Lecture 07

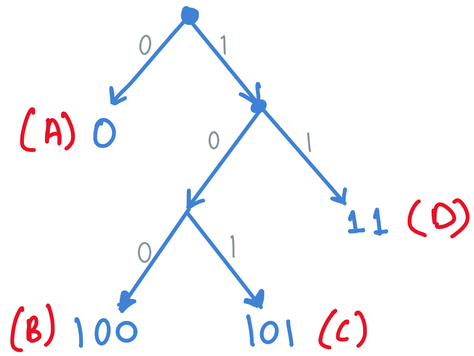
Tuesday, 24 August 2021 9:06 AM

Huffmann Encoding

	FREQ
A	45
B	9
C	11
D	35

→

	CODES
A	0
B	100
C	101
D	11



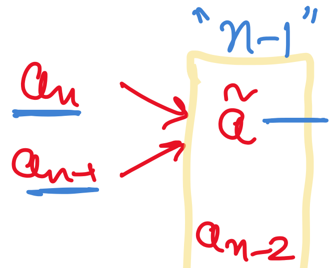
← leaves of Binary tree

Length of $\overset{\text{Encoded}}{\text{msg}}$ = $45(1) + 9(3) + 11(3) + 35(2) = 175$

12% Improvement ↑ $\underbrace{}_{200}$

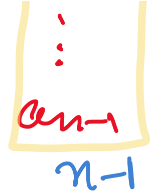
Observation (last class): If $a_1 .. a_n$ satisfy $f_1 \geq f_2 \geq \dots \geq f_n$. Then, there is an opt tree T where a_n, a_{n-1} are siblings.

↑ Prefix encoding



Sketch of: ① Replace two letters with least frequency, i.e. a_n, a_{n-1} by single new symbol $\tilde{a}$.

Algo



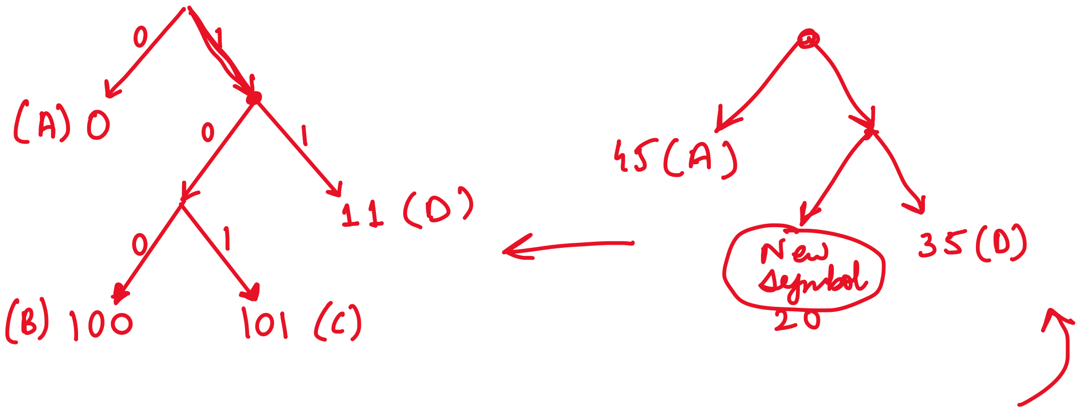
② So, $freq(\tilde{a}) = \tilde{f} := f_n + f_{n-1}$

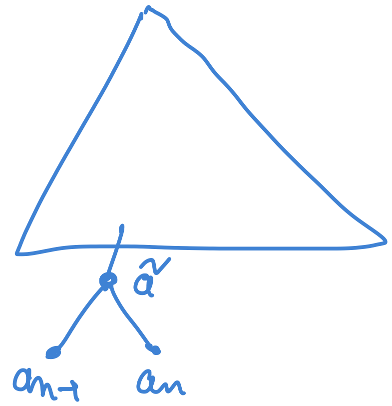
③ Solve $\tilde{F} = (F \cup \tilde{f}) \setminus \{f_{n-1}, f_n\}$, and find tree $\tilde{T}$

eliminating

④ If $\tilde{a}$ is node for $\tilde{f}$, then add children a_{n-1}, a_n to $\tilde{a}$

Huffman.





$F\ (45, 9, 11, 35) \longrightarrow F^* = (45, 20, 35)$

General Theorem: Let $F = (f_1, .., f_n)$ represent frequency of symbols $(a_1 .. a_n)$, and i, j be such that $\exists$ an opt tree in which a_i, a_j are siblings.
Then, problem $F^* = (F \setminus \{f_i, f_j\}) \cup \{f^*\}$ where $f^* = f_i + f_j$
satisfy $opt(F) = f_i + f_j + opt(F^*)$

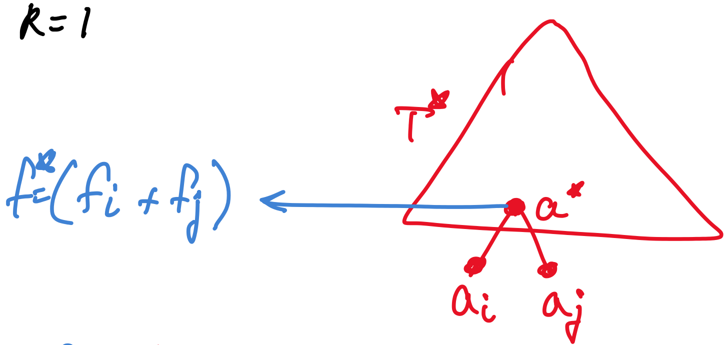
Proof :

(i) $opt(F) \leq opt(F^*) + f_i + f_j$

Let T^* be opt tree of F^*, and let a^* be node with freq $f^* = f_i + f_j$

Create a tree T with n leaf nodes from T^*, by just adding children a_i, a_j to a^*

$$\sum_{k=1}^{n} f_k \cdot depth(a_k, T) = opt(F^*) + f_i + f_j$$

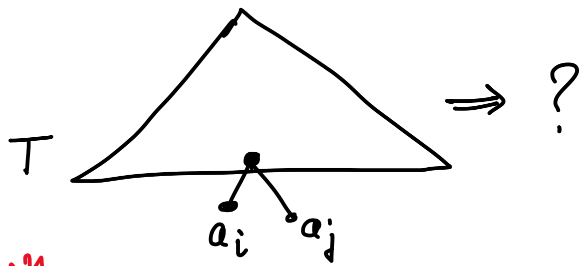


f_i $depth(a_i) = 1 + depth(a^*)$
f_j $depth(a_j) = 1 + depth(a^*)$

$\Rightarrow opt(F) \leq sol^n\text{-}size(T) = opt(F^*) + f_i + f_j$

(ii) $opt(F^*) \leq opt(F) - f_i - f_j$

Use Fact : $\exists$ an opt tree T for F in which a_i, a_j are siblings



H.W. Sol^n

Let a^* be parent of $a_i (a_j)$ and $\tilde{T}$ be tree obtained from T by removing a_i, a_j.

Then $\tilde{T}$ satisfy,

$$opt(F^*) \leq \sum_{\ell \in LEAVES(\tilde{T})} FREQ(\ell) * depth(\ell, \tilde{T})$$

$$= opt(F) - (f_i + f_j)$$

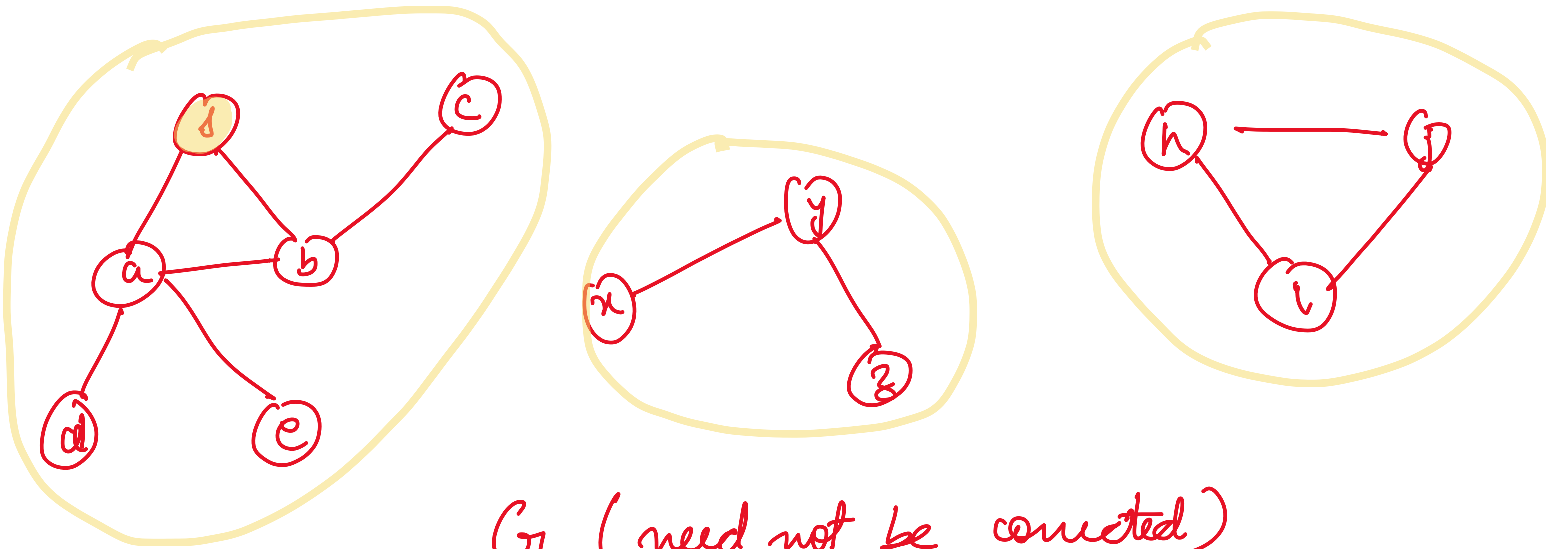
$\Rightarrow opt(F^*) \leq opt(F) - f_i - f_j.$

⊗ BFS / DFS / shortest-path algo in weighted graphs / negative weights

BFS — shortest path tree

DFS — connectivity

weighted — Dijkstra's algo.



6 vertices

G (need not be connected)

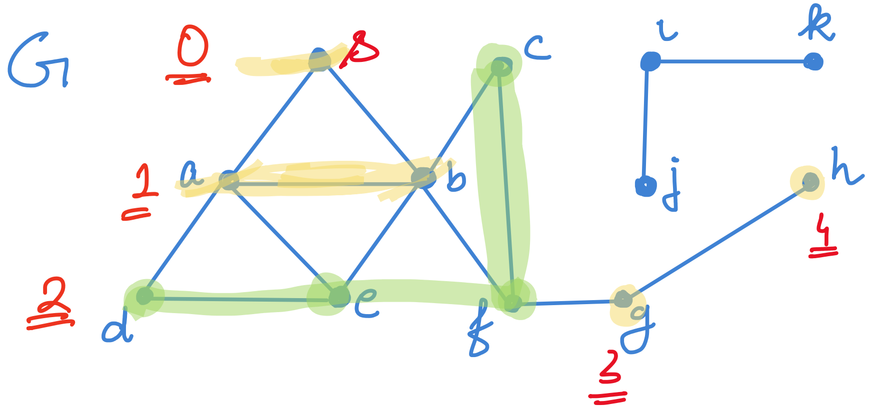
$L_0 = \{s\}$

L_1 = All neighbors of s.

$L_2 = \{ x \mid x$ is a neighbor of a vertex in L_1, but $x \notin L_0 \cup L$

$\vdots$

$L_i = \{ x \mid x$ is a neighbor of a vertex in L_{i-1}, but $x \notin L_0, L_1,$



$L_0 = \{s\}$

$L_1 = \{a, b\}$

$L_2 = \{c, d, e, f\}$

$L_3 = \{g\}$

$L_4 = \{h\}$

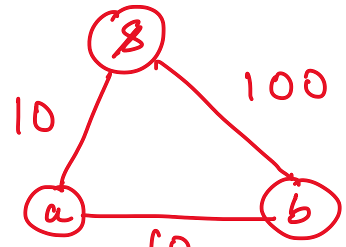
<u>Ques</u>: Can BFS tree diff

shortest - pa

Claim: If G is unweighted, then

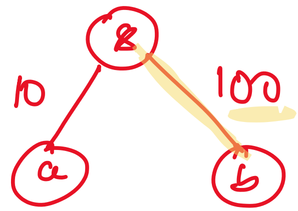
$$L_i = \{ x \mid dist(s, x, G) = i \}$$

Proof: H. W. (by induction).



Shor

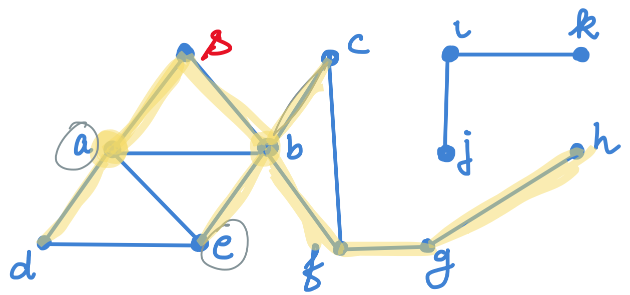
BFS tree



dist

Example:

G



$V = S_1$

Question:

Can you find BFS tree using

S_1, S_2 — two stacks

flag = visited / unvisited

$S_1 = \boxed{s}$ (0^{th} layer)

$V = \{1, \ldots, n\}$

⊛ stacks with integer entries is sufficent

code for this? ←

$S_2 = \boxed{a \mid b}$

(2nd layer)

$S_1 = \boxed{c \mid e \mid f \mid d}$

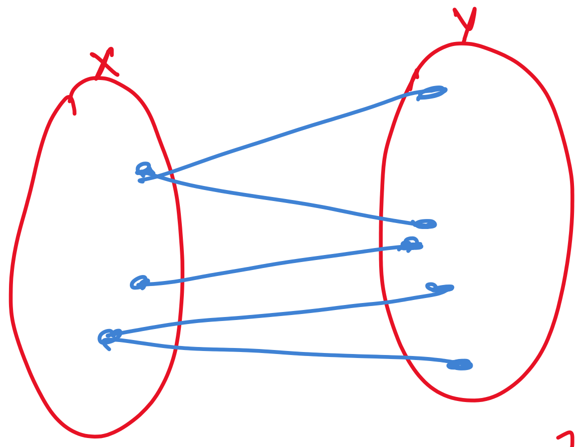
$S_2 = \boxed{g}$

H

⊛ Application of BFS Tree.

Bipartite graph (defn):

An undirected graph $G = (V, E)$ for which there is a partition (X, Y) of V satisfying $E \subseteq X \times Y$.



$V(G) = X \cup Y$

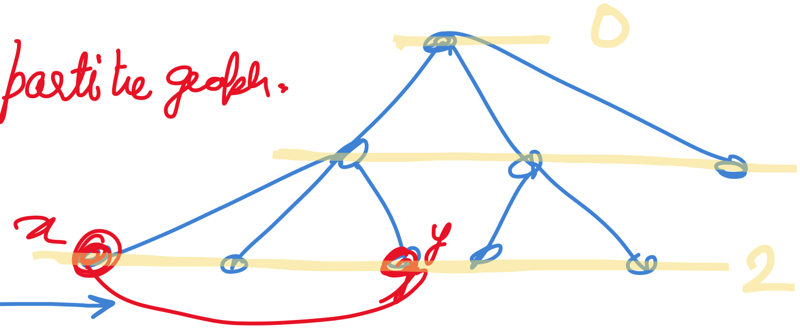
X & Y are di

each edge $(a, b) \in (X) \times (Y)$

Ques 1) If T is a BFS tree of G such that for each edge (x,y), $level(x,T) \neq level(y,$

Then, prove that G is Bipartite graph.

not possible



Ques 2: If G is bipartite, then G has no cycle of OD

Ques 3: Use Q1, Q2 to obtain $O(m+n)$ time algo to if a given graph is bipartite.