6/8/25

DAAA

Time Complexity

Page
Date

→ upper Bound (Max time taken by an algo)

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \dots < n^n$

E.g. $f(n) = 12n + 10 \quad n \geq 1$

$\leq 12n + 10 \leq$ → Max value [Bigoh(O)]

E.g. $30n, 30n^2, 30n^4 \dots 100n^n$

Best Case $30n \rightarrow O(n) \rightarrow$ nearest value

Worst Case $\leq n^n \rightarrow O(n^n) \rightarrow$ farthest value

→ lower Bound

Best Case $O(n)$

Worst Case $O(1)$

Omega(Ω) lower Bound → $1 < \log n < \sqrt{n}$

Bigoh(O) upper Bound → $n\log n < n^2 < n^3 < \dots < n^n$

~~Average~~

E.g. Factorial $(n!)$

$n! = 1 \times 2 \times 3 \times \dots \times (n-1) \times n$

$1 \times 1 \dots 1 \times 1 \leq n! \leq n^n \quad n \times n \times n \dots n$

$n! = O(n^n) \quad 1 \longrightarrow n^n$

Tight Constant for testing purpose

Best, Worst, Average.

small oh(o) → Tight upper Bound (not equal)

Note: will not provide guarantee that given problem is best

Greedy Approach

→ finding Best solution in current situation

Optimal sub-structure :-

→ divide the problem into sub problem and find best solution of each problem using greedy approach.

(c) Greedy choice property :- Here we are considering which choice we should make and that choice is selected by looking the best possible sol. for the current problem, without considering the result from other sub-problem.

(d) optimal sub-structure :- A problem exihibits OSS if an optimal sol. to the problem contains within the optimal sol. to sub-problems.

Element of greedy Strategy :-

(1) determine the optimal sub-structure of the problem.

(2) develop a sol. to the problem.

(3) apply greedy choice prop: and prove that it is safe to make greedy choice.

Activity selection problem

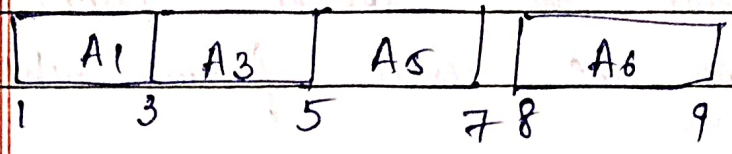
→ select activity so that resource utilize maximally.

(1) Sort the activities → Based on finish time

Activities are A1 ⟶ A7

	A1	A2	A3	A4	A5	A6	A7
start time	1	2	3	0	5	8	5
finish time	3	4	5	6	7	9	9

(2) Select activity:



Here no sorting so O(n)
if there is sorting. Sorting + O(n)
↓
for each activity got checked

6/8/25 WT

HTTPs (443 port)

→ file Transfer

Client ⟶ Server.

FTP

AWS
Azure
Godaddy

Vercel
github pages

Server To show data publically. Transfer the data to server.

DNS → provide domain name like eg: abc.com

→ should be accomodate with Serve.

domain name ⟶ (using nameserver, add to) server → Name Server

DNSchecker.org

```
.C .R1 {
  width = 1170
  Height = 100px
}

.C .R1 .C1 {
  width = 370px;
  Height = 100px;
  float = left;
}

.C .R1 .C2 {
  width = 800 px;
  Height = 100 px;
  float = float-left;
}
```

To tell the duction of Column / ways.

18/8/25

Matroid :

A Matroid A Mathematical structure that generalized the Concept of independence. It provides conditions under which greedy algo. always finds an optimal Sol.

$M = (S, I)$

S : finite set

I : Non-Empty family of subsets of S which are called Independent subset of S, such that if $B \in I$ and A is subset of B then A must Belong to I

Then I is said to be Hereditary if it satisfies this property.

If $A \in I$ and $B \in I$ and $|A| < |B|$ then there exist an element x which belongs to $x \in B - A$ such that $A \cup \{x\} \in I$

This says that Matroid satisfies Independent, Hereditary → Exchange property.

(1) Independent

$S = \{1, 2, 3\}$

$P(S) = \{\phi, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$

set of I / matroid

$\{3\}$ is necessary a member of I

(2)

$B \in I$	$B = \{2, 3\}$	Hereditary	no. of possibilities to add vector
$A \subseteq B$	$A = \{2\}$		
$\therefore A \in I$			

(3) Exchange

$B = \{1, 3\}$ $A = \{3\}$ $|A| < |B|$

$B - A = \{1\}$ $x = \{1\}$

$A \cup \{x\} = \{1, 3\} \in I$

Exchange smaller vector to bigger one as all vector all independent.

Real life / Simple Example / Problem

How / activity problem can be solved with Matroid

→ Depth of analysis.

→ Alternative Comparison.

→ Public Involvement.

5. Impact Comm. → affected parties

→ setting description.

→ summary format.

→ key issue

→ Compliance

Objective of Methodologies

→ Understand the nature and location of project and possible alternative.

→ Identify factor of analysis and assessment obj.

→ Preliminary identification and impact and scoping

→ Baseline studies and evolution in the absence of project.

→ Prediction and assessment of impact and alternative Comparison.

→ Mitigation of impact Mng.

22/8/25 ▷AAA

Rod Cutting problem.

	Profit
1	2
2	3
3	4
4	
5	

1 — 10 cm — 10

Qu

length(n) = 5

Profit = ~~max Profit~~

Length	Profit
1	5
2	8
3	9
4	

Page
Date

Profit = max (profit by excluding new piece , profit by including new piece)

length taken at a time ↓ | length → new piece

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	1	5	6	10	11
3	0	1	1	8	9	13
4	0	1	1	1	9	10
5	0	1	1	1	1	1

5 → 3, 2 13
3, 1, 1

length taken at a time ↓ | lengths →

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	1	5	6	10	11
3	0	1	5	8	10	13
4	0	1	5	8	10	13

Profit = 13

Max.

	Profit
1	2
2	5
3	7
4	8

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	2	4	6	8	10
2	0	2	5	7	10	12
3	0	2	5	9	11	14
4	0	2	5	9	11	14

Page
Date

Ex: 2/2/1 / 12
2/1/1/1

3	2
9	5

= 14 profit

Recursive Top down implementation:

CUT_ROD (p, n)
price length.

1. if n == 0
2. return 0
3. q = = -∞
4. [for i = 1 to n] → as for call n times and recursion call in the loop that
5. q = max(q, p[i] + cut_rod(p, n-1)) why(i)
6. return q

$O(2^n)$

(A) Memoized_Cut_Rod (p, n)

1. let r[0 ... n] to an array → use to store the result for reuse purpose.
2. for i = 0 to n
3. r[i] = -∞ (empty)
4. return Memoized_cut_rod (p, n, r) AUX

$O(n^2)$

(B) Memoized_cut_rod_AUX (p, n, r)

1. if r[n] >= 0
2. return r[n]

] if already profit.

3. if $n == 0$
4. $q = 0$
5. else $q = -\infty$
6. for $i = 1$ to n → reduce no. of recursion call as new array.
7. $q = \max(q, P[i] + \text{MemoCutRodAux}(p, n-i, r))$
8. $r[n] = q$
9. return q

Bottom up CutRod(p, n) (Tabular approach) → Break problem into sub problem. → no recursion.

1. Let $r[0 \ldots n]$ be an array.
2. $r[0] = 0$
3. for $j = 1$ to n
4. $q = -\infty$
5. for $i = 1$ to j
6. $q = \max(q, p[i] + r[j-i])$
7. $r[j] = q$
8. return $r[n]$

$O(n^2)$

Q. Longest Common subsequence.

→ string Matching algo.
→ spam checking & searching Browse.
→ doesn't search complete word.
but char of string.
→ always move forward not backward.
→ It is not about finding the exact match but it will find same order or sequence of char. occuring in the pattern. interest Matching is not allowed.

```
int LCS [i,j]
{
if ('A[i]==\0' || B[j]==='\0')
    return 0;
Else if (A[i] == B[j])
    return 1+LCS(i+1,j+1);
else
    return max(LCS(i+1,j), LCS(i,j+1))
```

$O(m \times n)$

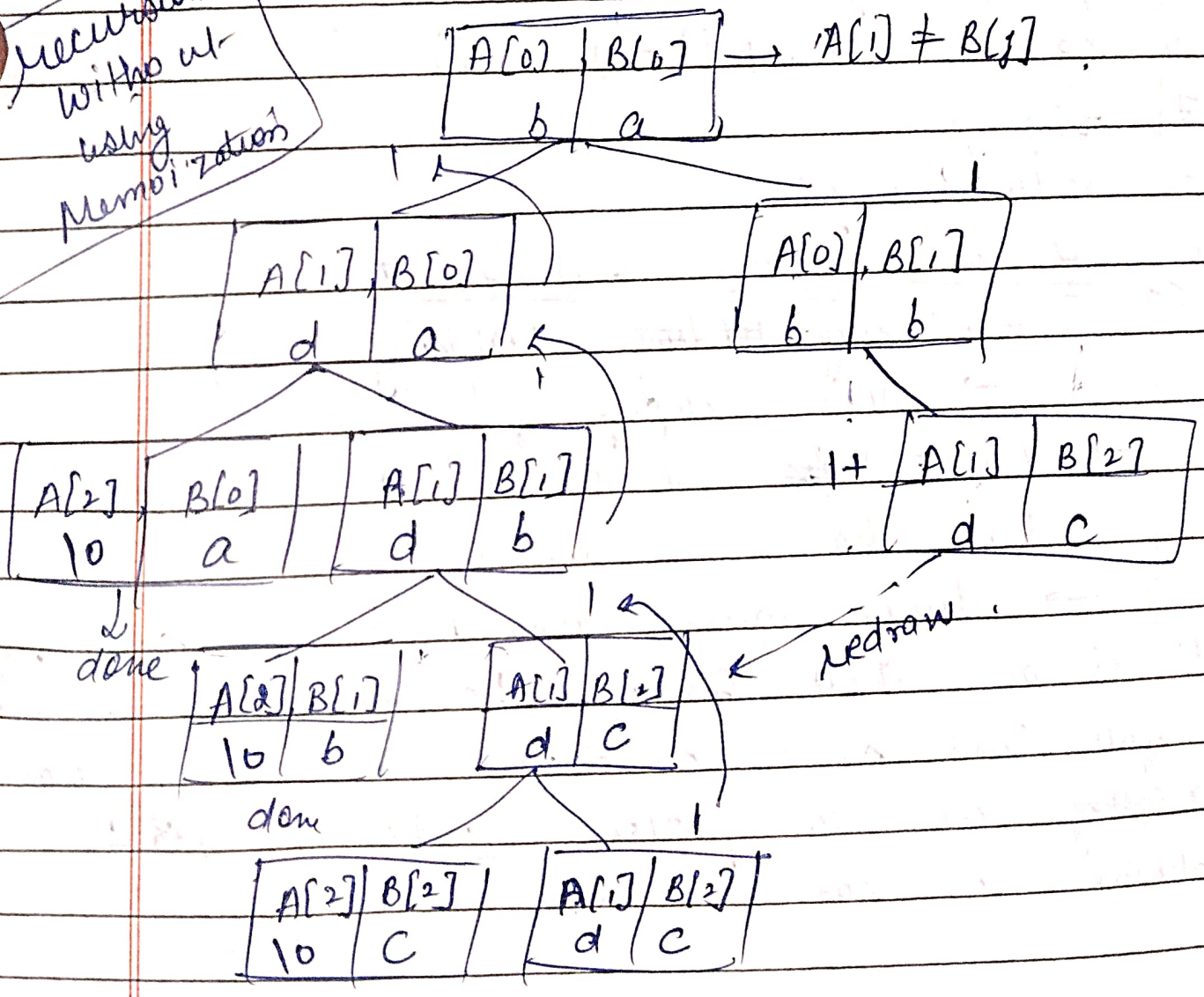
$O(n^2)$

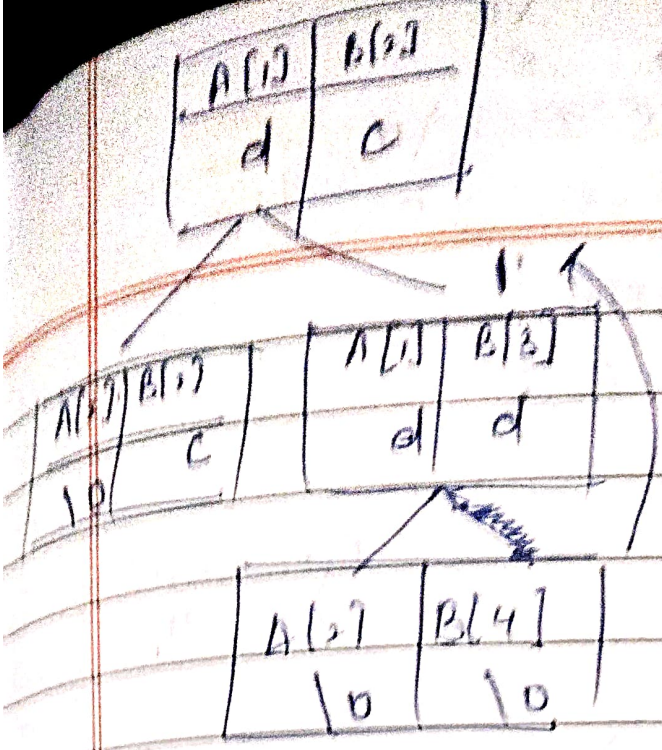
E.g. A = bd B = abcd

A = | b | d | \0 |

B = | a | b | c | d | \0 |
0 1 2 3 4

recursion without using Memoization





Memorization.

→ Tabular form. (~~Memorization~~)

	0	a	b	c	d
0	0	0	0	0	0
b	0	0	1	1	1
d	0	0	1	1	2

if (A[i] = B[j])

LCS [i, j] = 1 + LCS [i+1, j-1]
(1 + diagonal)

else

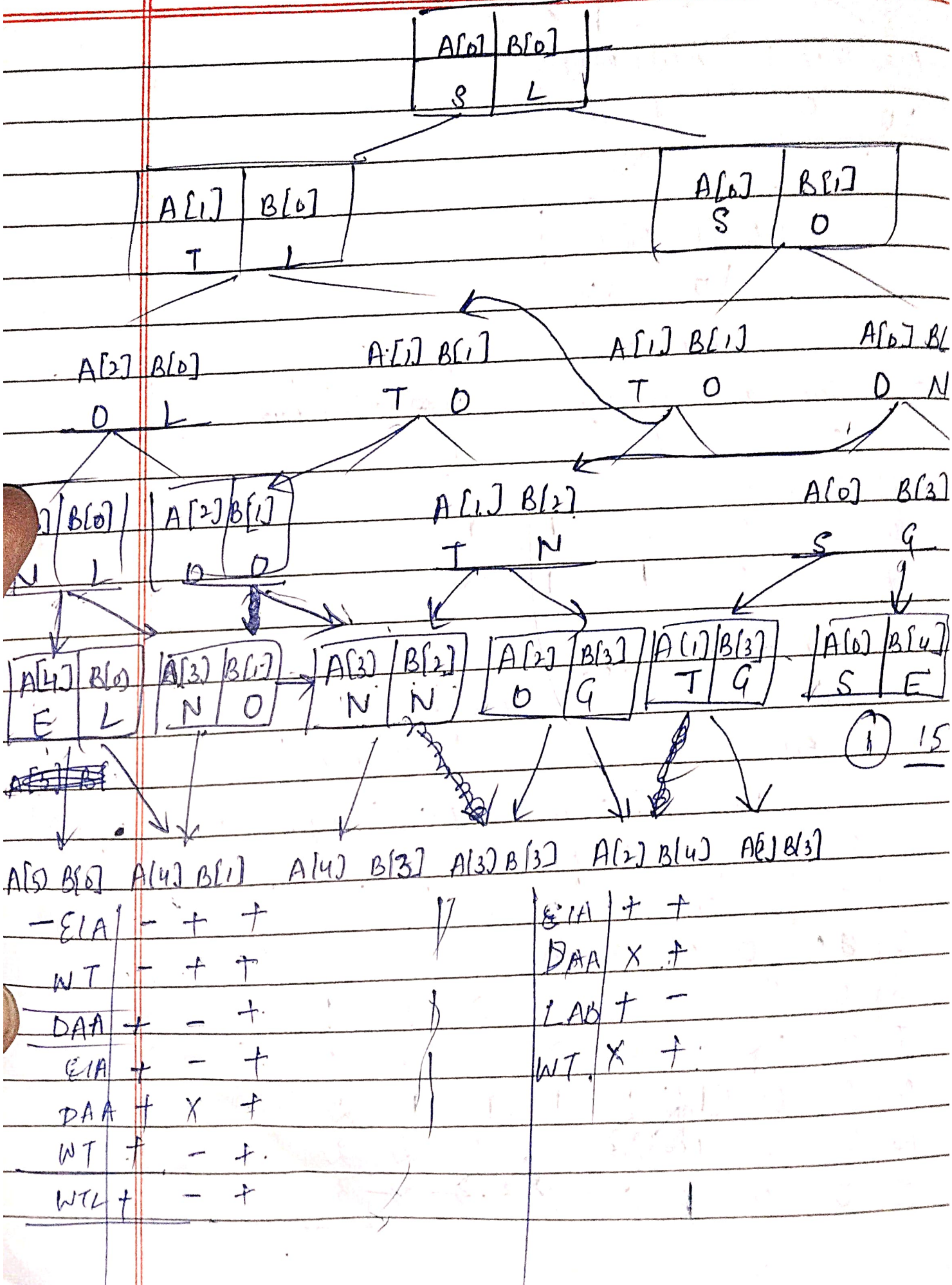
LCS[i, j] = max (LCS [i+1, j], LCS [i, j-1])

STONE LONGEST

	0	L	O	N	G	E	S	T
0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	1	1
T	0	0	0	0	0	0	1	2
O	0	0	1	1	1	1	1	2
N	0	0	1	2	2	2	2	2
E	0	0	1	2	3	3	3	3

A	S	T	O	N	E	\0		
	0	1	2	3	4	5		
B	L	O	N	G	E	S	T	\0
	0	1	2	3	4	5	6	7

Page
Date



-EIA	-	+	+
WT	-	+	+
DAA	+	-	+
EIA	+	-	+
DAA	+	X	+
WT	+	-	+
WTL	+	-	+

EIA	+	+
DAA	X	+
LAB	+	-
WT	X	+

1 15

10/9/25

Optimal Binary Search Trees.

Four keys, To find out minimal Cost

Page
Date

	1	2	3	4
keys	10	20	30	40
freq.	4	2	6	3

The cost is computed using minimum no. of comparison required to find ____

freq.

(5) 4

(10) 3 freq.

(5) 4×1

(10) 2×3

(10) 3×1

(5) 4×2

4+6 = 10 Better. 3+8 = 11

j / i	0	1	2	3	4
0	0	4	8	20	26
1		0	2	10	16
2			0	6	12
3				0	3
4					0

(10) (30) (0,2) (3,4)

step 1:- find value $j-i=0$

step 2:- $j-i=1$

(0,1) (1,2) (2,3) (3,4)

$Cost[i,j] = \min_{i<k\leq j} \{ c[i,k-1] + c[k,j] \} + w[i,j]$

$c[0,0] + c[1,2] = 0+2$

$c[0,1] + c[2,2] = 4+0=4$

10/9/25

Optimal Binary Search Tree:

Four keys, To find out minimum cost

	1	2	3	4
keys	10	20	30	40
freq.	4	2	6	3

The cost is computed using minimum no. of comparison required to find ——

freq.

(5) 4

freq.

(10) 3

(5) 4×1

(10) 2×3

4+6 = 10 Better: 3+8 = 11

(16) 3×1

(5) 4×2

j \ i	0	1	2	3	4
0	0	4	8	20	21
1		0	2	10	16
2			0	6	12
3				0	3
4					0

(10) (30) (0,2) (3,4)

Step 1:- find value $j-i=0$

Step 2:- $j-i=1$

(0,1) (1,2) (2,3) (3,4)

$$Cost[i,j] = \min_{i<k\le j} \{ c[i,k-1] + c[k,j] \} + w[i,j]$$

$c[0,0] + c[1,2] = 0+4=$

$c[0,1] + c[2,2] \quad 4+0=4$

$j-i=2$

step 3

c(0,2)	c(1,3)	c(2,4)
8	10	14
10	14	10
12	12	15
15		

step 4 $j-i=3$

c[0,3]	c[1,4]
10, 20, 30	20, 30, 40

$i=0$ $k=2$ $j=3$

$c[0,0]+c[1,3] = 0+10+12 = 22$

$c[0,1]+c[2,3] = 4+6+12 = 22$

$c[0,2]+c[3,3] = 8+0+12 = 20$ ✓

$c[1,4]$

$c[1,1]+c[2,4] = 0+12+11 = 23$

$c[1,2]+c[3,4] = 4+3+11 = 16$

$c[1,3]+c[4,4] = 10+0+11 = 21$

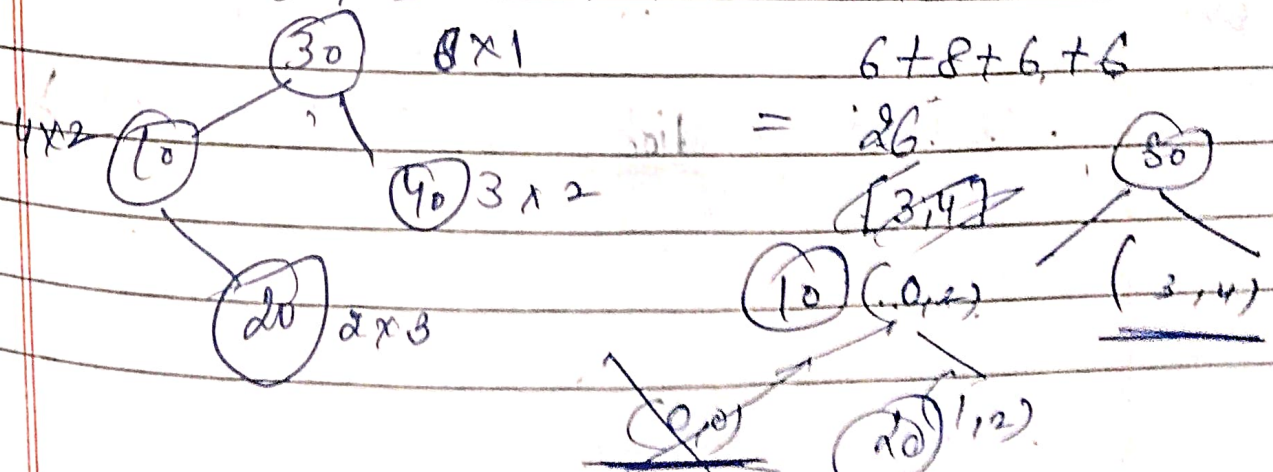
step 5 $j-i=4$

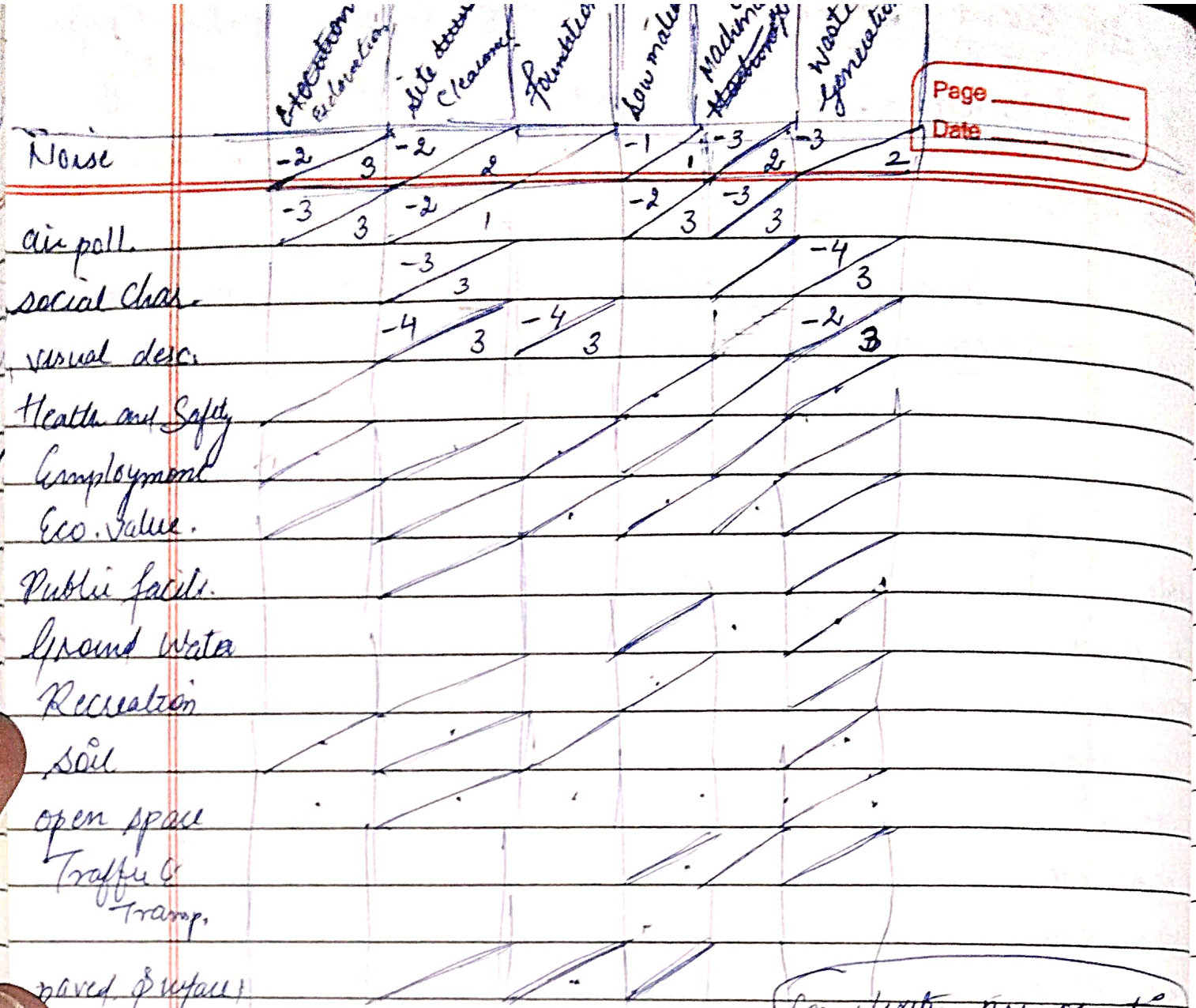
$c[0,0]+c[1,4] = 0+16+15 = 31$

$c[0,1]+c[2,4] = 4+12+15 = 27$

$c[0,2]+c[3,4] = 8+3+15 = 26$

$c[0,3]+c[4,4] = 20+0+15 = 35$





Page
Date

Complexity per operation

2/9/25 Amortized Analysis. →

Asymptotic Analysis → always ~~chose~~ chose highest value as we consider worst or max limit (loose Bound)

20 item each Rs 1

1 " of Rs 100

Asymptotic Anal.

= 2100.

Consider each Item of Rs 100

Amortized Analysis

(20×1) + (100×1)

= 210

H analysis average cost per operation. as worst case running time using asymptotic time complexity may not give a tight bound

Page
Date

Augmented Stack → pop, push, multipop
(b) (1) (k)

total pop = n $k \leq n$.

Methods

(1) Agrigate (2) Accounting (3) Potential

Average

1. Agrigate Method works on the prin. of finding the cost by n
2. Accounting Method assigns some cost to each operation and stores the credit value for each opsn it overcharges cheap operations to fund expensive operation. to find the cost of each opsn
3. It defines the fun called as potential fun to measure the work done in order to find out cost of each opsn

Accounting Method — opsn — credit

push — $2

pop — $0

Dynamic Table → for e.g. vector, list.

Table expension → load

Table Contraction → if expension is utilize atleast 25% then expand or contract (reduce)

Time will insert 8 elements into empty table

operations: 1 → 2 → 4 → 8
copy copy copy.

insert 1: [x] inserted, copy

insert 2: [x][x] copy, insert

3: [x][x][x][ ] copy, insert

4: [x][x][x][x] insert

5: [x][x][x][x][ ][ ][ ][ ] copy, insert

6: [x][x][x][x][x][x][ ][ ] insert

7: [x][x][x][x][x][x][x][ ]

8: [x][x][x][x][x][x][x][x]

only (entry) Copy = 1+2+4 ⇒ 7

insertion = 4 ⇒ 8

Aggregate Method: Total = 7+4 = 11/8 = 1.3 ≈ 1 O(1)

3 → Most Expensive (when copied)

(initial) 1 + 4 → Least Expensive (When only inserted)

Accounting Method.

Insert#	size Before	Capacity Before	Expansion	Copies done	Actual Cost	Credit/ Charge	Bal
1	0	1	N	0	1	3	0
2	1	1	Y	1	copy + insert = 2	3-1	2
3	2	2	Y	2	2+1=3	3-2	3
4	3	4	N	0	1	3-3	3
5	4	4	Y	4	4+1=5	3	5
6	5	6	N	0	1	3	3
7	6	8	N	0	1	3	5
8	7	8	N	0	1	3	7

Page
Date

0

X
3-1
2

[x | x']

2+3 = 5

use augmented stack where cost of push opr^n is 2 and cost of pop or multipop opr^n is 0. You want to add 5 elements and remove

first case - 3 element R

secon " → 5 Element R

Third " → 20 Eleme R

with assumption that Stack already hold 100 element.

(1) [X] 2

(2) [X|X] 4

(3) [X|X|X] 6

(4) [X|X|X|X] 8

(5) [X|X|X|X|X] 10

first case

10 - 5 = 5 (push)

5 - 3 = 2 (pop)

Second case

10 - 5 = 5 push

5 - 5 = 0 pop

O(1) → cost per opr^n.

Third case

Total = 100

multipop → 100 - 1

multipop → 100 - 20.

Potential Method

for d.T the potential fun is defined as

ϕ = no. of element in table $-\left(\frac{\text{capacity}}{2}\right)$

The amortized cost is calculated as

$\hat{C_i}$ = actual cost (C_i) + Change in potential $(\Delta\phi)$

where $\Delta\phi_i = \phi(D_i) - \phi(D_{i-1})$

where D_i : potential for i^{th} operation

if $\Delta\phi_i > 0$, $\hat{C_i} > C_i$ (overcharged)

else $\hat{C_i} < C_i$ (undercharged)

E.g. Total Element = n in stack.

push(1) = n+1

pop(1) = n-1

multipop(k) = n-k

push(1) = 1 + $(n+1-n)$ → $D_i - D_{i-1}$

= 2

pop(1) = 1 + $(n-1-n)$

= 0

multipop(k) = k + $(n-k-n)$

= 0

Q. Find out Solution.

CBA

17/9/25

19/9/25 #Dynamic Table Expension

PAAA

Page
Date

Q) Hash table Size = 4

We want to insert element sequencely as. 10, 20, 30, 40, 50. Resize the table and show its content after each expansion.

Size	Element inserted	array	Expansion
4	-	[_, _, _, _]	No
4	10	10, _ _ _	No.
4	20	10, 20, _ _	No
4	30	10, 20, 30, _	No
4	40	10, 20, 30, 40	No.
8	50	10, 20, 30, 40, 50,	Yes

_ _ _

Ans (2) Intial size = 4

Elements ⇒ 10, ——, 60

and load factor threshold is given 7.5

if load factor = 7.5 double the size.

Size	Elements

elements / size

1/4 2/4 3/4 ...

$\frac{1}{4}$, $\frac{2}{4}$, $\frac{3}{4}$, ~~$\frac{4}{4}$, $\frac{5}{4}$, $\frac{6}{4}$~~.

0.25 0.5 0.75 1 1.25 1.5

Page
Date

[1, 2 3 _]

1/4 2/4 3/4

0.75 Expand.

[1, 2, 3, 1/4 5 6 ~~_~~]

4/8 5/8 6/8

0.25 Expand. ~ ~ ~

Qw A Table expands by a factor of 1.5 instead of 2 when its full.

Initial size = 6 ⇒ 6 × 1.5

No. of elements = 10 ⇒ 9.0

Qw find the cost of a DTE using aggregate Method a Condition is Table size beam double when it is full.

Total El. → 8 Table size = 4

Let insertion Cost → 1

Copy Cost → 1

Expansion Cost → 1

[_ _ _ _] → 4

↓

[_ _ _ _ _ _ _ _]

4 + 4 + 5

= 13 Copy + Expansion

8 Cost.

= 1.5

with asymtotic notation : O(4)

Q1 Contraction

[A B C D E F GH]

Page
Date

Elements = 8 Initial → 2.

Q2 Table Expansion → when table full = 2.

deleted element = 5.

deleted element = 3 by checking the condition of 25%.

[1 2] [3 4 5 6] $\frac{1}{4}$ = 25% ✓

[1,2] [1 2 3 4]

1 2 3 4 5 6 7 8

1 2 3 4 5 6 7 8

1 2 3 4 5 6 7 8

8 25% = $\frac{2}{8} \times \frac{25}{100}$ 2

1 2 3 4 5 6 7 8

1 2 3 _ _ _

18/9/25 #Object → any real world entity.

→ instance of any car.

Exception Handling → try, catch, throw, finally.

async, await.

array → ascending order → use objects

Content no. to word → Try, Catch, finally, throw.

22 TWENTY TWO

Q-1

	A1	A2	A3	A4	A5	A6
S.	1	3	0	5	8	5
f	2	4	6	7	9	9

Page
Date

$k=1$

$m=2$

$S[2] \geq f[1]$

$3 \geq 2$ ✓

Put 2

$k=2$

$m=3$

$S[3] \geq f[2]$

$0 \geq 4$ ✗

$m=4$

$S[4] \geq f[2]$

$5 \geq 4$ ✓

4

$k=4$

$m=5$

$S[5] \geq f[4]$

$8 \geq 7$ ✓

5

$k=5$

$m=6$

$S[6] \geq f[5]$

$5 \geq 9$ ✗

$4 \rightarrow 2 \rightarrow 4 \rightarrow 5$

Capacity = 2
Insertion = 1
& remain = element copy.

Page
Date

(1) aggregate Method to Calculate
Total = 8.

Insert 1: [x] insert
2: [x|x] insert
3: [x|x|x|] copy + insert → 2
4: [x|x|x|x] insert
5: [x|x|x|x|x|] [] [] in + copy → 4
6: [x|x|x|x|x|x|] [] insert
7: in
8: in.

Total = 10/8 = 5/4 = 1.5 ~~= O(1)~~
= 8+6 = 14/8 = 7/4 = 1.75

Element	Capacity	insert	copy	Cost		Total
1	2	1	—	1	insertion	1
2	2	1	—	1	insert	2
3	4	1	1	copy+insert	(2+1)	5
4	4	1	—	1	insert	6
5	8	1	1	4+1		11
6	8	1	—	1		12
7	8	1	—	1		13
8	8	1	—	1		14

14/8 = 1.75

Q3 / Q4.

X = AGGTAB Y = GXTXAYB

Capacity → 4

Expansion → Page Overflows

Contraction

(a)

Element

1 [X][ ][ ][ ] 1/4

2 [X][X][ ][ ]

3 [X][X][X][ ]

4 [X][X][X][X]

5 [X][X][X][X][ ][ ][ ][ ] 8/4 = 2

6 [X][X][X][X][X][X][ ][ ]

7 [X][X][X][X][X][X][X][ ]

8 [X][X][X][X][X][X][X][X]

9 [X][X][X][X][X][X][X][X][X][ ][ ][ ][ ][ ][ ][ ] 16/4 = 4

10 [X][X][X][X][X][X][X][X][X][X][ ][ ][ ][ ][ ][ ]

element

1 10 + 6 → 9 + 7

2 → 8 + 8

3 → 7 + 9

4 → 6 + 10

5 → 5 + 11

6 4 + 4 → 4 + 12

7 3 + 5 → ~~2 + 6~~ → 3 + 13 → ~~2 + 6~~ → 3 + 5

8 ~~2 + 6~~ → 2 + 6 → 2 + 6

7 + 1

6 + 2

5 + 3

4 + 4

3 + 5

2 + 6

3 + 1

2 + 2

Q3

X = A G G T A B

Y = G X T X A Y B

	0	G	x	T	x	A	Y	B
0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	1	1	1
G	0	1	1	1	1	1	1	1
G	0	2	1	1	1	1	1	1
T	0	1	1	2	2	2	2	2
A	0	1	1	2	2	3	3	3
B	0	1	1	2	2	3	3	4

	0	G	x	T	x	A	Y	B
0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	1	1	1
G	0	1	1	1	1	1	1	1
G	0	2	2	2	2	2	2	2
T	0	2	2	3	3	3	3	3
A	0	2	2	3	3	4	4	4
B	0	2	2	3	3	4	4	5

G, G T A B