AOA

Rayyan Shabbir

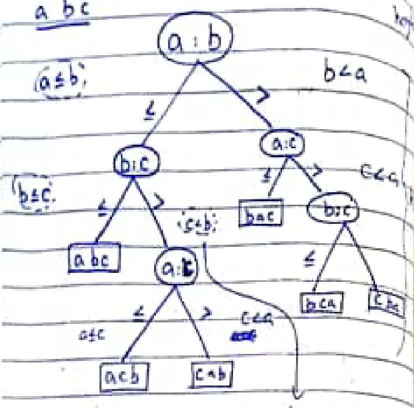
BITF20M535

SEM #05

Decision Tree : suppose we have 3 elements.

$>$ → This comparison operator is used.

a b c



it may b will be n lost now only compare a and c.

if

a b c
4 2 3

$\boxed{b\ c\ a}$ → sorted

$\boxed{2\ 3\ 4}$

• This decision Tree can be made for 3, 4, 5 ... elements also.

• Here no extra / unnecessary (not) comparisions we do.

For worst case

↳ we have to do 3 comparisions

↳ height of Tree

no. of comparisions = height of depest branch

Running time for sorting here = $O(h)$
↳ height of tree

• This tree have '6' leaves
↳ By This no. of combination can be made.

6:52

Comparing prv algo i.e 1.5n-2 ~~[illegible]~~ with sorting algo.

prv algo: $\{1.5^1 \times 2^{1}\}$ (i.e. i for i) not increases with this
constant

sorting: $(10)' \times 2^{floor}$
Pgn It increases with this

GREEDY ALGORITHMS,

Coin Change Problem.

GIVEN Infinite supply of all notes.

notes (5 = {1, 2, 5, 10, 20, 50, 100, 500, 1000, 20[illegible]

Item Rs : 13

given Rs: 1000

TO FIND X: 987 To give change

987 ⇒ change (how this change will be given?)

So, given notes

notes:	500	100	~~5~~ 50	20	10	5	[illegible]
	1	4	1	1	1	1	[illegible]

8 notes total

This solution is the min[illegible] one.

Logic: see which max note can be given.
and how many notes that will be [illegible]

→ This solution gives minimum no. of notes

Q why this solution is optimal / minimum ?

bcz we pick the max first without seeing of other remaining notes.

→ ↳ This is greedy choice/algorithm

→ ↳ This lead to optimal greedy solution.

↳ Our mind 1st try to solve greedily.

↳ we use greedy approach/algo for maximization and minimization solution problem.

Algorithm

Coin Change (S, x)

```
n = S.length
for i = n down to 1 .
        q = ⌊ x / S[i] ⌋
        x = x % S[i]
        if q > 0 then   } to not print 0
            print "q note(s) of S[i]"
        if x = 0 then
            quit / return
```

Total time: $O(n)$

6:52

* greedy approach is best for optimization → maximization or minimi...

The reverse approach to greedy approach is "Dynamic Programming" (DP)

* we also see the choices which are not seem to be best.

↳ we see all possibilities one by one

and by this we find the best solution.

↳ Not always go for the best.

⇒ Two ways of finding Optimal Solution ... DP

Way#01

item	value	weight
1	60	10
2	100	20
3	120	30

item capacity

⇒ V(3,50) profit

optimal Solution, (2, 3) → 220

item	value (v)	weight (w)
1	60	10
2	100	20
3	120	30
4	300	5

as optimal solution

* if we pick (2 and 3) and then get another item 4.

↳ Then we again empty knapsack and again put item in it.

* If we choose '4'th item: (180)

5 ⇒ 300 + V(3,45)

45 ⇒ 480 → profit ...

* So, it is very slow bcz it is recursion and problem repeat in it like factorial example.

→ This property is known as optimal sub-structure property.

* Then after it we see if there is any repetition in recursion solution.
↳ we remove the repetition by setting value once and when it repeats we pick the set value and place it.
Now this is known as "DYNAMIC PROGRAMMING".

* DP starts from Bottom to Top.
Recursion ...

* DP is when you memorized to avoid repeated computation ...
can

V(5, 10)

if not chosen

V(4,10) V(4,9) V(4,8) ...

if not chosen

V(3,10) V(3,9) 3,11

* If we got/see subproblem overlapping → Then we apply DP.

* If no overlapping → Otherwise no solution can be better than this.

Example: How many common subsequence exists in 'X' and 'Y'.

* We cannot change sequence.

X = A B C B D A B

Y = B D C A B A

Sol: * if strings are small we can find it by brute force.

* if strings are large we can find it by

Longest subsequence = BCB.A

BCAB

BDAB

How to find this solution?

→ * By using DP

* Steps.

(1) find optimal sub solution.

(we say we know solution of n-1 problem)

Problem

* X = A B C B D A B

Y = B D C A B A

Z = B D A B

X_6 X_5

Y_4 Y_3

* Many LCS exists in above problem

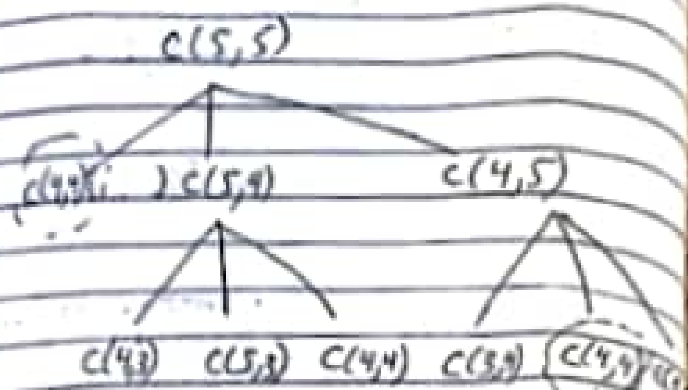
X = A B C B D A B Z

Y = B D C A B A Z

Z = B D A B Z

Graph to check overlapping.

C(5,5)



Thus, there exists overlapping. So, apply DP.

We apply DP in iteration way from bottom to top.

We know LCS of

C(0,0) and C(0,1) and C(1,0)

AOA

Lecture

PM 01

Longest Common Subsequence (LC…

Process problem: To Find LCS

Generalized Form

$X = x_1, x_2, x_3 \dots x_{i-1}, x_i \dots x_{m-1}, x_m$

$Y = y_1, y_2, y_3 \dots y_{j-1}, y_j \dots y_{n-1}, y_n$

Recursive solution

BASE case

$$C(i,j) = \begin{cases} 0 & ; \text{ if } i=0 \text{ or } j… \\ C(i-1, j-1)+1 & ; \text{ if } x_i = y_j \\ \max\{C(i-1,j), C(i,j-1)\} & ; \text{ if } x_i … \end{cases}$$

Now check overlapping a…

problem: Find LCS of below problem

X = A B C B D A B

Y = B D C A B A

* Solution for this problem is DAA.

* It has many LCS (more than 1)

LCS = B C B A

or

LCS = B C A B

or

LCS = B D A B

all are of 4 size.

* if we take from previous LCS

LCS = B C B A

They does not
may be
last element not 'B'.

X_5, Y_5

X = A B C B D A B

Y = B D C A B A

if same LCS will be → X_5, Y_5

if not same LCS will be → X_6, Y_6

or

X_7, Y_5

Confirmed this on pg#01

greedy Algo
↳ Quiz

paper

↳ MCM

↳ LCS

↳ Knapsack

Table

		j						
C			B	D	C	A	B	A
		0	1	2	3	4	5	6
	0	0	0	0	0	0	0	0
A	1	0	↑0	↑0	↑0	↖1	←1	↖1
B	2	0	↖1	←1	←1	↑1	↖2	←2
C	3	0	↑1	↑1	↖2	←2	↑2	↑2
B	4	0	↑1	↑1	↑2	↑2	↖3	←3
D	5	0	↑1	↖2	↑2	↑2	↑3	↑3
A	6	0	↑1	↑2	↑2	↖3	↑3	↖4
B	7	0	↖1	↑2	↑2	↑3	↖4	↑4

6:53

* dimension of : i =

* we want to parenthesize pairing problem.

Solution*

Brute force

#1 ↳ find all possible solution and find common and check the smallest one that will be our solution.

↳ not apply → ambiguous bcz we don't know total no. of sequences.

#2 ↳ if we have 3 2's matrices

↳ AB it have 1 way

if chain size: ways to multiply

(AB) → 1

A → ~~1~~0

ABC → 2

ABCD → 5 ways

A(BCD) ways
1 × 2 = 2

(AB)(CD)
1 × 1 = 1

(ABC)D
2 × 1 = 2

~~ABCD~~ → ~~4~~ 5

((A(BC))D)
((AB)C)D)
((AB)(CD))

1 × 2 = 2
A(BCD)
(AB)(CD) = 1
(ABC)D = 2
5

$(10 \times 100)(100 \times 5 \times 50)$

So $A(BC)$

$50000 + 25000 = 75,000$

$A(BC) \Rightarrow 75,000$

* So, order of multiplication matters.

* So, we have to find "ORDER" by which we have to do minimum no.# of multiplication.

Parsing problem# We have chain of matrices

$A_1\ A_2\ A_3 \ldots A_{k-1}\ A_k\ A_{k+1} \ldots A_n$

* We know all dimensions
  ↳ bcz all are in sequence and compatible.

$p_0 \times p_1 \quad p_1 \times p_2 \quad p_{k-1} \times p_k \quad p_{n-1} \times p_n$

* We take array of size $n+1$ for all matrices

	0	1	2	. . . .	n
P					

$p_0 \times p_n$

* How to multiply these 'n' matrices → (their order) So we have to do min multi…

6:53

AB $i \neq j$
BA $i \neq j$

- Total multiplications = pqr

- Multiplication operation is expensive operation
  it do $O(n^3)$

- We want to see "ABC", total multiplications in below problem.

A: 10×100

10×5

B: 100×5

C: 5×50

	0	1	2	3
P:	10	100	5	50

- First we do AB
  $(AB) \times C$

$((AB)C)$ $(A(BC))$

Or we can do
- first BC
  Then $A \times (BC)$

$(10\times5)(5\times50)$
$5000 + 0 = 5000 + 2500 =$

In ~~AB~~ $(AB)C$
how many scalar multiplication we do in above?

For $AB = 10 \times 100 \times 5 = 5000$

$\underset{10\times5}{(AB)}\ \underset{5\times50}{C}$

$5000 + 2500 = 7500 \Rightarrow (AB)C$

$(10\times100)(100\times50) = 50000$
$0 + 25000 = 25000$
Ans $(A)(BC)$ 75000

6:53

100x5 100x1 3 4

Soln of Previous Problem:

↳ We cannot do greedy soln to this.

↳ take smallest multiplication of matrices and then do multiplication it with some other and repeat.

↳ But this is not OPTIMAL SOLUTION

Not OPTIMAL SOLUTION (case #1)

* if we say we know the soln of A_{k-1} then it means we only know soln of A_{k-1} not of A_k.

So, we cannot do it by incremental way: whenever a new matrix comes, the all solution of multiplication will be ch...

$(A_1 A_2 A_3)$ 100x5 5x100

So Now

Optimal Solution

(case #2) ↳ by dividing

↳ We say optimal solution will be at by dividing at A_k

↳ we do it by recursing dividing

$P_1 \times P_2$ ... P_{n} P_n

A_1 A_2 A_3 ... A_{k-1} A_k A_{k+1} ... A_{n-1} A_n

optimal optimal

* Lets assume optimal solution will be when we divide at A_k.

* So, where we do cut/divide then prior of it and more of it will also be optimal.

bcz if these will be optimal then our overall ans will be optimal.

Now, we create Table
'm' is matrix 'i' is
row and 'j' is column

[illegible]
\# we save 'k' → where we do cut.

Its running time is

$O(n^3)$ → bcz here '3' loops are running.

Greedy Gas Station

We want to find minimum scalar multiplications?

$$\underset{2500}{\underbrace{((A\ B)}}\overset{5000}{\downarrow}\ C)$$

So total multiplications $= 5000 + 2500 = 7500$

$$\underbrace{(A_1 A_2 A_3 \cdots A_{k-1} A_k)}_{order = p_0 \times p_k} \overset{k}{\downarrow} \underbrace{(A_{k+1} \cdots A_{n-1} A_n)}_{order = p_k \times p_n}$$

Total scalar multiplications:

$$m(1,n) = m(1,k) + m(k+1,n) + \underbrace{p_0 p_k p_n}_{overall}$$

A B

Generalized form of previous

$$((A_i\ A_{i+1}\ A_{i+2} \cdots A_{k-1}\ A_k) \overset{k}{\downarrow} (A_{k+1} \cdots A_{j-1}\ A_j))$$

$$m(i,j) = \min_{i \le k < j} \{ m(i,k) + m(k+1,j) + p_{i-1} p_k p_j \}$$

↳ This is a Recursive Solution (Recurrence

Base-Case: if $i=j$ → '0' → total multiplications

AoA

MATRIX CHAIN Multiplication (M...

problem: We have ←

$$A = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}_{2\times 2}$$

$$B = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 4 & 3 \end{bmatrix}_{2\times 3}$$

$$C = \begin{bmatrix} 2 & 1 \\ 3 & 4 \\ 6 & 5 \end{bmatrix}_{3\times 2}$$

* We can multiply two matrix if they are compatible.
if
→ 1st column
→ 2nd rows

$$AB = \begin{bmatrix} (2\times1)+(1\times2) & (2\times3)+(1\times4) & (2\times2)+(1\ldots \\ - & - & - \end{bmatrix}$$

Resultant size = 1st row × 2nd column

if $A = p \times q$

$B = q \times r$

$AB = p \times r$ ⇒ no. of ... A resultant matrix

* We are performing ... $2\times6 = 12$ ...

		B	D	C	A	B	A
	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
A 1	0	↑0	↑0	↑0	↖1	←1	↖1
B 2	0	↖1	←1	←1	↑1	↖2	←2
C 3	0	↑1	↑1	↖2	←2	↑2	↑2
B 4	0	↖1	↑1	↑2	↑2	↖3	←3
D 5	0	↑1	↖2	↑2	↑2	↑3	↑3
A 6	0	↑1	↑2	↑2	↖3	↑3	↖4
B 7	0	↖1	↑2	↑2	↑3	↖4	↑4

↳ table on next page also

* cost to fill this table
  ↳ $= O(mn)$

strategy

* Agr equal hay to (diagnol + 1) kr lein gy

* Agr same nai hongy to top and left mein sy max wali lain gy
  ↳ or agr max bhi same ho to koi bhi.

* Now, we see max profit by seeing directions

* We get max profit → (4) in 7x6 = 4 — So, we start from there
  ↓
  we copy this from bottom

→ top or left pg move krna hai or likhna nai LCS mei

→ Jahan diagonal pg ja rha ho usko likhna hai LCS mei likh dena hai.
  ↓
  Jahan pg match ho woh likhna hai.

LCS = A B C B

↳ This in reverse bcz we start traversing it from last.

So, Reverse it for real LCS

LCS = B C B A

pg No 4

[illegible] Overlapping

C(7,6)

C(6,6) C(7,5)

C(5,6) C(6,5) ; C(6,5) C(7,4)

C(4,6) C(5,5) C(6,5) C(6,4) ; C(7,5) C(6,4)

Also huge overlapping exists

Thus, we have to apply DP here.

Total Leads of this $= \Omega(2^{m+n})$ → Run Time

ab, ek bar to
kam karni subproblem (ne kr ki

---

* Prev Problem is same like brute force.

  So, it is very bad problem.

* Applying Dynamic Problem.

we should start to compute from "Bottom to Top"

As, we know Base-Case 0,0 or 0,1 or 1,0

So, we have to start from bottom to top

Table on next page

<u>observations</u>: (from previous

if we have sequence

$X = x_1\ x_2\ x_3\ \dots\ x_m$

$Y = y_1\ y_2\ y_3\ \dots\ y_n$

$Z = z_1\ z_2\ z_3\ \dots\ z_k$ → common subsequence

(1) if $x_m = y_n = z_k$, then Z_{k-1} is an LCS of X_{m-1} and Y_{n-1}

(2) if $x_m \neq z_k$, then Z_k is an LCS of X_{m-1} and Y_n

(3) if $y_n \neq z_k$, then Z_k is an LCS of X_m and Y_{n-1}

→ Representing lengths of LCS of X_i, Y_j

if really add last element are same

$$C(i,j) = \begin{cases} C(i-1,j-1)+1 & ; \text{ if } x_i = y_j \\ \max[C(i-1,j), C(i,j-1)] & ; \text{ if } x_i \neq y_j \\ 0 & ; \text{ if } i=0 \text{ or } j=0 \end{cases}$$

base case

★ if ~~[illegible]~~ last element are not same we check both two cases.

Base case of above problem / subsequence

★ we stop at either both are '0' or any of them are zero.

★ After this we check if there subproblem overlapping

* It takes $O(n \times m)$ bcz we have table and we only apply '2' for loops.

- we use memory to avoid repetition.

<u>steps of previous way</u>

(1) Apply optimal sub-structure property to make recursive solution.

(2) Now draw Tree (solve some) of Recursive solution

(3) if overlapping exists → remove it by dynamic programming

Lecture

Longest Common Subsequence (L

Example: • two sequence can be made: 1 of choose a sequence and 1 of to ...

① A B C D E F

↳ by this we can make 2^n may subsequences

ABC → substring
ACE → subsequence
EFA → not a subsequence bcz there is no 'A' after 'F'

if elements are n we can only make 2^n subsequences

* we cannot change sequence

② F B D A C E G	① A B C D E F
BE	BE
BD	ABC
BCE	ACE

common sequence = BE

* Many subsequences are common in '1' and '2', ~~these are~~ so, we have to find

4/21

Now, doing "DP"

item	value	wgt
1	6	1
2	10	2
3	12	3
4	30	5

capacity = 8

→ V(4, 8) → we create table of this length

* We make table for do "Dynamic Programming"

i \ w	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	6	6	6	6	6	6	6	6
→2	0	6	10	16	16	16	16	16	16
3	0	6	10	16	18	22	28	28	28
x4	0	6	10	16	18	30	36	40	$\boxed{46}$

Required Soln

~~Avg~~ only fill this row by row

$$V[i, w] = \max \begin{cases} V[I-1, w] \\ 10 + v[1, 1] \\ \boxed{V_i + V[i-1, w-w_i]} \quad \text{if } w_i \le W \end{cases}$$

* Here we only get (by previous table) optimal solution not the items

→ we get items by back tracking.

items = { 1, 2, 4 }

→ pick last i.e 4th item optimal value, then see if this value is copy from previous 3rd value if not then it means item is included. Now, subtract 46 - 30 = 16 → ...

* if we not choose 4th item
  we get 220 profit.

* So, for 4th item, we must know solution of previous 3 items

* Now compare with 4th item optimal solution and without 4th item optimal solution, and pick the highest/best from it.

So on. (means for 5th, 6th ... items etc.)

items					i-1	i			
1	2	3	4	5		6	7	8	9 n

we have optimal solution of ...

Recurrence:

$$V(i, w) = \max \begin{cases} V(I-1, W) & \rightarrow \text{if not chosen} \\ V_i + (i-1, w-w_i) & \rightarrow \text{if } w_i \le W \end{cases}$$

This property is known as optimal sub-structure property

→ As for I, we see solution of I-1. So, it recursion.

Base case of this recursion is:

$V(i, w) = 0$ if $i=0$ or $w=0$

* It takes $\Omega(2^n)$.

* It is not DP, is simple recursive solution.

* It is bad as same as brute force (which checks for all solution individually and pick the best one).

Lecture #03

Knapsack Problem 0/1

AoA

item	v value	w weight	Price per unit or v/w ratio
1	60	10	6
2	100	20	5.1
3	120	30	4

Capacity of sack = 50.

we choose: (By Greedy approach)

w	v
10	60
$\boxed{30}$	$\boxed{160}$

→ max profit we make

↳ This is not optimal solution

↳ Thus, here greedy approach is not good.

#03: do one comparison
two elements

(1) (1) (1) (3)

2 5 8 1 3 6 4 7

max: 8

min: 1

→ for each 2 element we do 3 comparisons

↳ 1 for bet. the element

↳ 1 with max

↳ 1 with min

total = 3

So total = $\frac{3(n-2)}{2} + 1$ comparisons

1 → for 1st comparing start

$= 1.5n - 3 + 1$

$\boxed{\text{comparisons} = 1.5n - 2}$ → It is the best we can do.

if n = 100

Comparisons ...

$= 1.5(100) - 2$

#04: do 3 element 1

comparing bet. three 3 elements comparisons

$3 + 2 = 5$ → for compare with max and min

2 5 8 1 3 6 4 7

MAX =

min

$= \frac{5(n-2)}{3} + 1$

$= \frac{5}{3} n - 2$

$\boxed{\text{comparisons} = 1.6n - 2}$ } not better than '#03'

if n = 100

$\boxed{\text{Comparisons} = 164.3}$

$= \frac{5(100-2)}{3} + 1$

=

Prove of this Stirlings Formula:

$lg(n \times (n-1) \times (n-2) \dots)$

where

$lg\,n! =$ ~~[illegible]~~

$lg\,n! = lg\,n^n$

$lg\,n! = n\,lg\,n$

So,

$$h = \Omega(n\,lg\,n)$$ (atleast)

↳ As we know height is its running time.

So, its run time is atleast $\Omega(n\,lg\,n)$

↳ So no program can be made which sorts and runs in less than $n\,lg\,n$

This running time is of comparison based algo.

To find

Problem: Max element from data. As well as min element from data

2 5 8 1 3 6 4 7

ways

#01: make 1st element greatest and do comparison

first find max then find min.

max = 2

So, it is done in $(n-1)$ comparisons.

Also for min $(n-1)$ comparisons

So, total:

$2(n-1) = 2n-2$

→ if $n = 100$

198 → total comparisons.

#02: use 1 loop and compare.

max = 2

min = 2

↳ here also $n-1$

Leaves

$n! \le l \le 2^h$

If Total element = 3

$3 \times 2 \times 1$

$= 3!$

leaves = 6

If total elements = 4

leaves = 4!

In n elements

leaves = $n!$

So,

$n! \le l \le 2^h$

↳ atleast n leaves but can also be greater than this.

Max

$h = 2$

If $2^2 = 4$ leaves

$h = 4$

$2^4 = 16$ leaves

If height = h

$2^h \Rightarrow$ max no. of leaves

So,

$\Rightarrow n! \le 2^h$

or

$\Rightarrow 2^h \ge n!$

To find height:

$\Rightarrow \lg 2^h \ge \lg n!$

$h \ge \lg n!$ $\boxed{\because \lg_2 2 = 1}$

[illegible] atleast $\lg n!$.