Dynamic Programming

1. Fibonacci Series

1	2	3	4	5	6	7	8
1	1	2	3	5	8	13	21 . .

Iterative nth_fib(n)

```
$ i=0, a=1, b=1
while
if n=1 or n==2:
    Pf(1)
else: i=3
    while (i<=n):
        { c=a+b;
          a=b;
          b=c;
        i++;
        print(c);
```

O(n)

Bottom-Up

Recursive

```
int fib (int n)
{ arr=[n+1];
  if n==1 || n==2:
      ret 1;
  else:
      return fib[n-1]+fib[n-2];
}
```

O(2^n)

dp: int fib(n) — Top Down

```
{ int a [n+1];
  a[0]=0, a[1]=a[2]=1;
i=3; while(i<=n) c=a+b; a=b; b=c;
    { a[i]= a[i-1]+a[i-2];
      i++;
    }
ret a[n];
}
```

Bottom Up

Tabulation

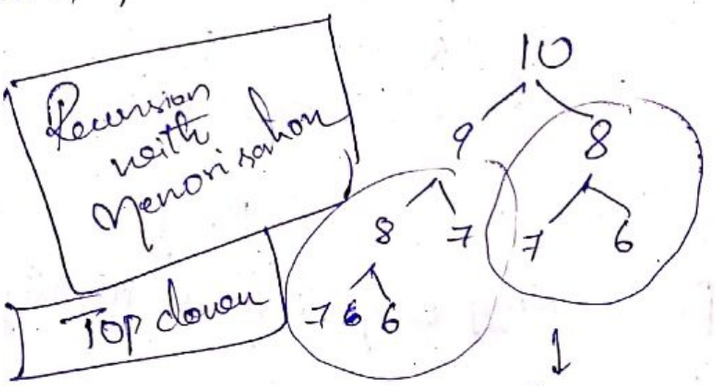
O(n), O(n)

Recursion with Memorisation

O(n), O(1)

DP Recursive (Time, Space) = O(N, N).

```
ar= [-1]*(n+1)
fib(n):
  if ar[n] != -1:
      return ar[n]
  else:
      k = fib(n-1) + fib(n-2)
      ar[n]=k;
      return a[n]
```



for → Overlapping Sublems o use Dynamic Programming

1. Overlapping Subproblems.
2. a[i] ⟶ ith Fibonacci number ⟶ DP state
3. a[i] = a[i-1] + a[i-2] ⟶ DP Expression
4. Tabulation Base Condition
5. Table Size
6. Time & Space Complexity

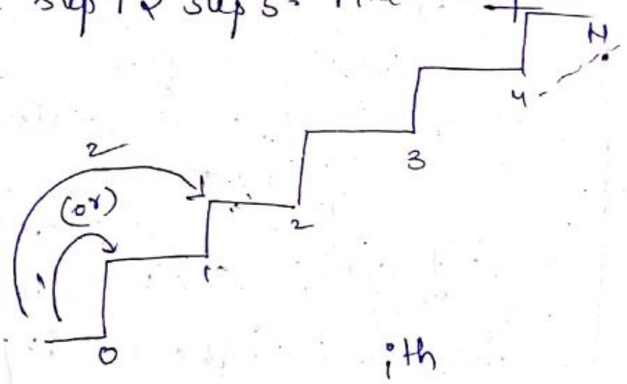
Time Complexity :-

TC = No. of states × Time taken for each state

7. find the answer location in Table
8. Space Optimization

---

* Given a staircase of size N. You can step up 1 step / 2 steps. find no. of ways to step nth step

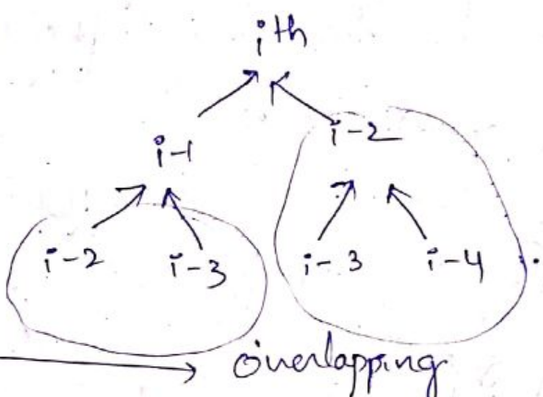


if N = 5

(1 1 1 1 1)
(2 1 2)
(2 2 1)
(1 1 1 2)
(1 1 2 1)
(1 2 1 1)
(2 1 1 1)

if N = 4

(1 1 1 1)
(2 2)
(1 1 2)
(1 2 1)
(2 1 1)



1. ⟶ overlapping
2. dp[i] = No. of ways to reach ith step ⟶ dp state
3. dp[i] = dp[i-1] + dp[i-2] ⟶ dp exp
4. dp[0] = 1
   dp[1] = 1 → Base Conditions
5. (N+1) ⟶ Table size.

6. Time O(N) ⟶ Time complexity

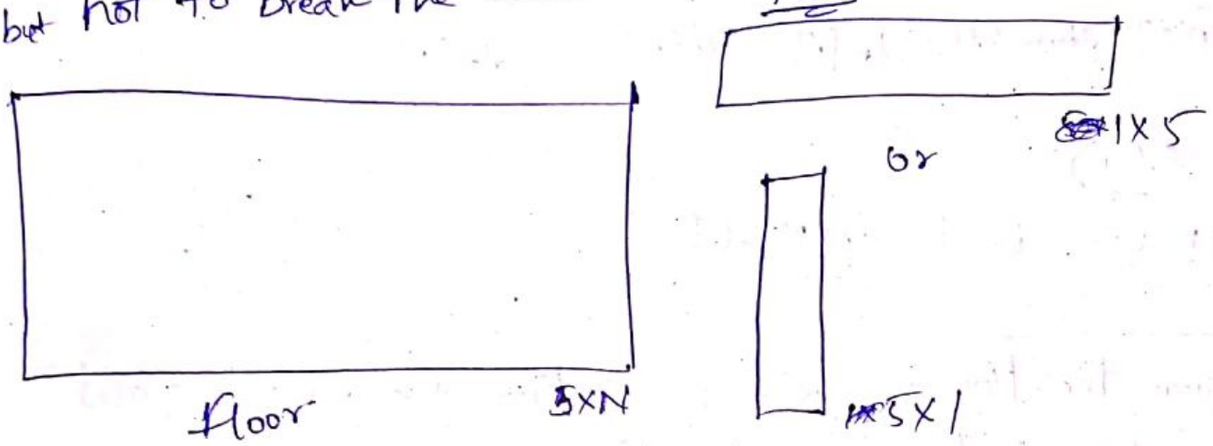
7. O(N)/O(1) → space complexity

7. return dp[N] ⟶ ans.

8 O(1) ⟶ If space is optimized.

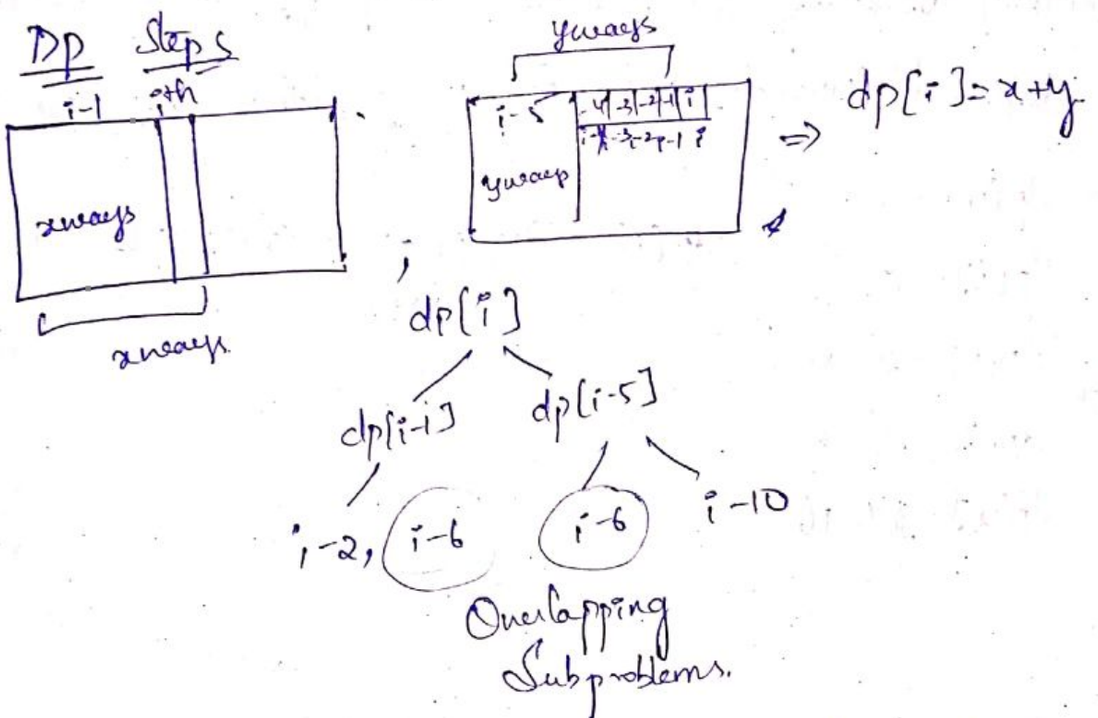
---

* fill the floor using the tiles either horizontally / vertically but not to break the tile. find no. of ways to cover the floor.



Sol: DP Steps

1.



2 dp[i] = No. ways to fill floor of size 5xi. using 1x5, 5x1 tiles

3 dp Expression ∀ i=5 dp[i] = dp[i-1] + dp[i-5]

4. Base Conditions: dp[0] = dp[1] = dp[2] = dp[3] = dp[4] = 1

5 Table Size: (N+1) x 1 = N

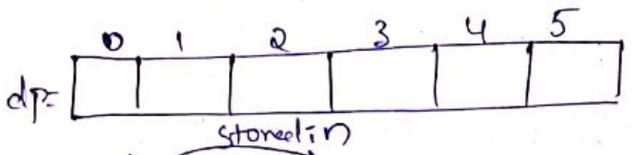
6 Time compl: (N+1) x 1 = N , Space complexity = O(N)

∴ dp[n] is answer

⇒ Space Optimization using Mod.

Just o[5+1] = O(6)

$\forall_{i=5}^{n}$ dp[i%6] = dp[(i-1)%6] + dp[(i-5)%5].



dp[6] ⇒ 6%6 = dp[0].

dp[7] stored in dp[1]

dp[6] → i-1 → dp[5]; dp[6] → i-5 → dp[1]

⇒ O(1)

Answer is at dp[N%6].

---

* Given the floor of size 5×1 and tiles of size should be placed horizontally / vertically and also the sides of the tiles are painted with different colors. Find no of ways filling the floor

Sol:- dp[0] = 0

dp[1] = 2

Base conditions dp[2] = 2^2 = 4

dp[3] = 2^3 = 8

dp[4] = 2^4 = 16.

dp[i] = 2^1 × dp[i-1] + 2^5 × dp[i-5].

(or)

* Given a length of binary string, generate the binary strings which donot have adjacent 1's in them.

N=3 ⇒ cnt = 5

Sol:

0	000
1	001
2	010
~~3~~	~~011~~
4	100
5	101
~~6~~	~~110~~
~~7~~	~~111~~

N=1 cnt=2

0
1

N=2 ⇒ cnt 3.

00
01
10
~~11~~

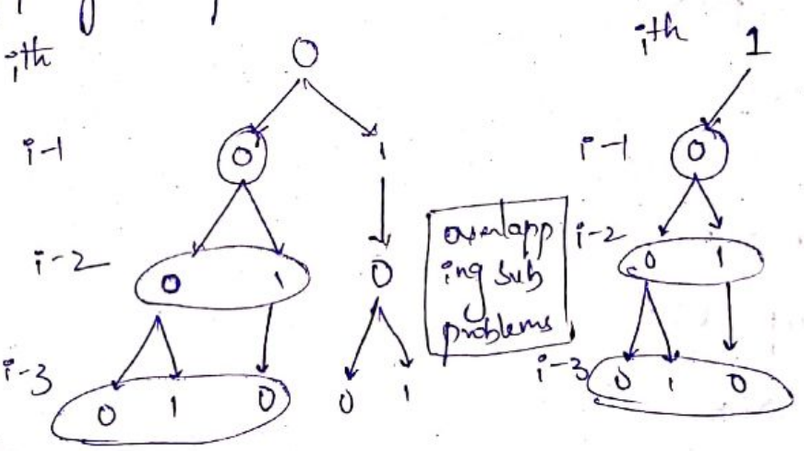
N=4 cnt = 3+5 = 8.

8	1000
9	1001
10	1010
~~11~~	~~1011~~
~~12~~	~~1100~~
~~13~~	~~1101~~
~~14~~	~~1110~~
~~15~~	~~1111~~

N	1	2	3	4	5	
cnt	2	3	5	8	13	⟶ fibonacci series

Sol: Nth fibonacci using DP → N[1], N[2] = 2, 3 - - - - -

1. Overlapping Subproblems:-



2. Dp State:

dp0[i] ⇒ # No. of binary strings of length i having no adjacent 1's ending with 0 (ith value = 0).

dp1[i] ⇒ # ending with 1 (ith value = 1).

3. DP Expression:

$$\forall_{i=2}^{n} \quad dp0[i] = dp0[i-1] + dp1[i-1]$$

$$dp1[i] = dp0[i-1]$$

4. Base Condition:

dp0[1] = 1

dp1[1] = 1

5. Table Size:-

int dp0[n+1]; } Space : O(n)
int dp1[n+1]; } Time:

6. Time complexity:-

T.C = No. of states × Time for each state

= O(N+1) × O(1) = O(N)

7. Answer:-

ans = dp0[n] + dp1[n]

8. Space Optimisation:-

prev0 = prev1 = 1

curr0 = prev0 + prev1

curr1 = prev0

prev0 = curr0;

prev1 = curr1;

ans = curr0 + curr1

Space comp = O(1)

Time Comp = O(n)

* Snake and ladder : No. ways to get a sum

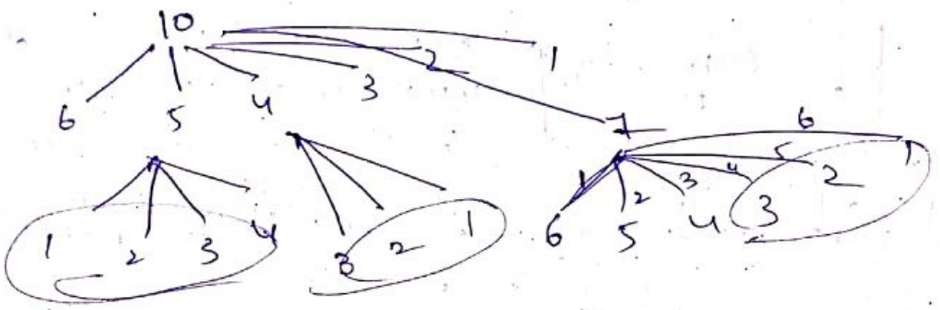
(N)

Sum = 10

- 6 4
- 5 5
- 4 6
- 1 1 1 . . . . 10 times.
- 1, 2 . . . . — —
- 3 3 3 31 . . . .

Dice

1. Overlapping Subproblems:



2. DP states

dp[i] = # ways to get sum = i

3. DP Expression:

$$dp[i] = \sum_{j=1}^{6} dp[i-j] \quad \text{and } i >= j$$

dp[i] = dp[i-1] + dp[i-2] + dp[i-3] + ... dp[i-6]

4. Base conditions:

only if no need

- dp[0] = 1
- dp[1] = 1
- dp[2] = 2
- dp[3] = 4 (1,1,1), (1,2) (2,1) (3)

5. Table size

int dp[n+1] ⟹ O(n)

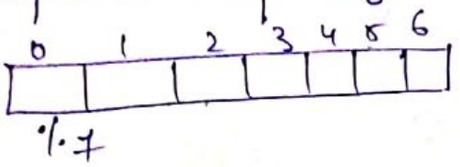
6. Time complexity:

N × 1 ⟹ O(N)

7. Answer

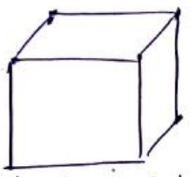
ans = dp[n]

8. Space Comm Optimization:



---

* for each number on the dice, cost is associated. find the minimum cost required to get the given sum



	0	1	2	3	4	5	6
cost	X	2	4	9	15	12	6

sum = 10 (N)

- 5,5 cost ⇒ 12+12 = 24
- 4,4,2 cost ⇒ 15+15+4 = 34
- 6,4 cost ⇒ 6+15 = (21) minm cost

1. Overlapping subproblems
2. Optimal Substructure property will be included if for the problem wants the optimisation. (Minimisation / Maximization)
3. DP state

   Dp[i] = # Minimum cost required to get sum = i
4. DP Expression:

$$dp[i] = \min_{\substack{j=1 \\ i-j>=0}}^{6} (dp[i-j] + c[j])$$

1. Overlapping Subproblems
2. Optimal Substructure
3. DP state
4. DP Expression
5. Base Condition
6. Table Size
7. Answer Time & Space Complexities
8. Answer
9. Space Optimisation.

* N = 1 2 3 4 5 6

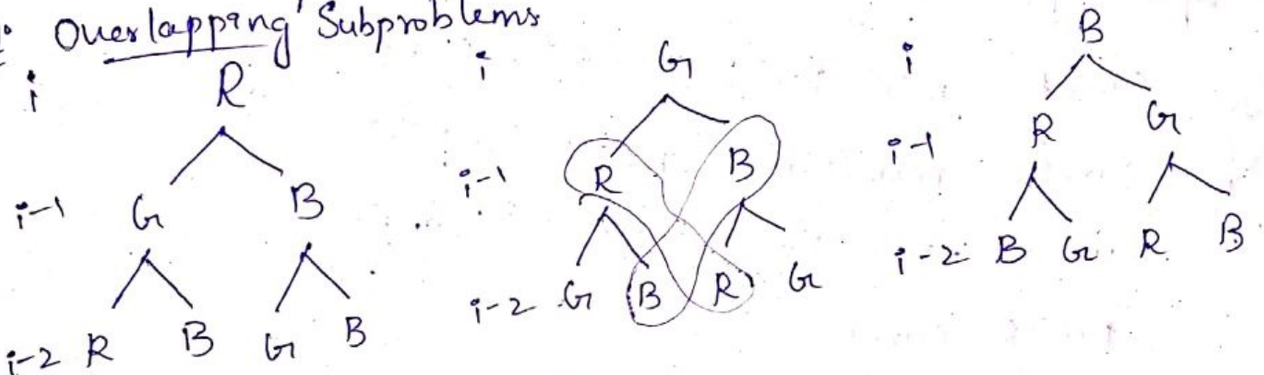
R : 5 8 3 10 2 15

G : 9 4 1 3 6 2

B : 12 5 2 8 7 5

Given N houses, find the minimum cost required to paint all the houses such that no 2 adjacent houses should be painted with the same color.

1. Overlapping Subproblems



2. Has Optimal Substructure

3. DP state:

~~dp[i] = # Minimum cost required to paint i houses such that no 2 houses have same colors~~

dpr[i] = # Minimum cost to paint i^{th} house with red such that no houses have same color

dpg[i] = i^{th} house with green

dpb[i] = i^{th} house with blue

4. DP Exp:

$dpr[i] = \min(dpg[i-1], dpb[i-1]) + R[i]$

$dpg[i] = \min(dpr[i-1], dpb[i-1]) + G[i]$

$dpb[i] = \min(dpr[i-1], dpg[i-1]) + B[i]$.

for $i = 1$

5. Base Cond

$$Dpr[0] = Dpg[0] = Dpb[0] = 0.$$

6. Table size $(n+1)$

$SC = O(n)$

7. Time complexity $= O(n)$

8. Answer $= \min(dpr[n], dpg[n], dpb[n])$

9. Space Opt

$$\text{Curr}R = \min(\text{prev}B, \text{prev}G) + R[i]$$

$$\text{curr}B = \min(\text{prev}R, \text{prev}G) + B[i]$$

$$\text{Curr}G = \min(\text{prev}R, \text{prev}B) + G[i].$$

$\forall\ i \geq 1$

$\text{prev}R = \text{Curr}R$

$\text{prev}B = \text{Curr}B$

$\text{prev}G = \text{Curr}G.$

$Ans = \min(\text{Curr}R, \text{Curr}B, \text{Curr}G).$

$Time = O(N)$

$space = O(1)$

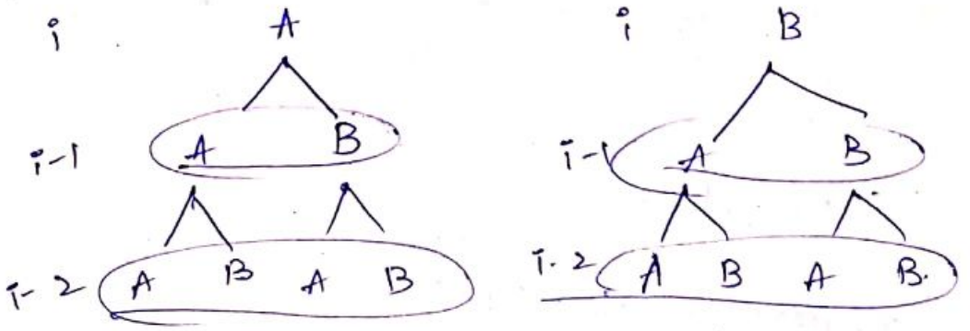
---

* Given 2 machines A, B and N tasks. find minimum cost required to finish N tasks.

N	1	2	3	4	5	6
A	5	10	3	8	12	18
B	2	9	4	5	13	8

$t = \text{latency} = 10$

1. Overlapping



2 Optimal Substructure ✓

3. DP states: (1 to i)

DPA[i] = # Minimum cost required to complete i tasks with ith task on machine A.

DPB[i] = # 1 to i tasks On B (ith task).

4. DP Expression:

DPA[i] = min((DPA[i-1], DPB[i-1] + t)) + A[i]

DPB[i] = min(DPB[i-1], DPA[i-1] + t) + B[i]

5. Base cond:

dpA[0] = dpB[0] = 0

6. Table size: O(N)

7. Time comp: O(N)

8. Answer: Min(DPA[n], DPB[n])

9. Space Optimisation ✓ Yes.

* Find Maximum Subarray Sum.

	0	1	2	3	4	5	6	7	8	9	10
ar[N]:	3	-1	8	-12	10	-4	3	9	-2	8	-40

Solutions:-

1. $N^3 \times N, 1$ (Brute force).

2. $N^2, 1$ (carryforward technique)

3. Prefix Sum: (carry fwd) Don't consider if sum is -ve

	3	-1	8	-12	10	-4	3	9	-2	8	-40
Psum	3	2	10	-2	10	6	9	18	16	24	-16

don't carry

DP State:

dp[i] = Maximum subarray sum till ith idx

DP Exp:

$\forall_{i=1}$ dp[i] = Max(dp[i-1] + ar[i], ar[i])

Base cond:

dp[0] = ar[0]

Table Size: (n)

Sp = O(n).

Time compl:

n x O(1) = O(n).

Ans = dp[n-1]. Max(dp[i])

Space Opt:

$\forall_{i=1}$ [curr = max(curr + a[i], a[i])

ans = max(ans, curr)]

Answer = ans.

Sp = O(1), Time = O(n)

	0	1	2	3	4	5	6	7	8	9	10	11	12
Ar:	3	5	2	8	-3	18	4	10	6	12	-5	2	4
DP:	3	5	5	13	2	31							

Find Maximum Subsequence Sum. such that no 2 elements are adjacent

Solutions:-

1. 2^N

2. DP

½ Overlapping sub problems X 2

3. DP state

DP[i] = Max subsequence sum till i including(i) such that no 2 eles are adjacent

4. DP Exp

$$\forall_{i=2}^{n-1} dp[i] = \forall_{j=i-2}^{0} Max(dp[j] + a[i])$$

$$\forall_{i=2}^{n+1} dp[i] = m + ar[i].$$

$$m = max(m, dp[i-1])$$

5. Base cond:

$dp[0] = Ar[0]$

$dp[1] = Ar[1]$

$dp[0] = ar[0]$

$dp[1] = ar[1]$

$m = ar[0]$.

6. Table size:- n

$O(n)$.

n

7. Time complexity => $N \times N-2$.

$O(n^2)$

$n \times 1 = O(n)$

8. Answer:- Max (dp)

Max(dp)

9. Space Optimisation x not possible

n-1 CP = m + a[i]

$\forall_{i=2}$ ~~m = max(m, p)~~

$m = max(p, m)$.

p = c

Base: P = ar[1], m = ar[0]

10. Sol $O(n^2)$, $O(n)$

3 sol $O(n)$, $O(1)$.

4 SOl

* Same as above ~~problem~~ / Solution but including or excluding?

Arr	3	5	2	8	-3	18	4	10	6	12	-5	2	4
dp:	3	5	5	13									

- max(ar[i])
- (dp[i-1])
- dp[i-2] + a[i]

$\forall_{i=2}^{n-1}$ dp[i] = max(ar[i], dp[i-1] , dp[i-2]+a[i])

dp[0] = a[0]

dp[1] = Max(a[0], a[1]).

Time
→ O(n) x1

→ Space: O(1)

---

* Longest Increasing Subsequence

	0	1	2	3	4	5	6	7	8	9	10	11
Arr:	10	3	12	7	2	9	15	20	11	13	6	8
LIS (length)	1	(1)	2	(2)	1	(3)	4	5	(4)	(5)	2	3

Ans: Sequence 13 11 9 7 3

DP State:-

dp[i] : # length of longest inc Subsequence till i including i

DP Exp:

dp[i] = $\forall_{j=0 \,\&\, a[j]<a[i]}^{i-1}$ max (dp[j]) + 1

Base Conditions:-

dp[0] = 1 X (No need to handle Base Conditions)

Table Size:

n => Space Comp = O(n)

Time Comp:

$n \times n = n^2 \Rightarrow O(n^2)$

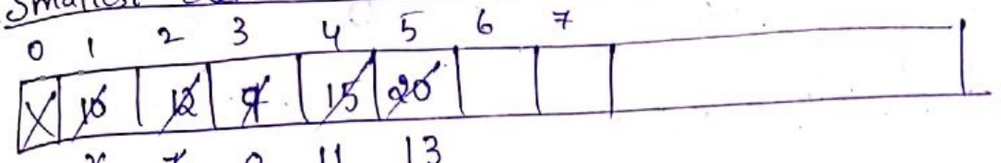
Answer:

Max(dp)

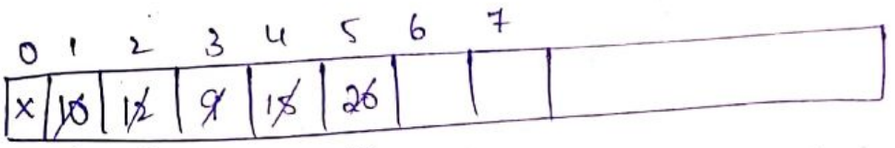
Space Optimisation:

No

Smallest Element of LIS ending at i



*

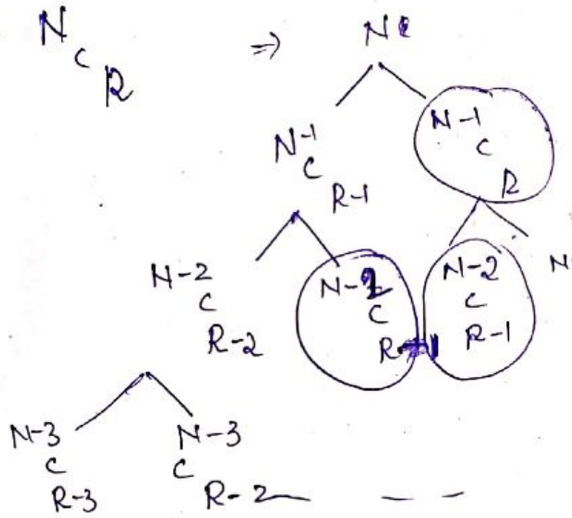


Sorted ⇒ This is not the

$O(N \times \log n)$ (search) , $O(N)$

---

* ${}^{N}C_{R} \Rightarrow$



$\frac{(N-1)!}{(N-1-R+1)! \times (R-1)!} + \frac{(N-1)!}{(N-R-1)! \times R!}$

$\frac{(N-1)!}{{}^{N-2}C_{R}\,(N-R)! \times (R-1)!} + \frac{(N-1)!}{R \times (R-1)! \times (N-R-1)!}$

$\frac{(N-1)!}{(N-R)(N-R-1)! \times (R-1)!} + \frac{(N-1)!}{R(R-1)! \times (N-R-1)!}$

$\frac{R(N-1)! + (N-R)(N-1)!}{(N-R)! \times R!}$

→ Overlapping Exists

→ Optimal Substructure

$${}^{N}C_{R} = \frac{(N-1)!\,(N-R+R)}{} \rightarrow \frac{N!}{}$$

dp state

dp[i][j] = # no. of ways to choose j items from i items

DP Exp: $\forall_{i=0}^{N} \forall_{j=0}^{R} dp[i][j] = dp[i-1][j-1] + dp[i-1][j]$

$({}^{N-1}C_{R-1} + {}^{N-1}C_{R})$

i => N
j => R

	0	1	2	- -	R
0	1	0	0	0	0
1	1				
:	1				
N	1				

$(N+1) \times (R+1)$

0th Col: ${}^{0}C_{0} = {}^{1}C_{0} = {}^{N}C_{0} = 1$

0th Row: ${}^{0}C_{0} = 1$, ${}^{0}C_{1} = {}^{0}C_{2} = \cdots = 0$

Base cond:

$\forall_{i=0}^{N} dp[i][0] = 1$ $\quad$ $\forall_{j=1}^{R} dp[0][j] = 0$

Table size:-

$O((n+1) \times (R+1))$

⇓ Sp comp

Time comp:-

$O((n+1) \times (R+1)) * 1$

Answer:

dp[N][R]

Space opt:- (No need of complete matrix)

2 Rows => of size R

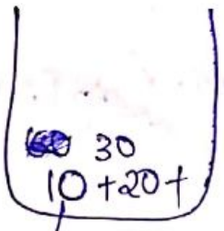
$(2 \times R+1)$

Fractional knapsack (K=50)

ls = (u, w)

ls = (6 10) (5, 20) (4, 30)

w[i] =	20	10	30
v[i] =	100	60	120
u = v/w =	(5)	(6)	(4)
	1	1	2/3



P = 100 + 60 + $\frac{2}{3}$(40)

= 240

Time: O(N + N log$_2$N + N).

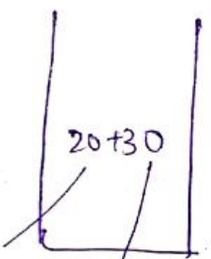
↓ find v/w ↓ sort v/w ↓ find ans

Space: O(N)

```
for u, w in ls:
    if K>0:
    units = min(w, K)
    ans += u * units
    K -= units
```

0-1 knapsack:- (K=50)

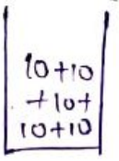
w[i]	20	10	30
v[i]	(100)	60	(120)



100 + 120 = 220

Infinite K=50

weights are infinite



⇒ 5 × 60 = 300

Sol 1: Overlapping

2 Optimal Substructure (Maximizing profit)

3 DP state: items, weights

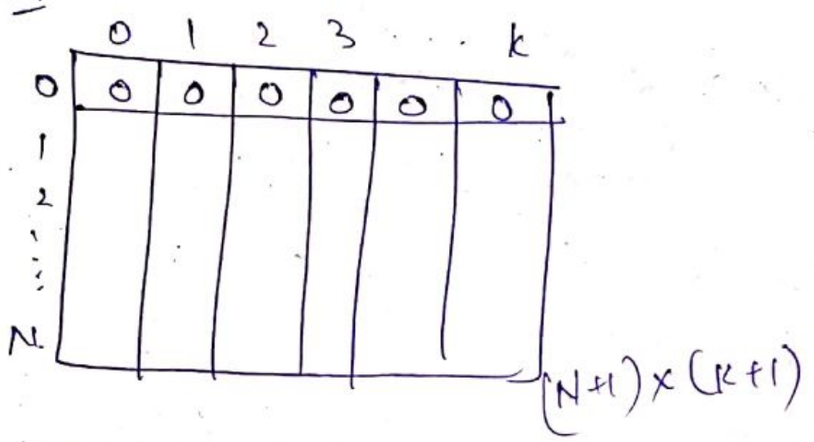
dp[i, j] = Maximizing the profit by filling knapsack of weight j by using first i items.

4 Dp Exp

$$\forall_{i=1}^{n} \forall_{j=0}^{K} \; dp[i][j] = \max(dp[i-1][j-w[i]] + v[i],\ dp[i-1][j])$$

$j - w[i] \geq 0$

5 B.C



6. Time Comp Table Size

$$(N+1)(K+1)$$

$$Sp = (NK)$$

7 Time

$$(N+1)(K+1) \times 1 = O(NK)$$

8 Ans: $dp[n][k]$

9. Space opts -

2 Rows enough $\Rightarrow 2 \times (K+1) = 2K$ space.

* for infinite knapsack.

$$\forall_{i=1}^{N} \forall_{j=0}^{K} \; dp[i][j] = \max(dp[i][j-w[i]] + v[i],\ dp[i-1][j$$

$j - w[i] \geq 0$.

K = 7

	1	2	3	4	5
w[i]	3	6	5	2	4
v[i]	12	20	15	6	10

items \ weight	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	12	12	12	12	12
2	0	0	0	12	12	12	20	20
3	0	0	0	12	12	15	20	20
4	0	0	6	12	12	18	20	20
5	0	0	6	12	12	18	20	22

22 - 10 = 12
(4) + (3) = 7

Ar_N:	0	1	2	3	4	5	6	7	8	9
	10	3	8	15	9	12	6	4	1	2

K = 14 ⇒ (10, 4)
(9, 1, 4)
(12, 2)
(8, 6)

Coin change Problem. ① It is possible to change.
② No. of ways to make the change
③ Min no. of elements required.

① Y/N:

Dp state: # Is it possible to get sum of j using first i elements. (bool)

Dp Exp

dp[i][j] = dp[i-1][j - a[i]] || dp[i-1][j]

(True // False.)

② No. of ways.

dp state = # No. ways to get sum j from first i elements

dp[i][j] = dp[i-1][j - a[i]] + dp[i-1][j]

3. Minimum Ele:

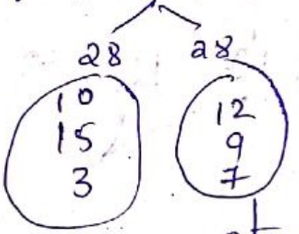
dp[i,j] = min ele to get sum j from 1st i elements.

$dp[i,j] = \min(dp[i-1][j-a[i]], dp[i-1][j]) + 1$

---

* Given an array, Divide the array into sets such that the difference between b/w their sums is minimum.

Ar_N : 10 12 3 15 9 7 = 56



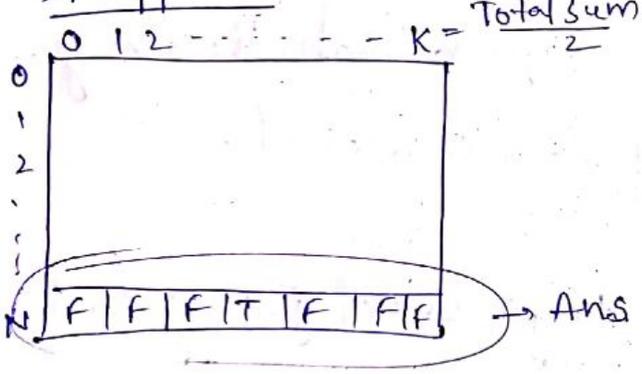
Solutions:

1. $\frac{\text{Total sum}}{2}$

Find the subsets by recursion or iteration.

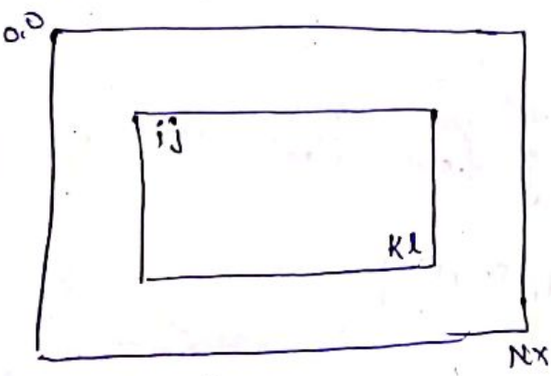
$O(\frac{\text{Total sum}}{2} \times 2^n)$

2. DP Approach:-



---

* Given a matrix and given some queries Q: i,j, k,l which are coordinates of top left and bottom corner



Q → i, j, k, l

$0 \le i,j \le n$

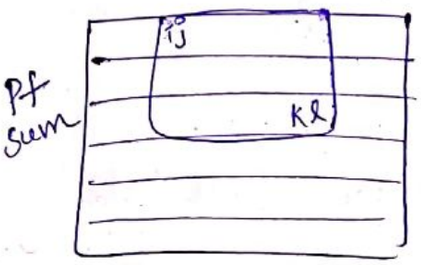
$0 \le k,l \le m$.

Solutions:-

1.
```
S = 0
for x = i to K:
    for y = j to l:
        S += mat[x][y].
```

Time comp: O(Q × N × M)

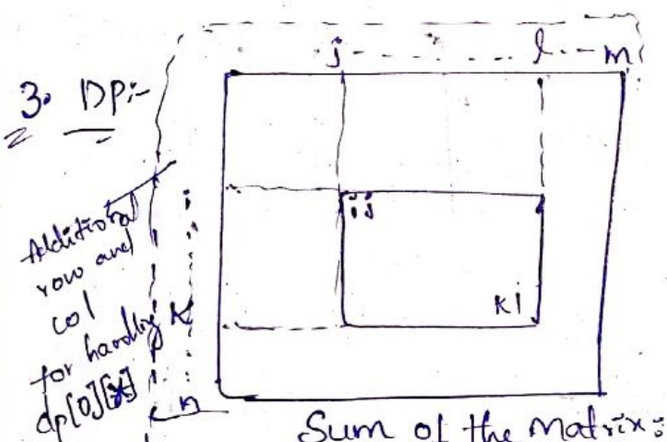
2. Prefix Sum.



NxM + QxN.
Pf sum

```
for x in range(i, K):
    S += pf[x][l] - pf[x][j]
```

3. DP:-



dp[i][j] = Σ

Sum of the Matrix:

dp state:

$$\sum_{x=0}^{i} \sum_{y=0}^{j} mat[x][y].$$ // sum from 0,0 to i,j

dp Exp:-

dp[i][j] = dp[k,l] = dp[k, j-1] - dp[i-1, l] + dp[i-1, j-1]

i) Sum of the Matrix:
(Pf sum)

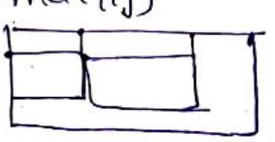
1	2	3	4
5	2	4	3

→

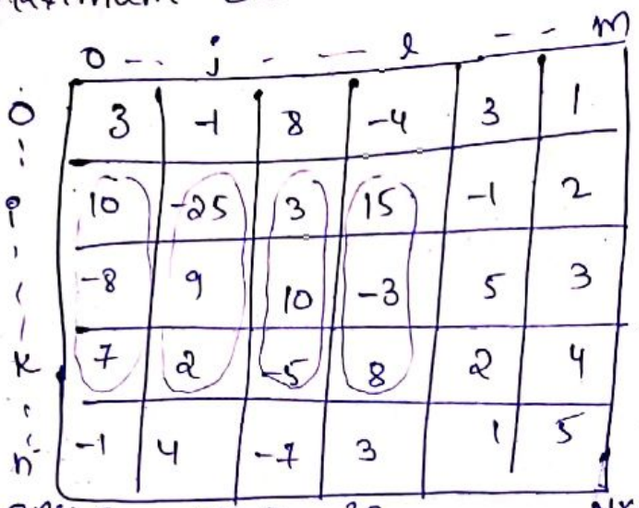
1	3	6	10
6	11	.	.

dp[i][j] = dp[i-1][j] + dp[i][j-1] + mat[i].

ii) dp[i][j] = dp[i-1][j] + dp[i, j-1] - dp[i-1, j-1] + mat[i][j]



Maximum Submatrix Sum.



Solutions: 9 -14 8 20 ..... NxM.

9 -5 8 28 ..... (If sum kadane's Alg)

1. Brute-force

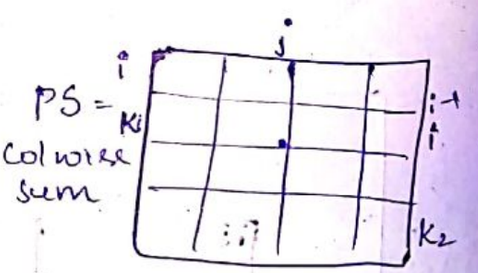
$N \times M \times (\underbrace{N \times M}_{\text{sum}}) = N^2 \times M^2$.

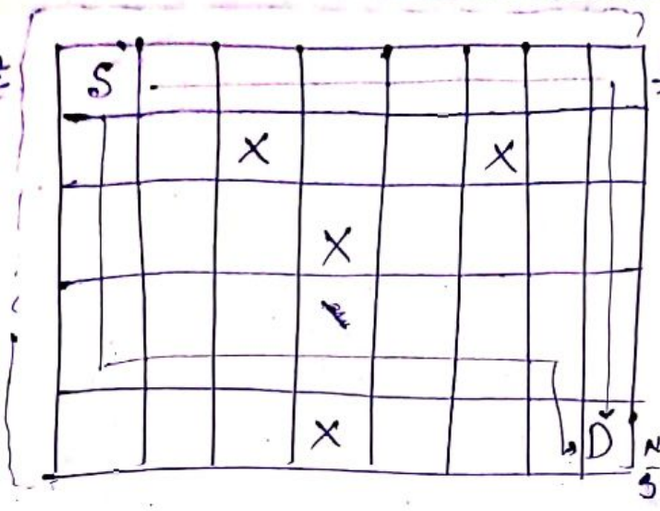
2. Prefix Sum by calculating the column wise sum before

$N \times M + N^2 \times M$.

(Colwise Sum)

```
ans = INT_MIN
for i = 0 to N:
    for k = 0 to N:
        currans = 0
        for j = 0 to m:
            currans += PS[k][j] - i == 0 ? 0 : PS[i-1][j]
            ans = max(ans, curr_ans)
            if currans < 0:
                currans = 0
```





Q:- No. of ways to reach from Source to destination.

DP state:- # ways to reach cell (i,j) from 0,0

DP Exp: DP[i][j] = DP[i-1][j] + DP[i][j-1] → Non block cell.

= 0 → block cell (X)

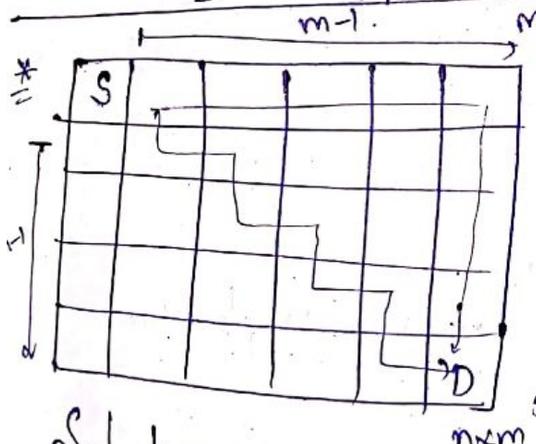
Base cond: To handle BC, add 1 row & 1 col

Table size:- (N+1) x (M+1)

Time Comp: (N+1)(M+1) x O(1) = O(NM)

Answer:- DP[n][m]

Space Opt:- Not possible



→ No blocks

→ No. of ways to reach from Source to Destination

If 5 x 8 then no. of steps required to reach destination = (8+5-2) = 13-2 = 11

(M+N-2)

Solutions:-

1:- Above DP Solution

2:- N+M-2 C_{N-1} = N+M-2 C_{M-2}

$^{5}C_{5} \times {}^{15}C_{10} = {}^{15}C_{5} \times {}^{10}C_{10}$

(1)

15 = 10 + 5

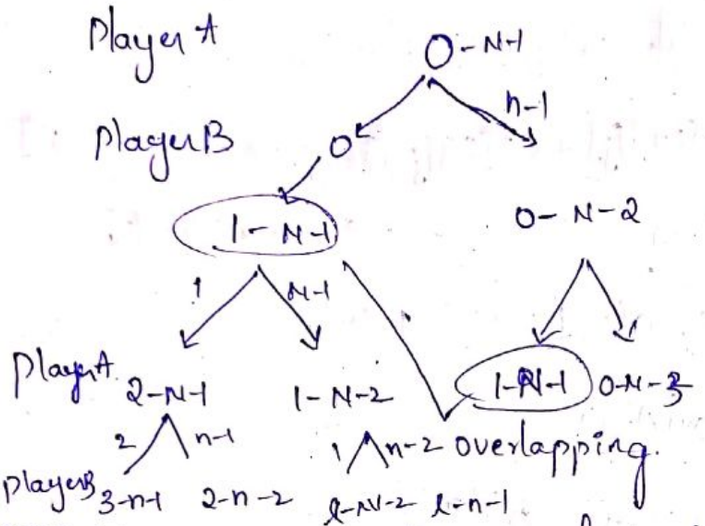
$N_C_R = N_C_{N-R}$

* Ar_N: 15 3 10 18 7 100 25

A: 25

B: 100

Two players



DP[i j] = Max sum for the first player playing on subarray

$$DP[i,j] = \text{Max} \begin{cases} Ar[i] + \min(dp[i+2, j], dp[i+1, j-1]) \\ Ar[j] + \min(dp[i+1, j-1], dp[i, j-2]) \end{cases}$$

---

* $A_{5\times3} \times B_{3\times8} = (A\times B)_{5\times8}$

(matrix Mult)

for each cell in 5×8 we will do multiplications for ③ multiplications.

=> (5×8×3)

i) $(A_{5\times3} \times B_{3\times8}) \times C_{8\times7}$

5×8

$(5\times8\times3) + (5\times7\times8) = 400$

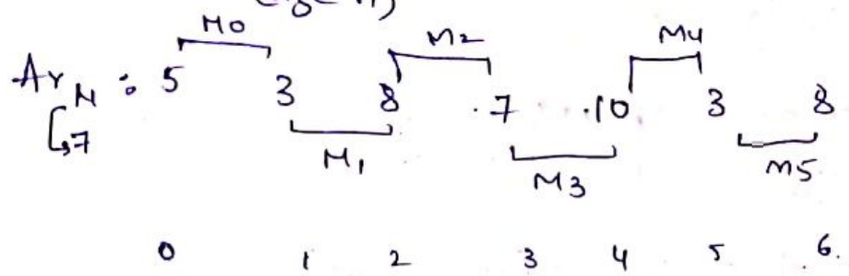
ii) $A_{5\times3} \times (B_{3\times8} \times C_{8\times7})$

3×7

$(3\times7\times8) + (5\times7\times3) = 273.$

* Given an array which has dimensions of (n-1) matrices.

(size n)



$$0-5 = \begin{cases} [0:0] + [1:5] + 5\times 8\times 3 \\ [0:1] + [2:5] + 5\times 8\times 8 \\ [0:2] + [3:5] + 5\times 8\times 7 \\ [0:3] + [4:5] + 5\times 8\times 10 \\ [0:4] + [5:5] + 5\times 8\times 3 \end{cases}$$

(i, k, k+1, j; Ar[i], Ar[j+1], Ar[k+1])

$j \geq i$

DP State:-

DP[i j] = min no. of iterations required to multiply matrices from i to j.

DP Exp.

(n-2 n-2) (Exchange for Bottom up, right to left)

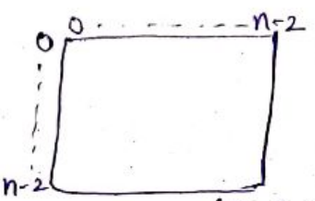
$$\forall_{i=0}^{n-2}\forall_{j=i}^{n-2} dp(i,j) = \min_{K=0}^{j-1} dp[i\ K] + dp[K+1\ j] + (Ar[i] \times Ar[j+1] \times Ar[K+1])$$

Base Cond:-

DP[i j] = 0

Table Size:-

(n-1) x (n-1)

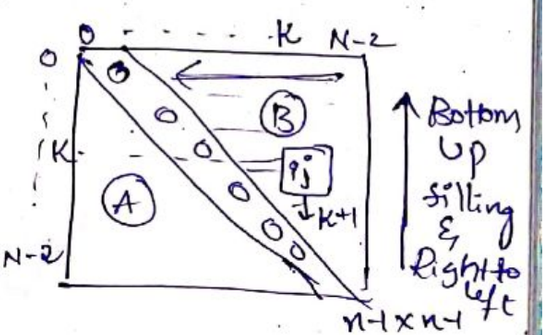


(n-1) x (n-1).

Time Complexity:-

$N^2 \times N = O(N^3)$

(no. of states)



Answer:-

dp[0][n-2]

Standard Dp problems:- (Also refer Geeks for Geeks Dp Section)

1. Building Bridges
2. Stacking Boxes } LIS
3. Bitonic Sequence
4. LCS (Longest Common Subsequence)
5. LCS (Longest Common Substring)
6. Edit Distance
7. Rod Cutting
8. Palindrome partitioning
9. Largest Square matrix with all 1's.
10. Largest Rectangular matrix with all 1's