruskal**

How to check for cycles?

keep track of connected comp. (cc)

**Disjoint Set Datastructure** ("Union Find")

- \text{makeSet}(x) \quad \text{makes singleton containing } x
- \text{find}(x) \quad \text{which set does } x \text{ belong to}
- \text{union}(x, y) \quad \text{merges sets contain. } x, y

\begin{align*}
\text{kruskal}(G, w) & \\
\{ & n = |V|, m = |E| \\
& n \text{ make set} \\
& \text{sort } 0(m \log m) \\
& = 0(m \log n) \\
& 2m \text{ find} \\
& n - 1 \text{ union} \\
\end{align*}

| For all \( v \in V \) \text{ makeSet}(v) | \begin{align*}
& X = \{ v \} \\
& \text{Sort edges in } E \text{ by } w(\cdot) \\
& \forall \{u, v\} \in E \text{ in that order} \\
& \quad \text{if } \text{find}(u) \neq \text{find}(v) \\
& \quad \quad \text{add } \{u, v\} \text{ to } X \\
& \quad \quad \text{union}(u, v) \\
& \quad \text{return } X
\end{align*} |
Running time

makeset O(n)  union, find log n

\[ \text{total: } O((m+n) \log n) \]

Master Algorithm

\[ X = \{ 5 \} \]
repeat until \( |X| = n-1 \)

pick \( S \subseteq V \) n-1 th. \( X \) contains no edge between \( S \) \( \bar{S} \)
let \( e \) be minimum weight edge between \( S \) \( \bar{S} \)
\[ X = X \cup \{ e \} \]

Prims Alg:

Choose \( S \) to be connected

Choose \( S \), \( A \rightarrow C \), \( A \rightarrow D \), \( A \rightarrow C \)

Effectively, need to minimize

\[ \text{cost}(v) = \min_{u \in S} w(u, u) \]
to decide which new vertex $u$ to add

$Prim(G, w)$

$\forall u \in V \; \text{cost}(u) = \infty, \; \text{prev}(u) = \text{nil}$

Pick any $u_0 \in V$

$\text{cost}(u_0) = 0$

$\forall u \in V \; \text{insert key}\ (v, \text{cost}(v))$

while queue non empty

$v = \text{Delete Min}$

$\forall \{v, u\} \in E$

if $\text{cost}(u) > w(v, u)$

$\text{cost}(u) = w(v, u)$

$\text{prev}(u) = v$

$\text{Decrease Key}(u)$

$O(n) \; \text{insert, delete} \; O(m) \; \text{decrease key}$

$\rightarrow O((n+m) \log n)$ running time

Example

\begin{figure}
\centering
\includegraphics[width=\textwidth]{example}
\end{figure}

\begin{table}
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
Set $S$ & $A$ & $B$ & $C$ & $D$ & $E$ & $F$ \\
\hline
$\{}$ & 0/nil & $\infty$/nil & $\infty$/nil & $\infty$/nil & $\infty$/nil & $\infty$/nil \\
$A$ & $\infty$/nil & 5/A & 6/A & $\infty$/nil & $\infty$/nil & $\infty$/nil \\
$A, D$ & $\infty$/nil & 2/D & 1/B & $\infty$/nil & $\infty$/nil & $\infty$/nil \\
$A, D, B$ & $\infty$/nil & $\infty$/nil & $\infty$/nil & 5/C & 4/F & $\infty$/nil \\
$A, D, B, C$ & $\infty$/nil & $\infty$/nil & $\infty$/nil & $\infty$/nil & 3/C & $\infty$/nil \\
$A, D, B, C, F$ & $\infty$/nil & $\infty$/nil & $\infty$/nil & $\infty$/nil & $\infty$/nil & $\infty$/nil \\
\hline
\end{tabular}
\caption{Cost/pre table}
\end{table}