**DYNAMIC PROGRAMMING :**

Power full technique used to solve pro that have been overlapping sub problem . and optimal substructure .

Instance of solving same problem multiple time DP store the sub problem solution and reuse them making algorithm more efficiently .

**Key concept :**

1. Overlapping sub problem

A problem is said to have overlapping sub problem in this same sub problem are solve multiple time

**ex :** Compute fibonacci series in recursively .

Optimal substructure .

**Two approach :**

**Top down approach** -> menoization(recursion + cache).

**Bottom up approach :**

uses the iteration and build the solution from the smallest sub problem to the main problem .

stores the result in an array and avoid the recursion override

**Problem 1 :**

package day23;

import java.util.Arrays;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class task2 {

static Map<Integer , Integer> *map* = new HashMap();

public static int fibonacci(int i)

{

if(i<=1)

{return i;}

if(*map*.containsKey(i))

{return *map*.get(i);}

int res=*fibonacci*(i-1)+*fibonacci*(i-2);

*map*.put(i,res);

return res;

}

public static void main(String args[])

{

System.***out***.println(*fibonacci*(10));

System.***out***.println(*fibonacci*(2));

System.***out***.println(*fibonacci*(1));

}

}

**problem 2 :**

grid traveler on a 2 d grid begins is works from top left corner and your goal is to travel to the bottom right corner you may only move down or right in how many ways you can travel on a grid with dimension M\*N

the above problem is dynamic because

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

2 \* 3

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

3 \* 3

* 2\*3

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

3 \* 3

* 2\*2

|  |  |  |
| --- | --- | --- |
| des |  |  |
|  |  |  |
|  |  | src |

|  |  |  |
| --- | --- | --- |
| 1 | 1 | 1 |
| 1 | 2 | 3 |
| 1 | 3 | 6 |

package day23;

import java.util.Arrays;

import java.util.Scanner;

public class task3 {

public static void main(String args[])

{

Scanner s = new Scanner(System.***in***);

int r=s.nextInt();

int c = s.nextInt();

int arr[][]= new int[r][c];

for(int i=0;i<r;i++)

{

arr[i][0]=1;

}

for(int j=0;j<c;j++)

{

arr[0][j]=1;

}

for(int i=1;i<r;i++)

{

for(int j=1;j<c;j++)

{

arr[i][j]=arr[i][j-1]+arr[i-1][j];

}}

for(int i=0;i<c;i++)

{

for(int j=0;j<c;j++)

{

System.***out***.print(arr[i][j]+ " ");

}

System.***out***.println();

}

}

}

3 row

3 col

1 1 1

1 2 3

1 3 6

2

3

1 1 1

1 2 3

**Problem 3:**

Given a set of number and tar sum to achieve write to the pro to return the Boolean value if the tar is achieve or not using the numbers in the array .

* Knapsack problem
* Coin change

**Coin change :**

package day23;

import java.util.Arrays;

import java.util.Scanner;

public class task6 {

public static int countWay(int coin[],int amount)

{

int dp[]=new int[amount+1];

dp[0]=1;

for(int c:coin)

{

for(int i=c; i<=amount;i++)

{

dp[i]+=dp[i-c];

}

}

System.***out***.println("dp table :"+Arrays.*toString*(dp));

return dp[amount];

}

public static void main(String args[])

{

int coin[] = {2,3,7};

int amount=7;

System.***out***.println(*countWay*(coin,amount));

}

}

dp table :[1, 0, 1, 1, 1, 1, 2, 2]

2