- Huffman Tree

- Encoding Problem

- Given string $S$ with $n$ different characters, length $m$
  Find a way to encode $S$ as a binary string $C$.
  Minimize the length of $C$.

- Fixed length encoding

  Can use $\lceil \log_2 n \rceil$ bits for every character.

  Example: DNA sequence $A = 00$, $T = 01$, $C = 10$, $G = 11$

  $$ATCCAC \rightarrow 00 \ 01 \ 10 \ 00 \ 11$$

  $$m = 6$$

  $$A \ T \ C \ C \ A \ G$$

  Length $= m \cdot \lceil \log_2 n \rceil$

- Variable length encoding

  Idea: use a shorter length code for more frequent characters.

  $$ABAAAAACA$$

  Encoding:

  $$A = 0 \ B = 10 \ C = 11$$

  $$0 \ 10 \ 00 \ 00 \ 110$$

  $$A \ B \ A \ A \ A \ A \ C \ A$$

  Problem: How to decode?

- Greedy Decoding

  Character $i \rightarrow a_i$:

  $$\begin{cases} a_1 = 0 \\ a_2 = 10 \\ a_3 = 11 \end{cases}$$

  Repeat

  - Find shortest prefix of $C$ such that $C[1..i] = a_i$

  Output $i$, remove $C[1..i]$.

  - Want: a mapping $i \rightarrow a_i$, so that greedy decoding can always correctly decode.

  - "Prefix free": mapping $i \rightarrow a_i$ is prefix-free

    if for any $i$, $a_i$ is not a prefix of $a_j$.
\[ a_1 = 0 \quad \text{prefix-free} \quad a_1 = 1 \quad \text{not prefix-free} \]
\[ a_2 = 10 \quad a_2 = 10 \quad \text{free} \quad a_3 = 01 \]

Theorem: Greedy decoding works if and only if mapping is prefix-free.

Proof: If mapping is not prefix-free, \( \exists a_i, a_j \)

- \( a_i \) is prefix of \( a_j \)
- Look at \( \text{enc}(\#j) \), when you run greedy decoding
  - will output \( \#i \), incorrect
  - if \( c = \text{enc}(s) \), \( s[c] = \#i \) (\( \text{en}(a_i) = t \))

Will prove: first output is \( \#i \)

\[ c[1..t] = a_i \]
\[ \forall t' < t \quad c[1..t'] = \text{prefix of } a_i \neq a_j \quad (\text{prefix-free}) \]

Greedy decoding will output \( \#i \), correct.

- Huffman tree: abstraction for prefix-free encoding
  - Huffman tree is a rooted binary tree
  - Each node has \( \leq 2 \) children, labeled by \( 0, 1 \)
  - Each leaf is labeled by a character
  - Encoding of a character = path from root to leaf.

Claim: prefix-free mapping \( \iff \) Huffman tree

- Cost of Huffman Tree

\[
\omega(T) = \sum_{i=1}^{n} d(\#i) \times f(\#i) = \text{total length of encoding.}
\]

Goal: minimize \( \omega(T) \)

Input: \( n, f(\#i) \) for \( i = 1, 2, \ldots, n \)
Repeat

find two characters #i, #j with smallest \( f(#i), f(#j) \)
merge them to a new character #new
\[ f(#\text{new}) = f(#i) + f(#j) \]

Character: A B C D E
Frequency: 12 10 5 1 3

- Theorem: Huffman tree algorithm always outputs a tree with smallest cost.

Observation

\[ w(T) = w(T') + \frac{f(#i) + f(#j)}{d(#\text{new}) + 1} \]

\[ w(T) = \sum_{l \neq #i, #j} f(#l) \times d(#l) + f(#i) \times d(#i) + f(#j) \times d(#j) \]

\[ w(T') = \sum_{l \neq #\text{new}} f(#l) \times d(#l) + f(#\text{new}) \times d(#\text{new}) \]

\[ f(#i) + f(#j) \]

Cost of merging = \( f(#i) + f(#j) \)

Claim: \( w(T) = \sum \) cost of merging at this step

Proof (for optimality of Huffman Tree):

Do induction on \( n \) (\( n \) characters)
When \( n = 1 \) there is only one tree.

Suppose algorithm is optimal for \( N \leq K \)
Consider a case when \( n = K + 1 \)
Consider a case when $n = k + 1$

Suppose in the first step $\text{ALG}$ merged $\#i$, $\#j$.

Assume $\text{OPT}$ is different: $\#i$, $\#j$ have different parents in $\text{OPT}$.

If in $\text{OPT}$ $d(\#i) \geq d(\#j)$

Swap $\#j$ with $\#i'$.

New cost = old cost $- d(\#i')x f(\#i') - d(\#j)x f(\#j)$

$+ d(\#i')x f(\#j) + d(\#j)x f(\#i')$

$= \text{old cost} - (f(\#i') - f(\#j)) \times (d(\#i') - d(\#j))$

$\geq 0$ $\geq 0$

(If $d(\#i) < d(\#j)$, swap $\#i$ with $\#j'$).

$\Rightarrow$ There is an $\text{OPT}$ with $\#i$, $\#j$ share the same parent.

$\text{ALG}$ outputs optimal tree for all other characters + $\#n$ new.

$T_{\text{alg}}$: algorithm’s tree

$T_{\text{alg}}'$: $\text{ALG}$’s tree with $\#i$, $\#j$ merged

$T_{\text{opt}}$: $\text{OPT}$’s tree

$T_{\text{opt}}'$: $\text{OPT}$’s tree with $\#i$, $\#j$ merged.

$w(T_{\text{alg}}') = w(T_{\text{opt}})$ (induction hypothesis)

$w(T_{\text{alg}}) = w(T_{\text{alg}}') + f(\#i) + f(\#j)$

$w(T_{\text{opt}}) = w(T_{\text{opt}}') + f(\#i) + f(\#j)$

$w(T_{\text{alg}}) = w(T_{\text{opt}})$