- Huffman Tree
- Encoding Problem
  - Given string S with n different character, 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 $A=00$ $T=01$ $C=10$ $G=11$

    $ATCCAG \longrightarrow 00\ 01\ 10\ 10\ 00\ 11$
    $A\ T\ C\ C\ A\ G$
    $m=6$
    length $= m \cdot \lceil \log_2 n \rceil$
  - Variable length encoding
    idea: use a shorter length code for more frequent character.

    $ABAAAACA$
    enc: $A=0$ $B=10$ $C=11$
    $0\ 10\ 00\ 00\ 11\ 0$
    $A\ B\ A\ A\ A\ A\ C\ A$

    problem: How to decode?
  - Greedy Decoding character $\#i \longrightarrow a_i$ $\begin{pmatrix} a_1=0 \\ a_2=10 \\ a_3=11 \end{pmatrix}$
    repeat
    find shortest prefix of C such that $C[1..t]=a_i$
    output i, remove $C[1..t]$
  - want: a mapping $\#i \longrightarrow a_i$, so that greedy decoding can always correctly decode.
  - "Prefix free": mapping $\#i \longrightarrow a_i$ is prefix free
    if for any i, j a_i is not a prefix of a_j

$a_1 = 0$
$a_2 = 10$
$a_3 = 11$
prefix free

$a_1 = 1$
$a_2 = 10$
$a_3 = 01$
not prefix free

- 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 $enc(\#j)$, when you run greedy decoding will output $\#i$, incorrect.

if $C = enc(S)$, $S[i] = \#i$ $\quad len(a_i) = t$

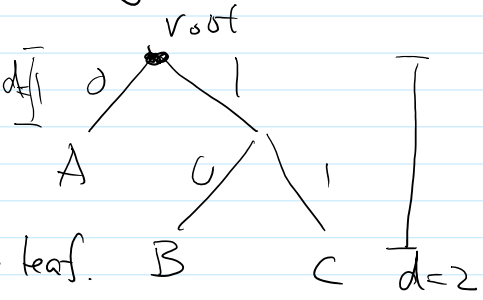
will prove: first output is $\#i$

$C[1..t] = a_i$

$\forall t' < t \quad C[1..t'] =$ prefix of $a_i \neq a_j$ (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 ≤ 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 $\Longleftrightarrow$ Huffman tree

- Cost of Huffman Tree

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

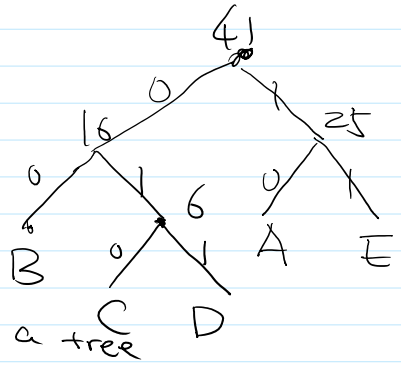
$d(\#i)$: depth of character $\#i$; $f(\#i)$: frequency of character $\#i$

Goal: minimize $w(T)$

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

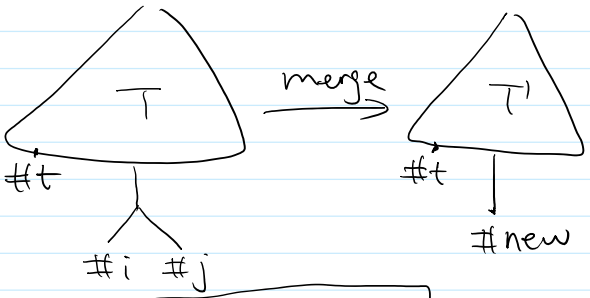
repeat
find two characters #i, #j with smallest f(#i), f(#j)
merge them to a new character #new
$f(\#new) = f(\#i) + f(\#j)$

character	A	B	C	D	E
frequency	12	10	5	1	13



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

Observation



$$w(T) = w(T') + \boxed{f(\#i) + f(\#j)}$$

$d(\#new)+1$

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

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

$f(\#new) = f(\#i) + f(\#j)$

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

Claim: $w(T) = \sum_{\text{steps}}$ cost of merging at this step

Proof (for optimality of Huffman Tree):

induction:

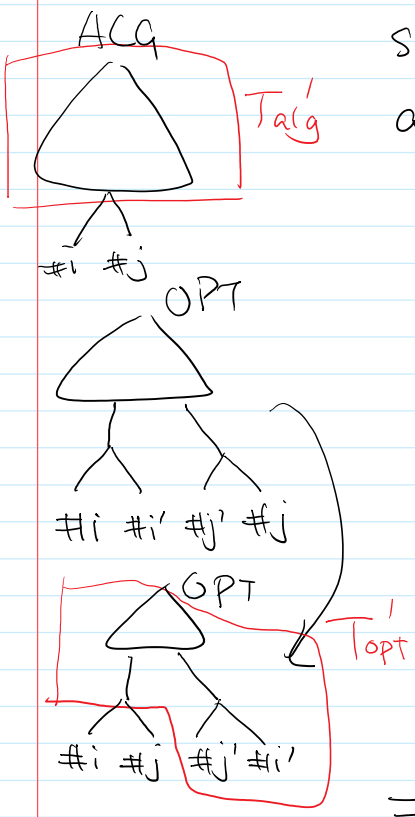
Do induction on n (# characters)

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 ALG merged #i, #j

assume OPT is different: #i, #j have different parents in OPT

if in OPT $d(\#i) \geq d(\#j)$

swap #j with #i'

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

$$= \text{old cost} - \underbrace{\left(f(\#i') - f(\#j)\right)}_{\geq 0} \times \underbrace{\left(d(\#i') - d(\#j)\right)}_{\geq 0}$$

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

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

ALG outputs optimal tree for all other characters + #new

T_{alg} : algorithm's tree

T'_{alg} : alg's tree with #i, #j merged

T_{opt} : OPT's tree

T'_{opt} : OPT's tree with #i, #j merged.

$w(T'_{alg}) = w(T'_{opt})$ (induction hypothesis)

$w(T_{alg}) = w(T'_{alg}) + f(\#i) + f(\#j)$

$w(T_{opt}) = w(T'_{opt}) + f(\#i) + f(\#j)$

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