OFF TODAY: 1:15 - 2:15pm

Emails

Exam 1

Huffman Coding.

Input: n characters in a text

S: alphabet

\( \forall x \in S, f_x: \text{fraction of times } x \text{ appears in the text} \)

Objective: To obtain a prefix code \( R \) s.t.

\[
ABL := \sum_{x \in S} f_x |R(x)| \text{ is minimized}
\]

Avg. bits per letter

- Binary trees in which characters in \( S \) are associated with leaves give us prefix codes.
- Binary trees correspond to optimal prefix codes.
must be full binary trees.

- We need:
  - shape of the binary tree
  - assignment of classes in $S$ to leaves in the tree.

- Observation: two lowest freq. letters are assigned to leaves at the bottommost level. Since the binary tree corr. to an optimal prefix code is full, there must be two leaves in the bottommost level that are siblings.

**Lemma**: There exist optimal prefix codes with corresponding binary tree $T^*$ in which the two lowest freq. letters are assigned to sibling leaves in the bottommost level.
Huffman (S)

if |S| ≤ 2 then

Done (assign 0 to one letter & 1 to another)

else

Build MinHeap (S) \(O(n)\)

ExtractMin \(y^*, z^*\): two lowest freq. letters in \(S\) \(O(\log n)\)

\(S' \leftarrow S \setminus \{y^*, z^*\} \cup \{0, 1\}\)

\(f_w \leftarrow fy^* + fz^*\)

\(\text{Insert}(w) \rightarrow O(\log n)\)

\(T(n-1) \leftarrow T' \leftarrow \text{Huffman}(S')\)

\(S_T \leftarrow \text{tree created from } T' \text{ by} \)
Example:

```
   a   b   c   d   e   f   g
0.35 0.22 0.18 0.10 0.08 0.04 0.02
```

```
   a   b   c   d   e
0.35 0.22 0.18 0.1 0.08
```

```
   a   b   c   d
0.35 0.22 0.18 0.1
```

```
   a   b   c
0.35 0.22 0.18
```

```
a   bc
0.35 0.40
```

```
   d   e   f   g
```

```
   defg
```

```
   bc
```

---

Running time: \( |S| = n \)

\[ T(n) = T(n-1) + O(n) = O(n^2) \]

Can we do better?

\[ T(n) = T(n-1) + O(\sqrt{n}) = O(n^{\sqrt{2}}) \]
Correctness: We will prove that Huffman yields an optimal binary tree using induction on $S$.

**IH**: Let $k > 0$ be an integer. Assume that when Huffman is called on $S$ and $|S| = k$ then the tree returned is optimal.

**BC**: $|S| = 1, |S| = 2$ ✓

**IS**: We want to prove that Huffman yields an optimal when $|S| = k+1$. That is, we want to prove that the tree $T$ is
Lemma: Let $F$ be a full binary tree in which the two lowest leaf letters $y^*$ and $z^*$ are associated with the two sibling leaves at the lowest level. Let $F'$ be the tree obtained from $F$ by removing $y^*$ and $z^*$. Then we have

$$\text{ABL}(F) = \text{ABL}(F') + f_y y^* + f_z z^*,$$

i.e.,

$$\text{ABL}(F) - \text{ABL}(F') = f_y y^* + f_z z^*.$$

Proof sketch

$F$: [Diagram of a full binary tree]

$F'$: [Diagram of a full binary tree with $y^*$ and $z^*$ removed]
\[
\begin{align*}
&= \text{depth}_{f}(\omega) + \text{depth}_{P}(\omega) \\
&\leq \text{depth}_{f}(\omega) + \text{depth}_{P}(\omega) + 1 \\
&= \text{depth}_{f}(\omega) + \text{depth}_{P}(\omega) + 1 \\
&\leq \text{depth}_{f}(\omega) + \text{depth}_{P}(\omega) + 1 \\
&\leq \text{depth}_{f}(\omega) + \text{depth}_{P}(\omega) + 1.
\end{align*}
\]

**IS (Cont'd)** Assume for contradiction that

\( T \) is not optimal, but \( Z \) is. Furthermore, among all optimal binary trees, \( Z \) is a tree in which \( y^* \) is an sibling at the lowest level. We know such an opt' tree exists (by a previously proved lemma). Let \( T' \) &
$T'$ be the trees obtained from $T$ and $T$, resp., by removing $y^*$ & $z^*$ from them trees. By the prior lemma,

$$\text{ABL}(T') = \text{ABL}(T) + f/\lambda.$$  

$$\text{ABL}(T) = \text{ABL}(T') + f/\lambda.$$  

$$+ve \quad = \quad \leq 0$$

**Contradiction!**
Graph Algorithms.

Q: Given an undirected graph $G=(V,E)$ and vertices $s \in V$ and $t \in V$, is there a path from $s$ to $t$ in $G$?

Consider the following algorithm that finds all the vertices reachable from $s$ in $G$. 
\[ R \leftarrow \{ \emptyset \} \]

**while** there is an edge \( e = (u,v) \) that crosses the cut \((R, V \setminus R)\) do

- **partition of vertex** \[ R \leftarrow R \cup \{ v \} \]

**return** \( R \)

**Correctness**

**Lemma**: \( v \in R \) if and only if \( v \) belongs to the connected component containing \( u \) in \( G \).

\( (\Rightarrow) \) \( v \in R \) then \( v \) belongs to the cc
Proof: Let $v$ be the first vertex that is brought into $R$ that is not in the cc containing $s$. Who brought $v$ into $R$? Vertex $u$, say. Clearly $u \in$ cc containing $s$, since $v$ was the first vertex that did not belong to cc containing $s$. Then there is a path from $s$ to $v$ given by $s \xrightarrow{} u \xrightarrow{} v$, contradicting that $v \notin$ cc containing $s$.

($\Leftarrow$) $v \in$ cc containing $s$ $\Rightarrow$ $v \in R$.

Since $v \in$ cc containing $s$, by def, there must be a path, say $P$ from $s$ to $v$ in $G$. 

Going from $s$ to $v$, let $y$ be the first vertex that does not belong to $R$. Let $x$ be the vertex preceding $y$ in $P$.

Clearly $x \in R$. Since $x \in R$ and $y \notin R$, the edge $(x, y)$ must be crossing the cut $(R, V \setminus R)$, contradicting that the algorithm ended.

BFS (Breadth First Search)

DFS (Depth First Search)
BFS

DPS