UNIT-3 Dynamic programming

Is a power arithmetic technique used to solve optimization problem by breaking down into simpler sub problems. It is particularly useful when the problem has overlapping sub problems and optimal sub structure that means the optimal soln to the problem can be constructed from optimal soln of its sub problems

Overlapping sub problems

The problem can be divided into smaller sub problems ~~are~~ and the same sub problems are solved multiple times.

Optimal sub structure

The optimal soln to the problem can be constructed from the optimal soln of its sub problems.

Memoization —

The technique to store the results of expensive function call and reuse them when the same inputs occure again

Tabulation

A bottom-up approach where the results of sub problems are stored in a table and used to solve larger sub problems.

Advantage of DP :-

(i) Avoid redundand computation
(ii) ~~Effecti~~ Efficient
(iii) simple

Fibonacci sequence problem —

Fib(0) = 0
Fib(1) = 1
Fib(n) = Fib(n-1) + Fib(n-2)
n ≥ 2

Naive Recursive approach

1- Base Case :
if n ≤ 1 return n

2- Recursive case :
Return of Fib(n-1) + Fib(n-2)

II Dynamic programming approach

1- Base case
if n ≤ 1 , return n

2- Initiali of DP array
Create an array of size n+1 to store Fibonacci series.

3- Set base case
dp[0] = 0
dp[1] = 1

4- Fill the DP array
for each i from 2 to n
Compute dp[i] = dp[n+1] + dp[n-2]

III Recursive

1- fib(5)

fib(5) = fib(4) + fib(3)
3 + 2 = 5

2- fib(4)

fib(4) = fib(3) + fib(2)
2 + 1 = 3

3- Fib(3)

Fib(3) = Fib(2) + Fib(1)
1 + 1 = 2

4- Fib(2) = Fib(1) + Fib(0)
1 + 0 = 1

5- Base case

Fib(1) = 1
Fib(0) = 0

→ intial dp = [0, 1, 0, 0, 0]

dp[0] = 0 dp[1] = 1

dp[2] = dp[1] + dp[0]

dp[3] = dp[2] + dp[1]
1+1 = 2

dp[4] = dp[3] + dp[2]
2+1 = 3

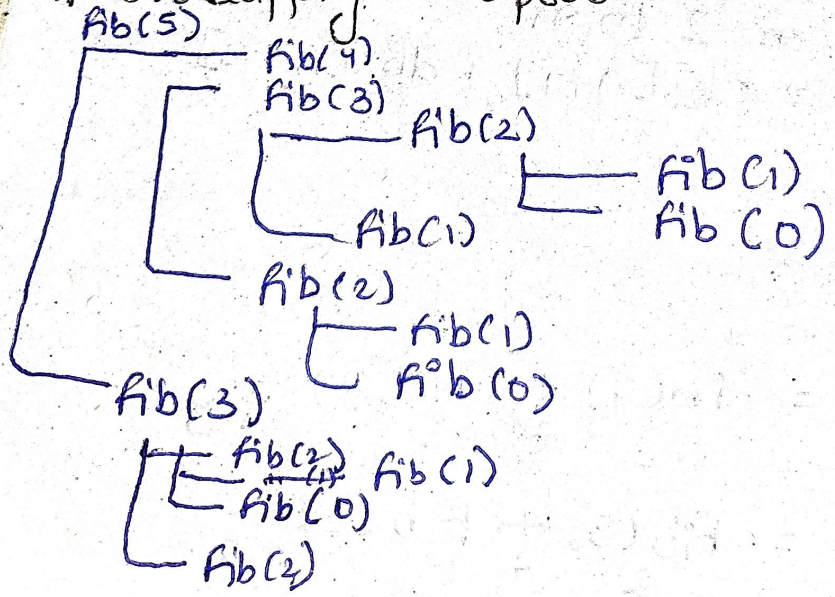
dp[5] = dp[4] + dp[3]
= 3+2 = 5

Recursive -

It is stop because it takes exponencial time

dynamic - It is fast because it takes linear time

Overlapping subproblems



2 - Optimal Substructure.

Fib(5) = Fib(4) + Fib(3)

If we know the correct values of Fib(4) and Fib(3). then we can compute Fib(5)

3 - Memoization (Iterative memoization)

Top-down approach where we start with the original problem and solve sub problems as needed.

4 - Tabulation

Bottom-up approach where solve the problem iteratively by filling a table (usually an array) with the results of sub problems

Iterative memoization Program

1 - memo table

i	memo [i]
0	-1
1	-1
2	-1
3	-1
4	-1
5	-1

2 - Stack

i	value
0	5
1	-
2	-
3	-
4	-
5	-

Push '5' onto the stack

3 - top = 1 -1+1=0

i	value
0	5
1	-
2	-
3	-
4	-
5	-

4 - Execution Loop

Check if current = 5 memo [5] is computed

memo [5] = -1 (not computed)

if (memo[current] != -1)

X

if (current <= 1)

X

0/1 Knapsack :-

It is a classic optimization problem where you have a set of items with a weight and a value (profit) the goal is to maximize the total value of items placed in knapsack how exceeding its capacity. 0 means excluded and 1 means included fraction is not allow

0. item	weight	value (v)
1	2	12
2	1	10
3	3	20
4	2	15

Soln

Knapsack capacity (W) = 5

rearrange above table decending order by weight (w)

item	weight (w)	value (v)
3	3	20
1	2	12
4	2	15
2	1	10

Item/ Capacity	0	1	2	3	4	5
0 Item	0	0	0	0	0	0
3 (w=3 v=20)	0	0	0	20	20	20
1 (w=2 v=12)	~~0~~	~~0~~	12	20	20	32
4 (w=2 v=15)	0	0	15	20	27	35

e you have
value
re the
psack
s excluded
not

2 (w=1, v=10)	0	10	15	25	30	37

maximum profit = 37

verification [0, 1, 1, 1] binary vector

weight : $0 \times 3 + 1 \times 2 + 1 \times 2 + 1 \times 1 = 5$

value : $0 \times 20 + 1 \times 12 + 1 \times 15 + 1 \times 10$

$= 12 + 15 + 10 = 37$

Q2

item	weight	value (v)
1	2	12
2	1	10
3	3	20
4	2	10

W = 5

item	weight (w)	value (v)
3	3	20
1	2	12
4	2	10
2	1	10

Item/capacity	0	1	2	3	4	5
0 item	0	0	0	0	0	0
3 (w=3, v=20)	0	0	0	20	20	20
1 (w=2, v=12)	0	0	12	20	20	32
4 (w=2, v=10)	0	0	12	20	22	32
2 (w=1, v=10)	0	0	12	20	30	32

Verification [0, 1, 1, 1]

Weight : $0\times3 + 0\times2 + 1\times2 + 1\times1 = 5$

Value : $0\times20 + 1\times12 + 1\times10 + 1\times10$

$= 12 + 10 + 10 = 32$

Q3.

Item	weight (w)	value (v)	
1	60	10	
2	100	20	W = 50
3	120	30	

Matrix chain multiplication (MCM)

MCM is an optimization problem that envolves finding the most efficient way to multiply a sequence of matrices. The goal is to minimize the total no. of scaler multiplication required it is typically solved using DP

Basic condition for multiplication of matrices

Let's suppose there are two matrices for multiplication

$M_1 = m \times n \quad M_2 = n \times p$

$m \times \boxed{n = n} \times p$

must be equal

where n is column for M_1 and n is row for M_2

let $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$ $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$

$M_1 = m \times n$ $M_3 = p \times q$ dimension
$M_2 = n \times p$ $M_4 = q \times r$ $m \times n$, $p \times q$

$M_1 = 2 \times 3$ $M_2 = 3 \times 2$

2 3 2 = 12 (number of operation)

There are 12 multiplication operations to get result b/w k and v

Ex - $M_1 \times M_2 \times M_3$

$M_1 = 2 \times 3$ $M_2 = 3 \times 4$ $M_3 = 4 \times 2$

$M_1 \times M_2 \times M_3$

2 3 3 4 4 2

d_0 d_1 d_2 d_3

Possible ways $(M_1 \times M_2) \times M_3$ $M_1 \times (M_2 \times M_3)$
it holds associative property.

$(M_1 \times M_2) \times M_3$

$(2 \times 3 \times 4) \times 4 \times 2$

2 4 0

2 4 2

$24 + 0$

$2 \times 4 \times 2$

$24 + 0 + 16$

$= 40$

$M_1 \times (M_2 \times M_3)$

$2 \times 3 (3 \times 4 \times 2)$

0 24

2 3 2

$0 + 24 = 24$

$2 \times 3 \times 2 = 12$

$24 + 0 + 12$

$= 36$

no. of operations is 36

$36 < 40$

So 36 has less affect of multiplication to get result.

But what if their is n no. of matrices then we have to calculate total no. of possibilities by given formula

Total no. of parenthesization

Chain possibilities $= \frac{^{2(n-1)}C_{n-1}}{n}$ $\left[{}^{n}C_{r} = \frac{n!}{r!(n-r)!} \right]$

if $n=4$

$\frac{^{2(4-1)}C_{4-1}}{4} = \frac{^{6}C_{3}}{4}$ $\left[\frac{6!}{3!(4-3)!} \right.$

$= \frac{20}{4} = 5$ $= \frac{6!}{3! \times 3!}$

5 (parenthesization)

$= \frac{6 \times 5 \times 4 \times 3 \times 2 \times 1}{3 \times 2 \times 1 \times 3 \times 2 \times 1}$

$= \frac{720}{36} = 20$

$^{n}C_{r} = \frac{n!}{r!(n-r)!}$

n	No. of Parenthesization
1	1
2	1
3	2
4	5
5	14
6	32
7	132
8	729
9	1430

10

Suppos

formula

$i \leq k < j$

Set C[

C[1, 2] =

C[2, 3

C[3,

C[1

C[

C[1

multiplication no. of

10 | 4836

Suppose matrices

	A_1		A_2		A_3		A_4	
3	2	2		4	4	2	2	5
d_0		d_1			d_2		d_3	d_4

Formula :- $C[i,j] = \min_{i \le k < j} \{C[i,k] + C[k+1,j] + d_{i-1} \times d_k \times d_j\}$

Set $C[1,1] = C[2,2] = C[3,3] = C[4,4] = 0$

$C[1,2] = \min_{1 \le 1 < 2} \{C[1,1] + C[2,2] + 3 \times 2 \times 4\}$

$= 0 + 0 + 24 = 24$

$C[2,3] = \min_{2 \le 2 < 3} \{C[2,2] + C[3,3] + 2 \times 4 \times 2\}$

$= 0 + 0 + 16 = 16$

$C[3,4] = \min_{3 \le 3 < 4} \{C[3,3] + C[4,4] + 4 \times 2 \times 5\}$

$= 0 + 0 + 40$

	1	2	3	4
1	0	24	28	58
2		0	16	36
3			0	40
4				0

Cost matrix

	2	3	4
1	1	1	3
2		2	3
3			3
4			

(k - matrix)

$C[1,3] = \min_{k=1,\ k=2} \{C[1,1] + C[2,3] + 3 \times 2 \times 2,\ C[1,2] + C[3,3] + 3 \times 4 \times 2\}$

$A_1 \times A_2 \times A_3$

$C[1,3] = \min_{1 \le 1 < 3} \{C[1,1] + C[2,3] + 3 \times 2 \times 2\}$

$= 0 + 16 + 12 = 28$

$C[1,3] = \min_{1 \le 2 < 3} \{C[1,2] + C[3,3] + 3 \times 4 \times 2 = 0 + 24 + 24 = 48\}$

$$C[2,4] = \begin{cases} \min_{2 \le 2 < 4} \{C[2,2] + C[3,4]\} + 2 \times 4 \times 5 = 0 + 40 + 40 = 80 \\ \min_{2 \le 3 < 4} \{C[2,3] + C[4,4]\} + 2 \times 2 \times 5 = 16 + 0 + 20 = 36 \end{cases}$$

$$C[1,4] \begin{cases} \min_{1 \le 1 < 4} \{C[1,1] + C[2,4]\} + 3 \times 2 \times 5 = 0 + 36 + 30 = 66 \\ \min_{1 \le 2 < 4} \{C[1,2] + C[3,4]\} + 3 \times 4 \times 5 = 24 + 40 + 60 = 124 \\ \min_{1 \le 3 \le 4} \{C[1,3] + C[4,4]\} + 3 \times 2 \times 5 = 28 + 0 + 30 = 58 \end{cases}$$

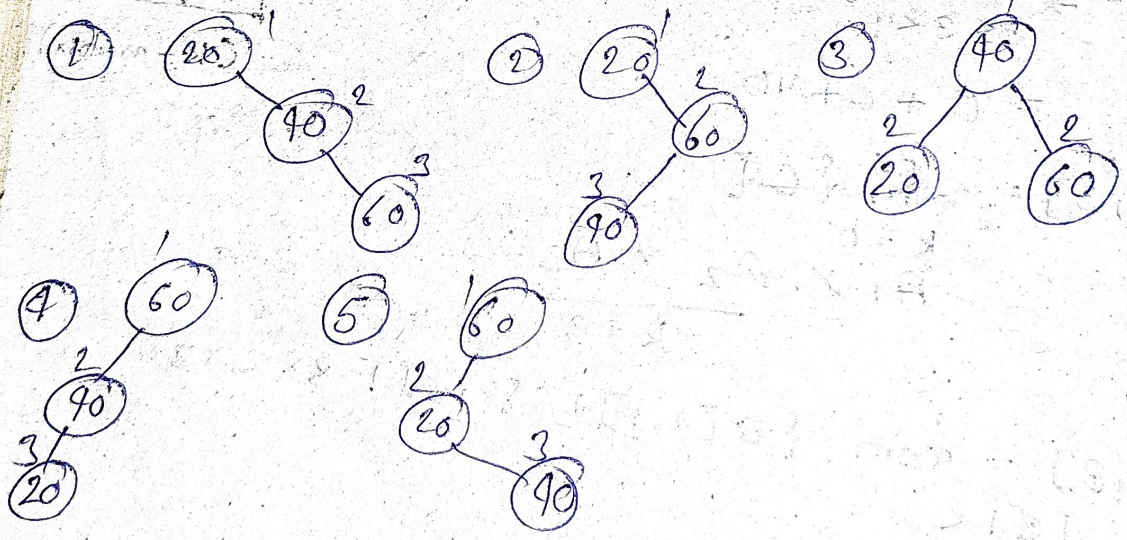
Optimal Binary Search Tree (OBST) ⟶

Qw Keys 20 40 60

frequency 3 2 5

For n keys possible trees

$$= \frac{^{2n}C_n}{n+1} = \frac{2 \times {}^3C_3}{3+1} = \frac{{}^6C_3}{4} = \frac{20}{4} = 5$$



① $1\times3 + 2\times2 + 3\times5 = 22$

② $1\times3 + 2\times5 + 3\times2 = 19$

③ $1\times2 + 2\times3 + 2\times5 = 18$

④ $1\times5 + 2\times2 + 3\times3 = 18$

⑤ $1\times5 + 2\times3 + 3\times2 = 17$

Qu

Keys	20	30	40	50
frequency	4	2	4	2

$$dp[i,j] = \min_{i<k<j} \{[dp[i,k-1] + dp[k+1,j]] + sum[i,j]\}$$

	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	4			
i=2	0		2		
i=3	0			4	
i=4	0				2

Key: 20, 30, Root : 20, k=1

$dp[1,2] = \min_{1\le k<2} \{dp[1,0] + dp[2,2] + Sum[1,2]$

$0 + 2 + 6 = 8$

Root : 30, k=2

$dp[1,2] = \min_{1\le k<2} \{dp[1,1] + [3,2] + sum[1,2]$

$4 + 0 + 6 = 10$

Sum	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	4	6	10	12
i=2	0		2	6	8
i=3	0			4	6
i=4	0				2

→ Key: 20, 40

[2,3] K=2 K=3

i j

$dp[2,3] = \min_{2\le k<3} \{dp[2,1] + dp[3,3] + Sum[2,3]$

$= 0 + 6 + 4 = 6 + 0 + 4 = 10$

$6 + 0 + = 2 = 8$

$dp[2,3] = \min_{2 \le 3 < 3} dp[2,2] + dp[3,3] + sum[2,3]$

$= 0 + 4 + 6 = 10$

→ Key: 40, 50

[3, 4] $K = 3$

$K = 4$

$dp[3,4] = \min_{3 \le 3 < 4} \{dp[3,2] + dp[4,4] + sum[3,4]$

$= 0 + 2 + 6 = 8$

$dp[3,4] = \min_{3 \le 4 < 4} \{dp[3,3] + dp[5,4] + sum[3,4]$

$= 0 + \quad + 6 =$

Q. Keys
Fract

$dp[i,j$

$dp[$

$dp[1,2]$

$dp[1,2] =$

$dp[2,3] = m$

$=$

$dp[$

$dp[1,3$

Q. Keys 5 15 25
freq 2 1 4

$$dp[i,j] = \min_{i \le k < j} \{dp[i,k] + dp[k+1,j] + sum[i,j]\}$$
$$i \le k \le j$$

$$dp[1,1] = \min_{1 \le 1 \le 1} \{dp[1,0] + dp[2,1] + sum[1,1]\} = 2$$

$$dp[1,2] = \min_{1 \le 2 \le 2} \{dp[1,1] + dp[3,2] + sum[1,2]\} = 2+0+3 = 5$$

$$dp[1,2] = \min_{1 \le 1 \le 2} \{dp[1,0] + dp[2,2] + sum[1,2]\} = 0+1+3 = 4$$

$$dp[2,3] = \min_{1 \le 2 \le 2} \{dp[2,1] + dp[3,3] + sum[2,3]\} = 0+4+5 = 9$$

$$dp[2,3] = \min_{2 \le 3 \le 3} \{dp[2,2] + dp[4,3] + sum[2,3]\} = 1+0+5 = 6$$

sum	j=0	j=1	j=2	j=3
i=0	0	0	0	0
i=1	0	2	3	7
i=2	0		1	5
i=3	0			4

	j=0	j=1	j=2	j=3
i=1		2	4	11
i=2			1	6
i=3				4

$$dp[1,3] = \min_{1 \le k \le 3} \begin{cases} dp[1,0] + dp[2,3] + sum[1,3] = 0+6+7 = 13 \\ dp[1,1] + dp[3,3] + sum[1,3] = 2+4+7 = 13 \\ dp[1,2] + dp[4,3] + sum[1,3] = 4+0+7 = 11 \end{cases}$$

(vi) DEFAULT :-
It ensures that a default value is used when a value is not provided

Unique constraint (Error)
violation of UNIQUE key constraints
'UQ - Employee AD...';
Can't Insert duplicate key in object

CHECK constraint - (Error)
The Insert statement conflicted with CHECK constraint "CK - Product - Price".

NOT_NULL constraint :- (Error)
Can't insert the value NULL into column 'Name'

Key :- A Key is a field or set field in a database table that uniquely identifies each record in the table.
Keys are essential for maintaing data integrity and stablishing relationships b/w tables.

Primary Key :-
It crucial as a uniquely identify each record in a table that ensures no duplicate values.
It is often referenced by foreign key in other tables.

Table - Employee

Emp ID(PK)	Emp Name	Salary
E100	A	50000
E200	B	70000

Foreign key :- It is important for stablish relationship b/w tables.
A foreign key in one table refers to the primary key in another that enforce referencial integrity and linking data across tables.

① Ex :- Foreign key

Table - Department

Dep ID	Dep_Name
D1000	marketing
D2000	Financial

Table - Employee

Emp ID (PK)	Dep ID	Emp Name
E100	D1000	A
E200	D2000	B

② Ex :- [Foreign Key]

```
CREATE TABLE Student (
    StudentID INT Primary key,
    Name VARCHAR (50);
    Age  INT;
    Mark  INT;
);
INSERT INTO Student VALUES
(1, 'Alice', 20, 85);
(2, 'Bob', 18, 76);
(3, 'Charlie', 22, 92);
```

```
SELECT * FROM Student;

CREATE TABLE Courses (
    CourseID INT Primary Key,
    CourseName VARCHAR(50),
    StudentID INT,
[FOREIGN KEY (StudentID) REFERENCES
    (Student ID));

INSERT INTO Courses VALUES
    (101, 'Math', 1);
    (102, 'Science', 2);
    (103, 'History', 3);
SELECT * FROM Courses;
```

output -

Student ID	Name
1	Alice
2	Bob
3	Charlie

Course ID	Course_Name
101	Math
102	Science
103	History

Candidate Key :- A Is a set of columns that can uniquely indentifying a record and once is selected as a primary key knowing what candinati key are helps in understanding the possible choices for the primary key.

Ex :- Table - Employee

Emp ID (PK)	Dep_ID	Emp_Name	Aadhaar_No
E100	D1000	A	123456789123
E200	D2000	B	321678912312

Super key - Is any set of attribute that can uniquely identify a record in a table.
Super key in this table.
Any of these ~~are~~ can be super key

(I) Student_ID
(II) Aadhaar_Number
(III) (Student ID, Name)
(IV) (Aadhaar_Number, Age)
(V) (Student ID, Aadhaar Number).

All are uniquely identify a record

Note :-

Primary key ≤ Super key
Candidate key ≤ Super key
Super key ≥ Candidate key

Basic attention before creating key :-

(I) Uniqueness
(II) NOT_NULL
(III) Stability
(IV) Minimal attributes
(V) Avoid sensitive data

straint:
ensures that a default
used when a value is

contain

① 1987 - ISO standard : SQL approved by ISO

② Later versions : SQL - 89, 92, 99, 2003, 2008, 2011, 2016

* what is SQL injection ?

It is a cyber attack that involves tricking a DB with SQL queries. Hackers used SQL injection to retrive, modify or corrupt data in a SQL DB.

ex - They might fill in a SQL query instead of name of the person in a submission for to carry out a SQL injection attack.

⇒ Command, Keyword and clause -

ⓐ Command :- A command refers to an action or operation that you instruct database perform or manipulate DB objects for example ~~select~~ ~~insert~~ and etc.
SELECT INSERT

ⓑ Keyword :- It refers to reserved words with pre define meanings and functionality
ex - SELECT, ~~INSERT~~ etc.
INSERT

ⓒ Clause - It refers to a component of an SQL statement that ~~of~~ provides additional instructions or condition to the DBMS.
ex :- ~~Set~~ SELECT, FROM WHERE, GROUP BY, HAVING, ~~ORDER~~ ORDER BY, LIMIT, JOIN WITH

~~WITH~~ :- ~~It~~ creates

→ Basic SQL operators —

WITH clause :- It creates temporary result. It is also known as comman table expression (CTE)

→ AS - Is a keyword that is used to give alias columns or tables temporarily

→ CHAR :- (i) Fixed length
(ii) Always uses set space
CHAR(10) Always 10 chars
(iii) Good for codes.

→ VARCHAR — (i) variable length
(ii) uses only needed good for articals and comments.

→ TEXT :- (i) large text type
(ii) can hold long string
(iii) good for articals and comments.

Quaries and sub quaries :-

These are powerfull tools for extract expecting usedful information from DB by speciping condition and relationship b/w data and Elements.

Query :- A query is a request for information from a DB.

Ex:- SELECT firstly FirstName, LASTName

Date

ex - SELE
Em

sub quer
ne
is used
ex:- SE
WHERE
FROM

Q. Write

So

(i) Cas

(ii) fin

(iii) Fi

(iv) Fi

(v)

ex- SELECT FIRSTName, LastName FROM Employee WHERE Department 'Sales';

Sub queries :- It is also known as nested query or inner query that is used within another query.

ex:- SELECT FirstName, LastName FROM Employee WHERE Department IN (SELECT Department FROM Departments WHERE Location = 'New York');

Q. Write SQL program of following table

Sale ID	Product	Price	Quantity
1	Pen	10	50
2	Notebook	50	30
3	Eraser	5	100

(i) Calculate the total Quantity of product sold
(ii) Find the maximum price among all product
(iii) Find the minimum price among all product
(iv) Find the average price among all product
(v) Find the quantity of Eraser.

```
CREATE TABLE Ace_Product (
    SaleId INT, PRIMARY KEY,
    Product varchar (10),
    Price INT,
    Quantity INT );
INSERT INTO Ace_Product (SaleID,
Product, price, Quantity ) VALUES
    (1, 'Pen', 10, 50),
    (2, 'Notebook', 50, 30),
    (2,
```