S 67

$$A \cdot B = A_1B_1 \times 10^n + (A_1B_2 + B_1A_2)10^{n/2} + A_2B_2$$

Multi(A, B, n)

$$T(n) = 4T\left(\frac{n}{2}\right) + 4n \quad \in O(n^2)$$

A_1B_1
A_2B_2
A_1B_2
B_1A_2

$(A_1 + A_2)(B_1 + B_2) - A_1B_1 - A_1B_2$

$$T(n) = 3T\left(\frac{n}{2}\right) + 6n$$

$$\in O(n^{\log_2^3})$$

Counting inversions in an Array :

(How far array is from sorted state)

inversion → $a_i > a_j$ $\quad i < j$

| 1 | 3 | 2 | 5 | 4 | 6 |

A $\quad n/2$ $\quad$ B $\quad n/2$

| 7 | | | | 5 | | | | |

Sorted $\quad$ Sorted

| | | | | | | |

M $\quad$ n

$i = 1, j = 1$

$K = 1$

M T W T F S S

Page No.:

Date:

inv = 0

if A[i] < B[j]

M[k] = A[i]

i++, k++

otherwise M[k] = B[j]

j++, k++

inv = inv + $\frac{n}{2}$ - i + 1

16/01/25

inversion(A, l, r)

if r - l = 0

then return 0;

else i = inversion (A, l, $\frac{l+r}{2}$)

j = inversion (A, $\frac{l+r}{2}$ + 1, r)

k = cross-inversion (A, l, r)

return i + j + k

Cross - inversion (A, l, r)

a = l

b = $\frac{l+r}{2}$ + 1

c = l

while c <= r

if (A(a) < A(b)

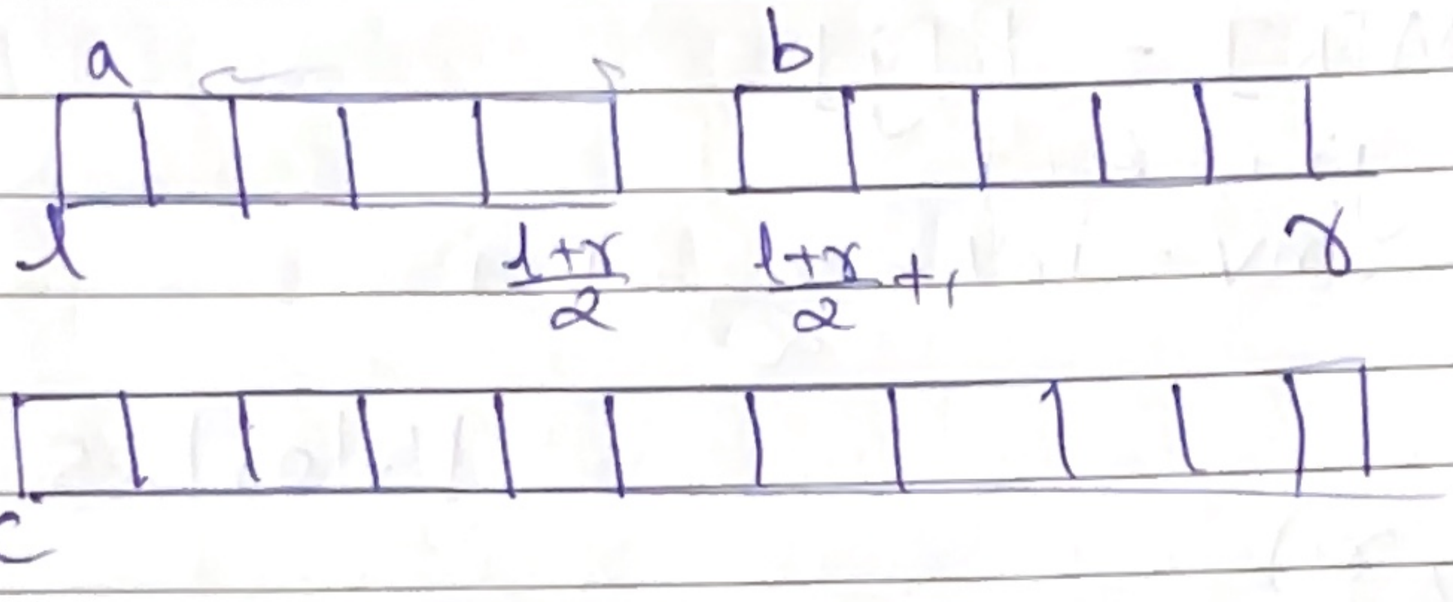
B(c) = A(a)

c++, a++

else: B(c) = A(b)

c++, b++

Cinv = Cinv + $\left(\frac{l+r}{2}\right) - a + 1$



Selection / Order Statistic

Input: An array A, an integer n

Output: K-th smallest element

P ← carefully chosen Pivot

Swap A(l) ⟷ A(P)

(← Partition(A, l, r, K)

If j = k, return A(i)

else if j < k return Selection(A, i+1, r, K-1)

else return Selection(A, l, i-1, k)

M T W T F S S

Page No.:

Date:

Selection(A, k)

1. Group the array into 5-element groups
2. Find the median of each group and store them into an array B.
3. Call selection $(B, \frac{n}{10})$
4. Choose the median element of B as your pivot and run Partition or quick sort. Let j be the position of the pivot.
5. If $j < k$ return Selection(A, j+1, r, k−j)

if $j = k$ return A(k)

if $j > k$ return Selection(A, l, j−1, k)

Time Complexity Analysis

$$T(n) = T\left(\frac{n}{5}\right) + \max\{T(i), T(n-i)\}$$

M T W T F S S

Page No.:

Date: 21/01/25

Greedy Algorithm

Problem: n items $i_1, i_2 \dots i_n$

with $(w_1, v_1), (w_2, v_2) \dots (w_n, v_n)$

Knapsack Capacity (W)

until the bag is full or no item left

Pick the item with maximum $\frac{v}{w}$ value (not yet picked) as much as possible.

Problem: 0-1 Knapsack

Weight = (15)

weights -	6	7	8	9
value -	1	13	15	18

The greedy strategy of choosing most valuable item gives optimal solution.

$i_1, i_2 \dots i_n$ (sort based on $\frac{v_i}{w_i}$)

$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \dots \frac{v_n}{w_n}$

A $x_i \in (0,1]$

$(x_1, x_2 \text{---} x_n)$ $\sum_{i=1}^{n} x_i w_i = W$

O

$(y_1, y_2 \text{---} y_n)$ $\sum_{i=1}^{n} y_i w_i = W$

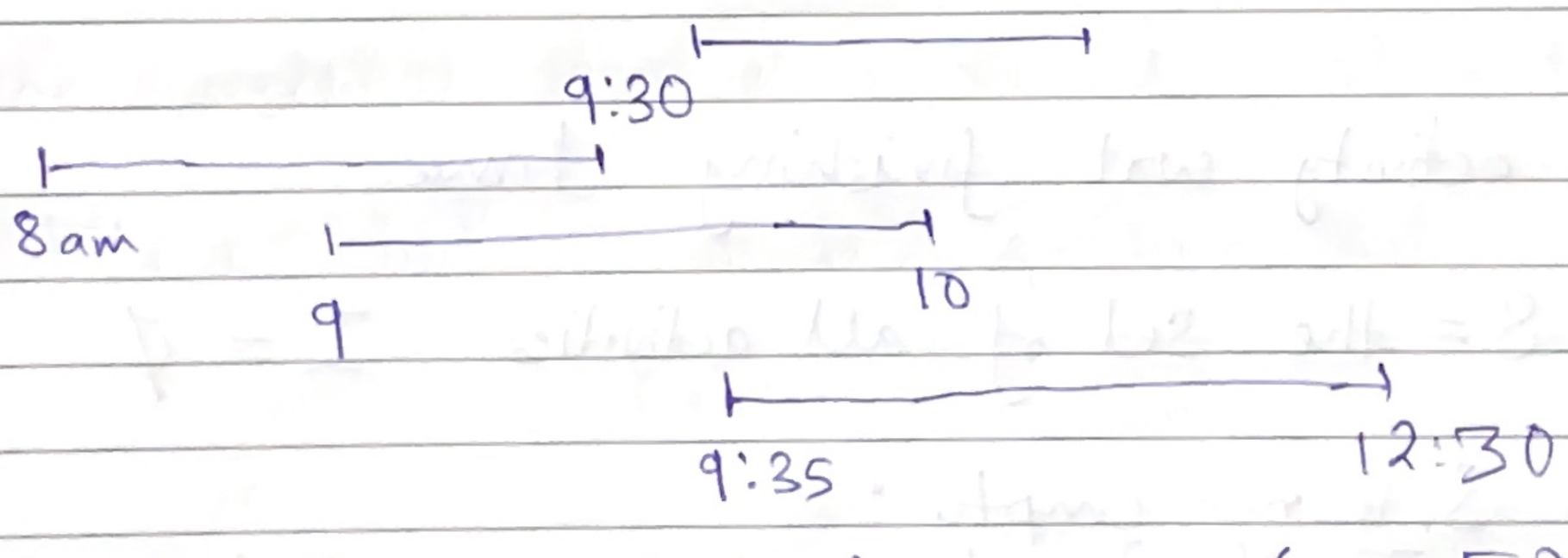
$y_i \in (0,1]$

Let i is the smallest value such that $x_i \neq y_i$

$(y_1', y_2', \text{---} y_n')$

O' $(y_1, y_2 \text{---} y)$

Interval scheduling / Activity scheduling



Input: (S_1, F_1) (S_2, F_2) --- (S_n, F_n)
a_1 a_2 --- a_n

Output: A set $b_1, b_2 \text{---} b_m$ s.t $f_{b_i} \leq S_{b_{i+1}}$

Possible greedy choice

(1) choose the activity that starts earliest
(2) That ends earliest
(3) Smallest length
(4) Minimum overlap

Greedy Strategy: Select the activity with earliest finishing time.

Algo:

1. Sort the activity wrt finishing times.
2. $i = 1$, S = the set of all activities, $I = \phi$
3. While S is non empty:
   $I = I \cup \{a_i\}$ Let m be the minimum index such that $S_m \geq f_i$
   Delete $a_{j+1}, a_{j+2}, \ldots a_{m-1}$ from S
   Set $j = m$

$O = \{O_1, O_2 \dots\}$

$I = \{i_1, i_2 \dots$

$O' = \{i_1, O_2\ O_3 \dots$

27/01/25

Job Scheduling Problem

Input : n Jaws $a_1, a_2 \dots a_n$ with $(l_1, w_1), (l_2 w_2) \dots (l_n w_n)$ where l_i denotes the time to complete job i and w_i is the weight (priority) of Job i.

A schedule P is a Permutation of $(1, 2, \dots n)$ that denote the order in which the jobs are scheduled.

The completion time of a job i according to is $C(i)$

Observe: Find a permutation I such that

$$\sum_{i=1}^{n} C(i) w_i \text{ is minimum}$$

Algorithm

1. Sort the jobs with respect to $\frac{w}{i}$ value

2. Schedule the job in the decreasing order $\frac{w}{i}$ value

3

M T W T F S S

Page No.:

Date:

2		5			O_5
5		4			O_4
4		3			O_3
3		2			O_2
1		1			O_1

28/01/25

Optimal Encoding Problem:

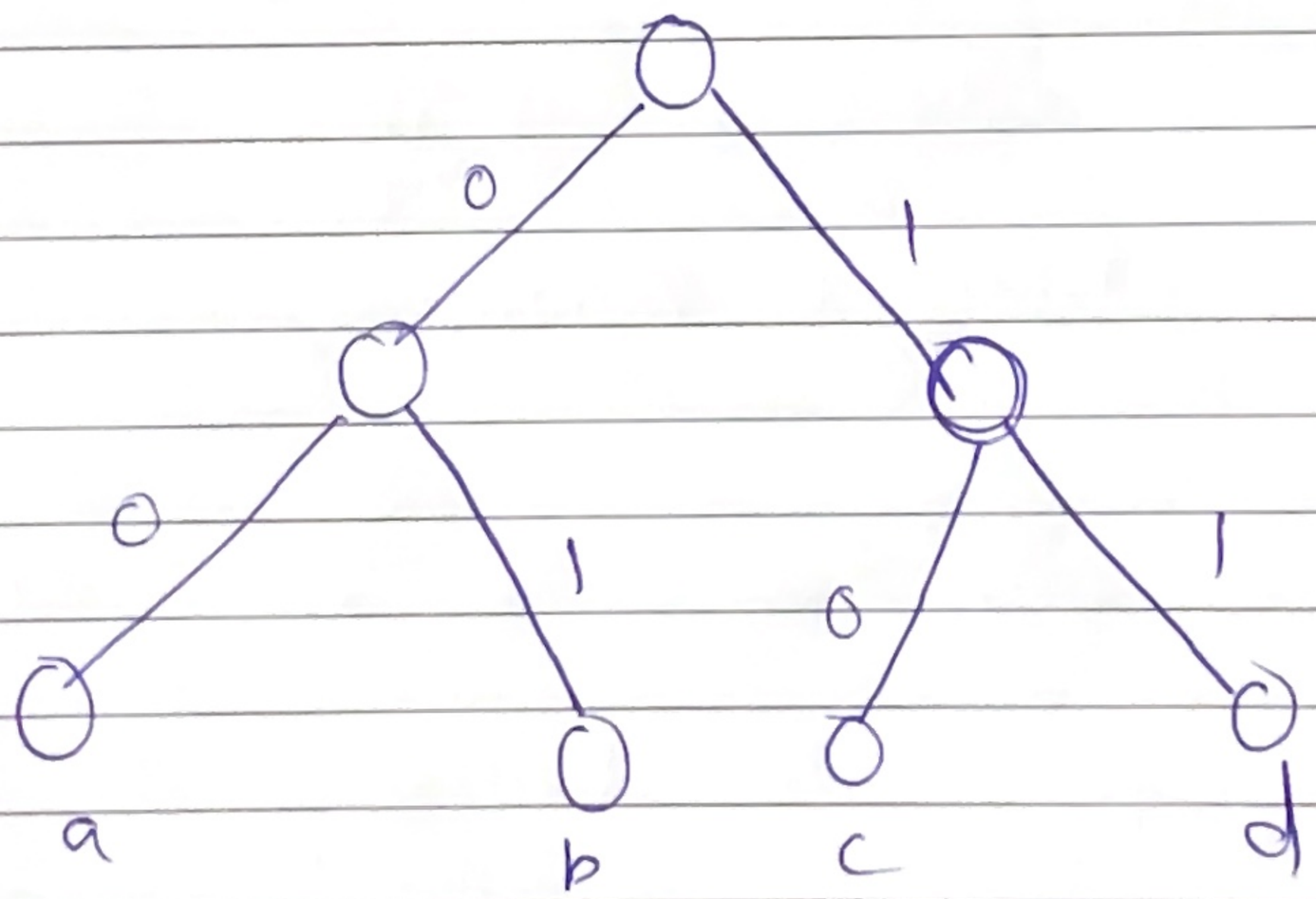
Input: A set of alphabet $A\{a_1, a_2 \text{----}, a_n\}$ with frequency distribution $f(a_1), f(a_2) \text{---} f(a_n)$

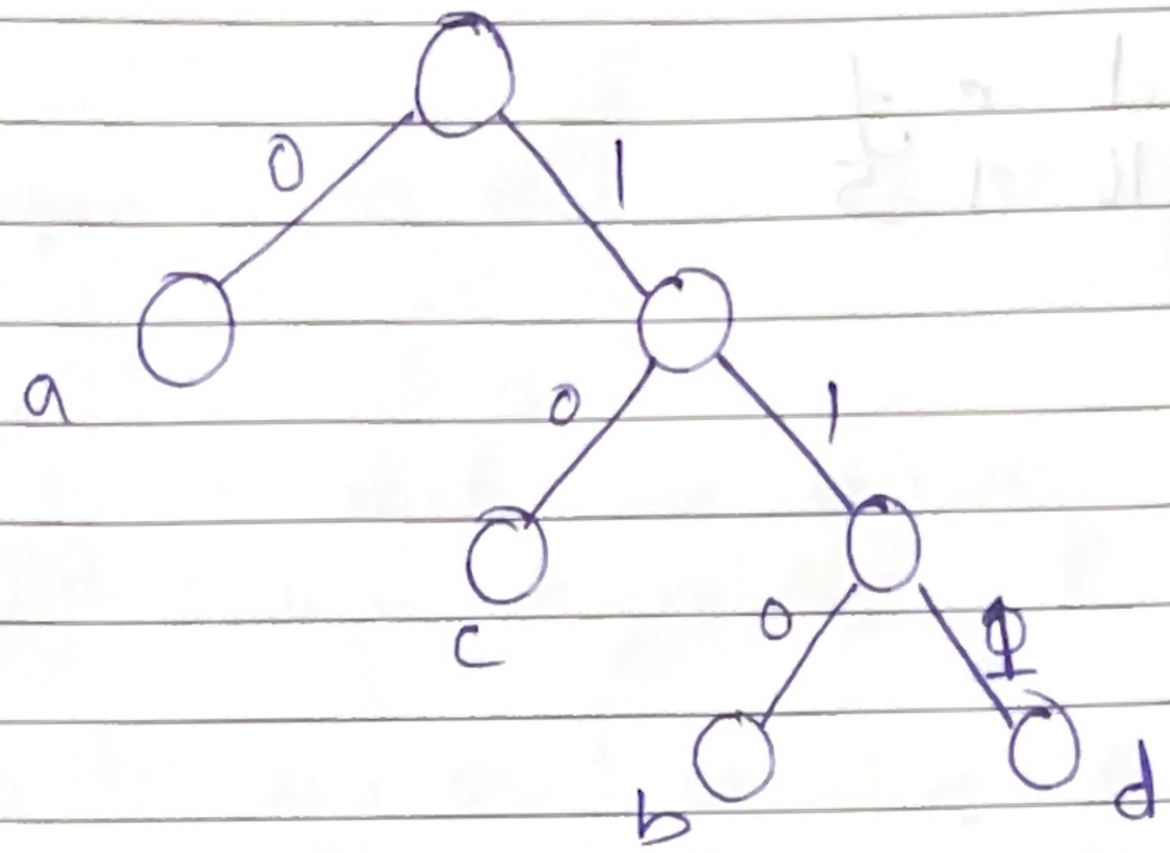
Output: A function $L: A \longrightarrow [0,1]^*$, such that $\sum_{i=1}^{n} f(a_i)\,|L(a_i)|$ is minimized and $L(A)$ must be a set of prefix free codes.

Every prefix free code can be represented in a binary tree.

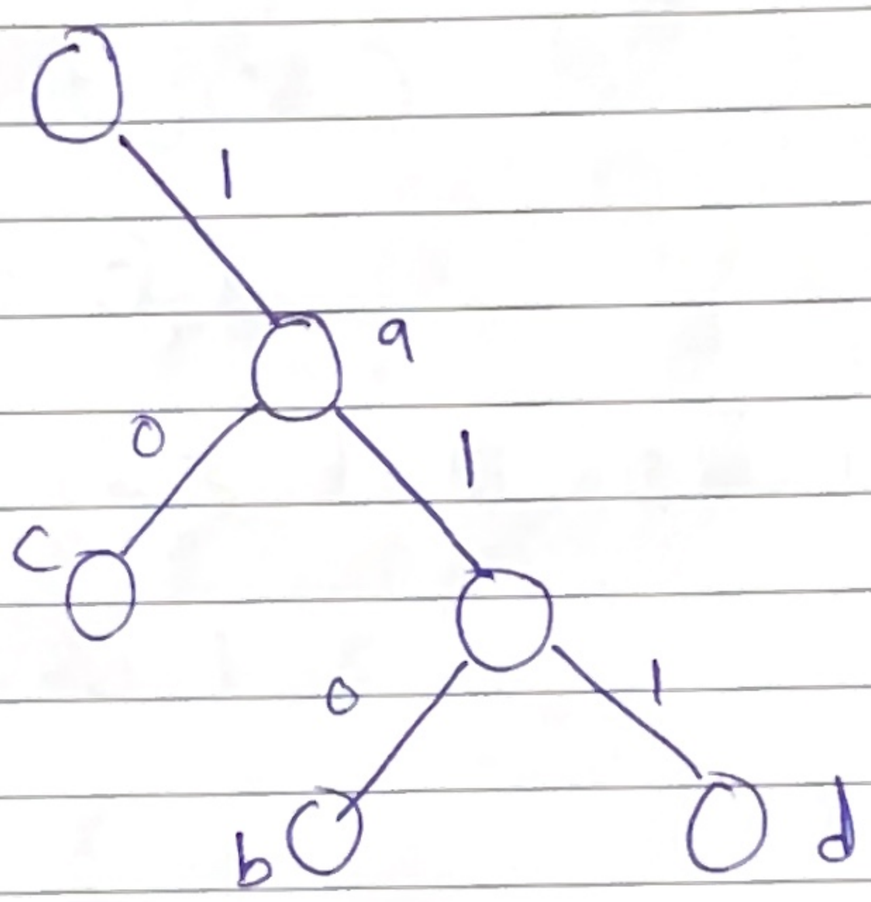
$C_1 = \{a=00,\ b=01,\ c=10,\ d=11\}$

$C_2 = \{a=0,\ b=110,\ c=10,\ d=111\}$





$C_3 = \{a=1,\ b=110,\ c=10,\ d=111\}$



Output : A prefix free T such that

$$\sum_{i=1}^{n} f(a_i)\, d_T(a_i) \text{ is minimum}$$

d_T : depth of the character

M T W T F S S

Page No.:

Date:

Algorithm

Q ← frequency of the character (Priority Queue)

While $|Q| \neq 1$ do

Z_1 = extract min (Q)

Z_2 = extract min (Q)

If nodes are not created then create nodes from Z_1 & Z_2. create another node Z.

$Z \rightarrow left = Z_1$, $Z \rightarrow right = Z_2$,

$f(Z) = f(Z_1) + f(Z_2)$

insert (f(Z), Q)

Eg a = 25, b = 31, c = 3, d = 9, e = 21, f = 1, g = 5, h = 7

{25, 31, 3, 9, 21, 1, 5, 7}.

M T W T F S S

Page No.:

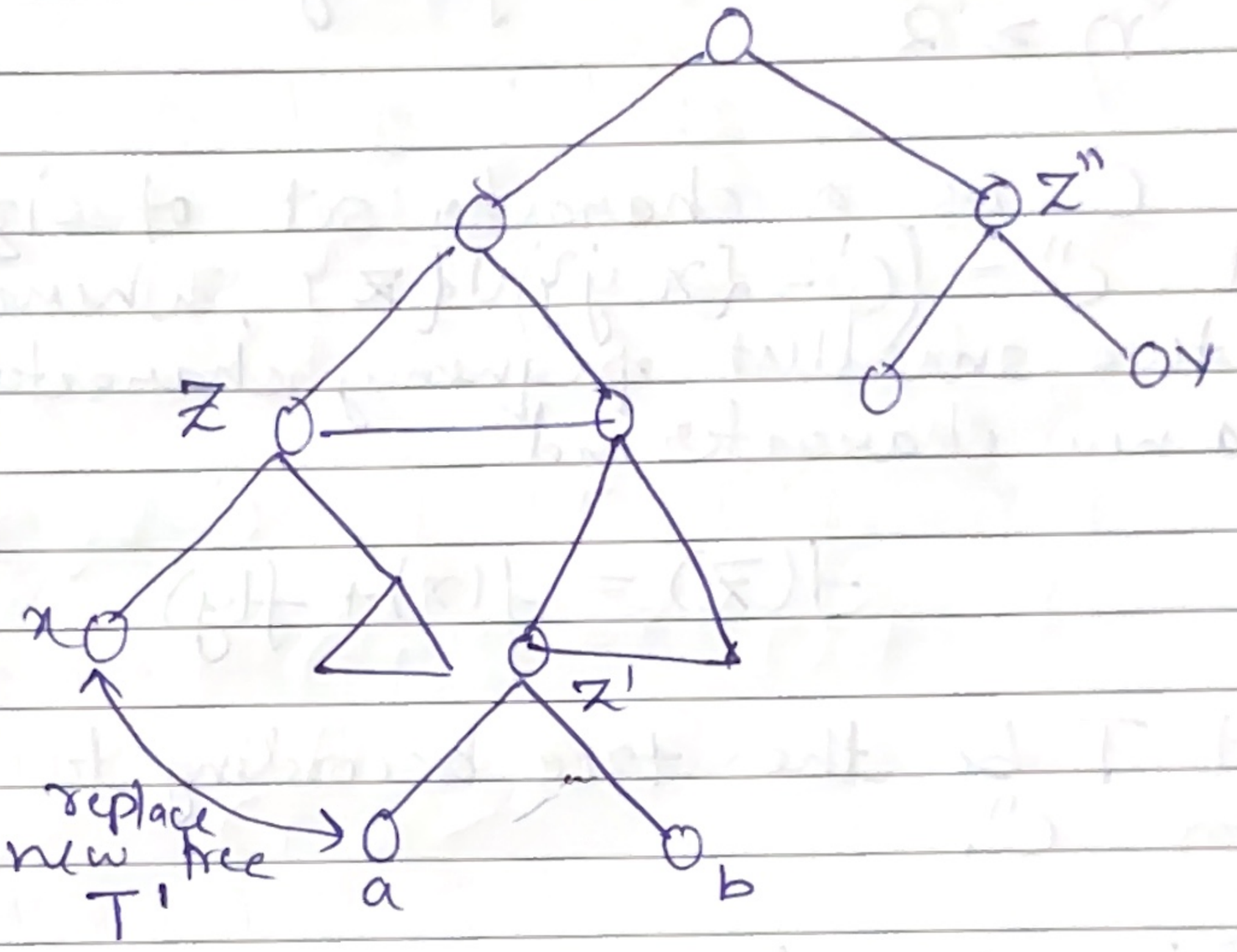
Date:

Observation: Every optimal tree must be full.

Lemma 1: Let x & y be two characters with smallest frequencies then there exists an optimal tree T where x & y are siblings and appears in the lowest depth.

$d_T(x) < d_T(a)$
$d_T(x) < d_T(b)$
$d_T(y) < d_T(a)$
$d_T(y) < d_T(b)$

$f(x) < f(a)$
$f(x) < f(b)$
$f(y) < f(a)$
$f(y) < f(b)$



Theorem: Huffman coding algorithm produces an optimal tree.

$$B(T') = \sum_{a \in C} f(a) \cdot d_{T'}(a)$$

$$= \sum_{a \in C} f(a)\, d_T(a) - f(a)\, d_T(x) - f(a)\, d_T(a) + d_T(a) f(x) + d_T(x) f(a)$$

$$B(T') = B(T) - (f(a) - f(x))(d_T(x) - d_T(a))$$

M T W T F S S

Page No.:

Date:

Proof of theorem

B.C

If $|C| = 2$, Huffman coding algorithm is optimal

Inductive hypothesis: Assume that huffman code algorithm gives optimal tree for any character set of size $n \geq 2$

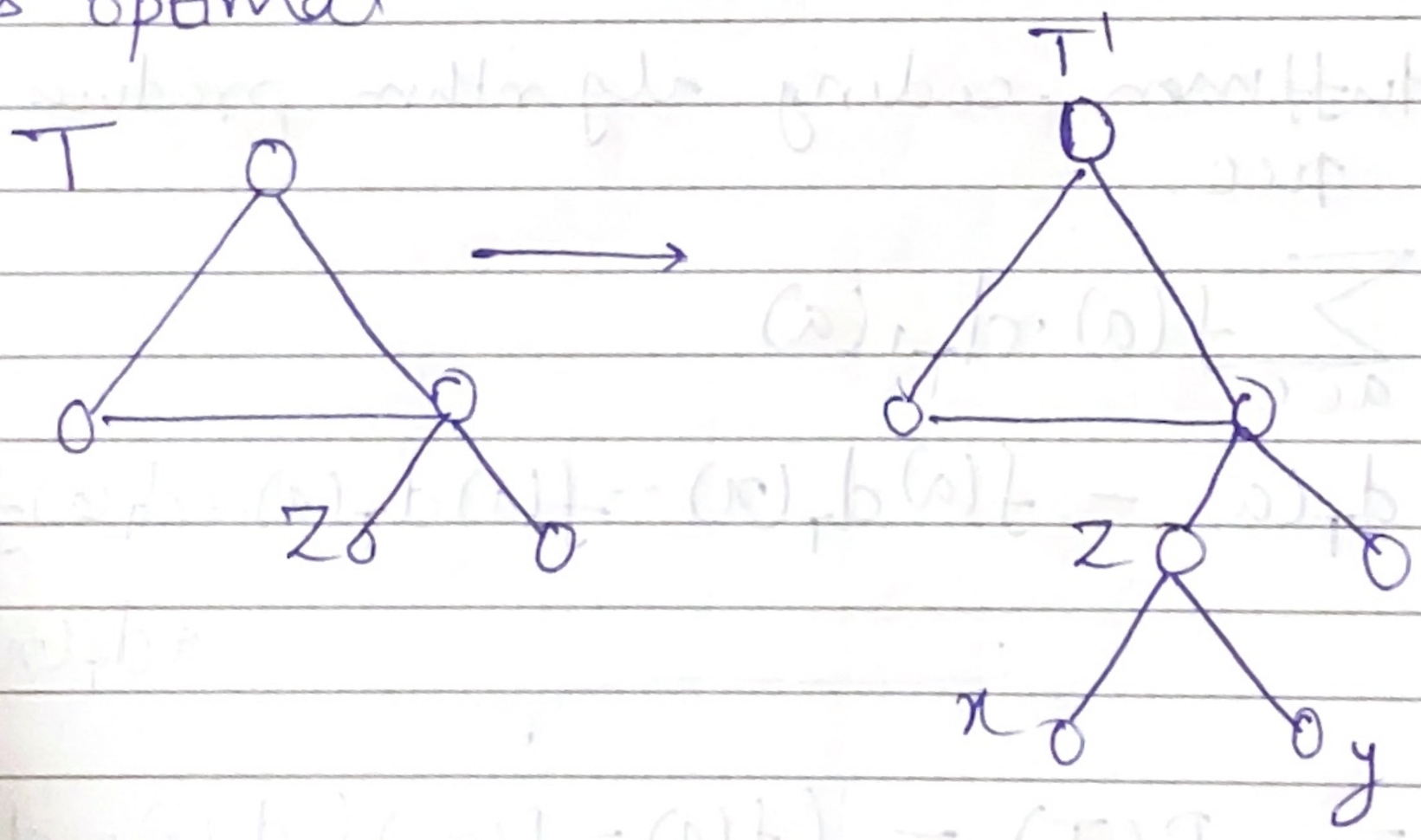
Let C' be a character set of size $n+1$

Let $C'' = (C' - \{x, y\}) \cup \{z\}$, where x & y are the two smallest frequency character in C' & z is a new character and

$$f(z) = f(x) + f(y)$$

Let T be the tree according to Huffman algorithm for C''.

T is optimal

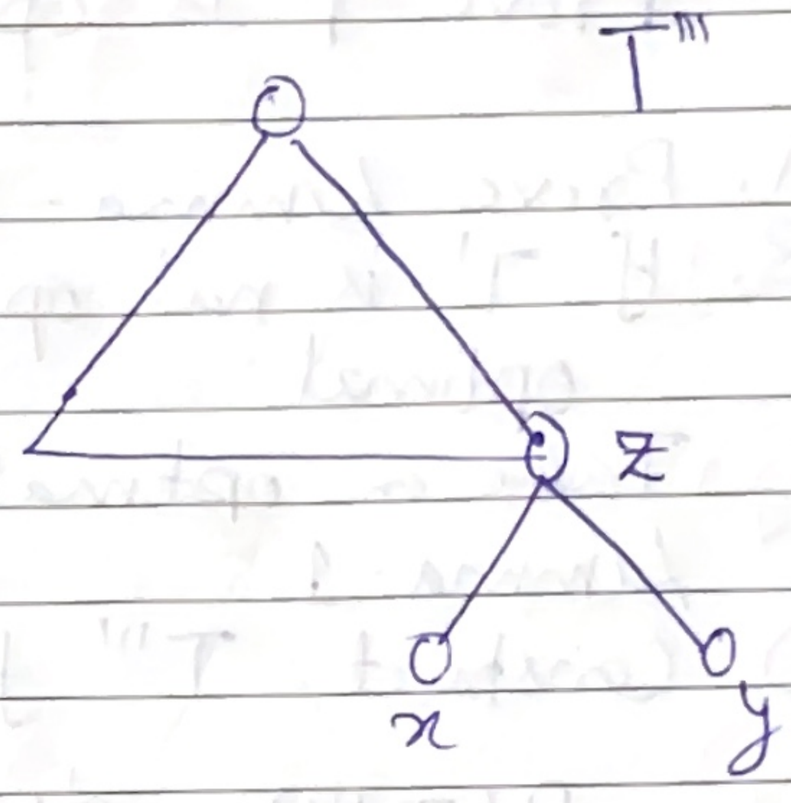


$$B(T') = B(T) - f(z)d_T(z) + f(x)d_T(x) + f(y)d_T(y)$$

$$= B(T) - (f(x)+f(y))\,d_T(z) + (d_T(z)+1)f(x) + (d_T(z)+1)f(y)$$

$$B(T') = B(T) + f(x) + f(y)$$

T''



$$B(T'') < B(T')$$

$$B(T''') = B(T'') - f(x) - f(y)$$

$$< B(T') - f(x) - f(y)$$

$$< B(T)$$

Summary :

Step-1 : Prove for $|C| = 2$, Huff. algo is optimal

Step-2 : Assume for n, prove for $n+1$

A. Take C with $|C| = n+1$

B. Construct $C' = C - \{x, y\} \cup \{z\}$

$f(z) = f(x) + f(y)$

C. Construct T for C' from Huff. algo.

By step 2 T is optimal

D. Construct T' for C by adding x & y as children of (z)

E. $B(T') = B(T) + f(x) + f(y)$

Step-3: Prove T' is optimal

A. Prove Lemma - 1

B. If T' is not optimal then T can not be optimal T''

C. Take an optimal tree^ with the property of Lemma - 1

D. Construct T''' for C' by removing x and y

$$B(T''') = B(T'') - f(x) - f(y)$$
$$\leq B(T') - f(x) - f(y)$$
$$< B(T)$$

M T W T F S S

Page No:

Date: 03/02/25

Dynamic Programming

Fibonacci Sequence :

$$f_n = f_{n+1} + f_{n-2} \qquad f_1 = 1$$

$$f_0 = 0$$

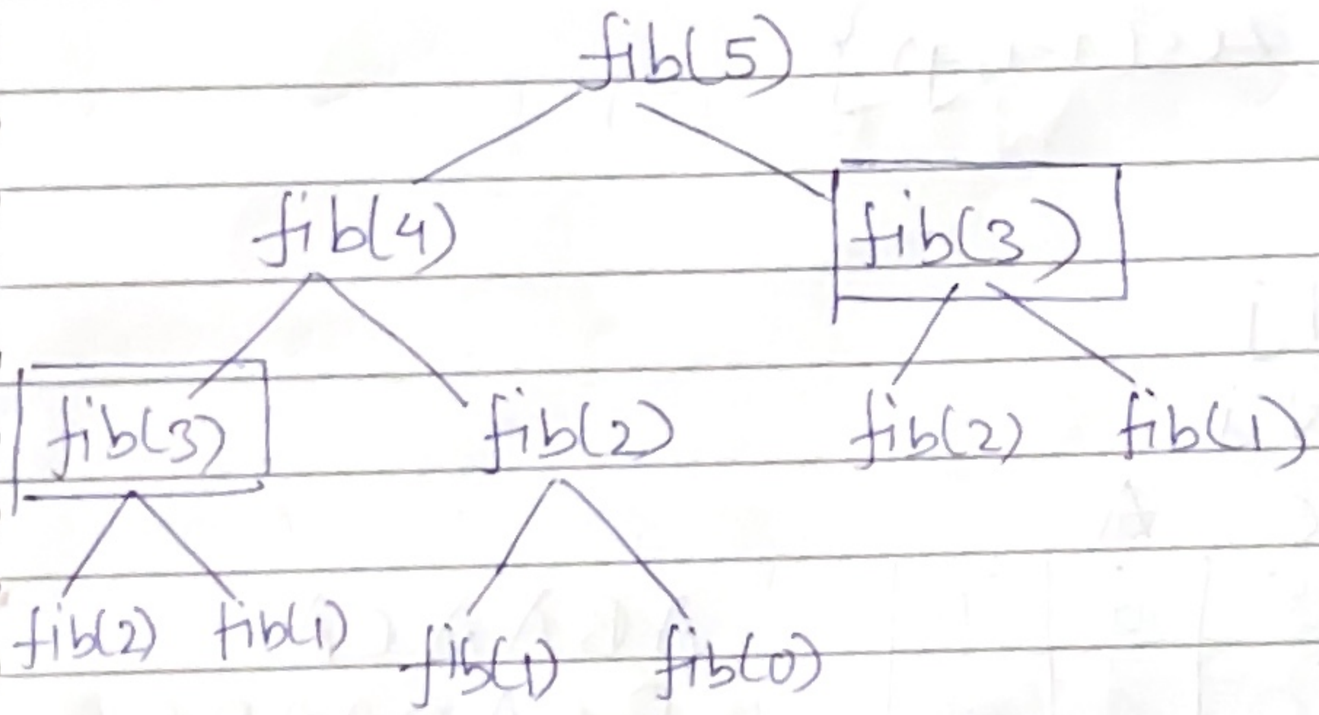
```
fib(n) {
  if (n ≤ 0) return n
  else return fib(n-1)+fib(n-2)
}
```

Recursive method
TC : $O(2^n)$

```
fib(n) { f(0) = 0, f(1) = 1
  for i = 2 to n
      f(i) = f(i-1) + f(i-2)

}
```

D.P.
T.C $O(n)$



Memoization : Storing values which are used again

Optimuss Substructure

Problem ; Longest common Subsequence

Input: Two sequence $X = x_1, x_2 \text{ ---- } x_n$

$Y = y_1, y_2 \text{ ---- } y_n$

eg. ABAACADBC
BCADAABCBC

AAACBC Longest

AAABC

$X = x_1 x_2 \text{----} x_m$

$Y = y_1 y_2 \text{----} y_n$

$LCS(X_m, Y_n) = 1 + LCS(X_{m-1}, Y_{n-1})$ when $x_m = Y_n$

$= \max\{LCS(X_m, Y_{n-1}), LCS(X_{m-1}, Y_n)\}$ when $x_m \neq Y_n$

Sub Problem :

$LCS(i,j)$ = the Largest common subsequence of $x_1 \text{---} x_i$ & X_1 to y_j

$LCS(i,j) = 1 + LCS(i,j)\, LCS(i-1, j-1)$

$= \max\{LCS(i, j-1), LCS(i-1, j)\}$

B.C

$LCS(0, j) = 0 \quad \forall j$

$LCS(i, 0) = 0 \quad \forall i$

		A	B	A	A	C	A
C	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	
B 1	0	0	1	1	1	1	1	
C 2	0	0	1	1	1	2	2	
A 3	0	1	1	2	2	2	3	
D 4	0	1	1	2	2	2	3	
A 5	0	1	1	2	3	3	3	
A 6	0	1	1	2	3	3	4	
B 7	0	1	2	2	3	3	4	
A 8	0	1	2	3	3	3	4	

ABAACA

BCADAABA

```
LCS(X,Y) {
  1. m ← length(X)
  2. n ← length(Y)
  3. for i = 0 to m
        L(i,0) = 0

  4. for j = 0 to n
        L(0,j) = 0

  5. for (i = 1 to m) {
       5.1 for (j = 1 to n) {
             5.11 if xi = yi, then L(i,j) = 1 + L(i-1,j-1)

             5.12 else
                  if L(i-1,j) ≥ L(i,j-1)
                     L(i,j) = L(i-1,j)
                  else L(i,j) = L(i,j-1)
       }
  }
}
```

0-1 Knapsack Problem

04/02/25

n items $a_1, a_2 \dots a_n$
with weights $w_1, w_2 \dots w_n$
and values $v_1, v_2 \dots v_n$
and a given weight W.

find subset S of $\{1,2,\dots n\}$ s.t. $\sum_{j \in S} w_j \leq W$

& $\sum_{j \in S} v_j$ is maximized.

$K(n, w) = K(n-1, w-w) + V_n$ — a_n is picked

$K(n-1, w)$ — a_n is not picked

Define $K(i, w')$ the optimal solution for the set $\{1, \dots i\}$ and w' capacity for the knapsack

then $K(i, w) = \max\{ V_j + K(i-1, w-w_i), K(i-1, w)\}$

Base Case: $K(i, 0) = 0 \quad \forall i = 1, 2 \dots$

$K(0, w) = 0 \quad \forall w = 1, 2, \dots W$

Eg: items a_1 a_2 a_3 $W = 5$

weights 1 2 3

value 12 3 4

	0	1	2	3	4	5
0	0	0	0	0	0	0
1						
2						
3						

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	12	12	12	12	12
2	0	12	12	15	15	15
3	0	12	12	16	16	16

T.C → $O(nW)$

$O(nW \log V_{max})$

$\sum \log V_i + \sum \log W_i + \log W$

$n \log V_{max} + \log W$

Sorting - $\sum \log a_i \leq n \log a_{max}$

$O(n \log n \log a_{max})$

Weighted Interval Scheduling

Input: n activities with

$[s_1, f_1)$, $[s_2, f_2)$ --- $[s_n, f_n)$ with weights

w_1 w_2 --- w_n

Output: A subset S of $\{1, \ldots, n\}$ such that for any two $a_i, a_j \in S$ $a_i \cap a_j = \phi$ and $\sum_{j \in S} w_j$ is maximized

$L(n)$ = the optimal sol^n with activities $[a_1 \ldots a_n]$

$L(n) = L(n-1)$ if $a_n \notin S$

$= w_n + \underbrace{L(t)}_{L(p(n))}$