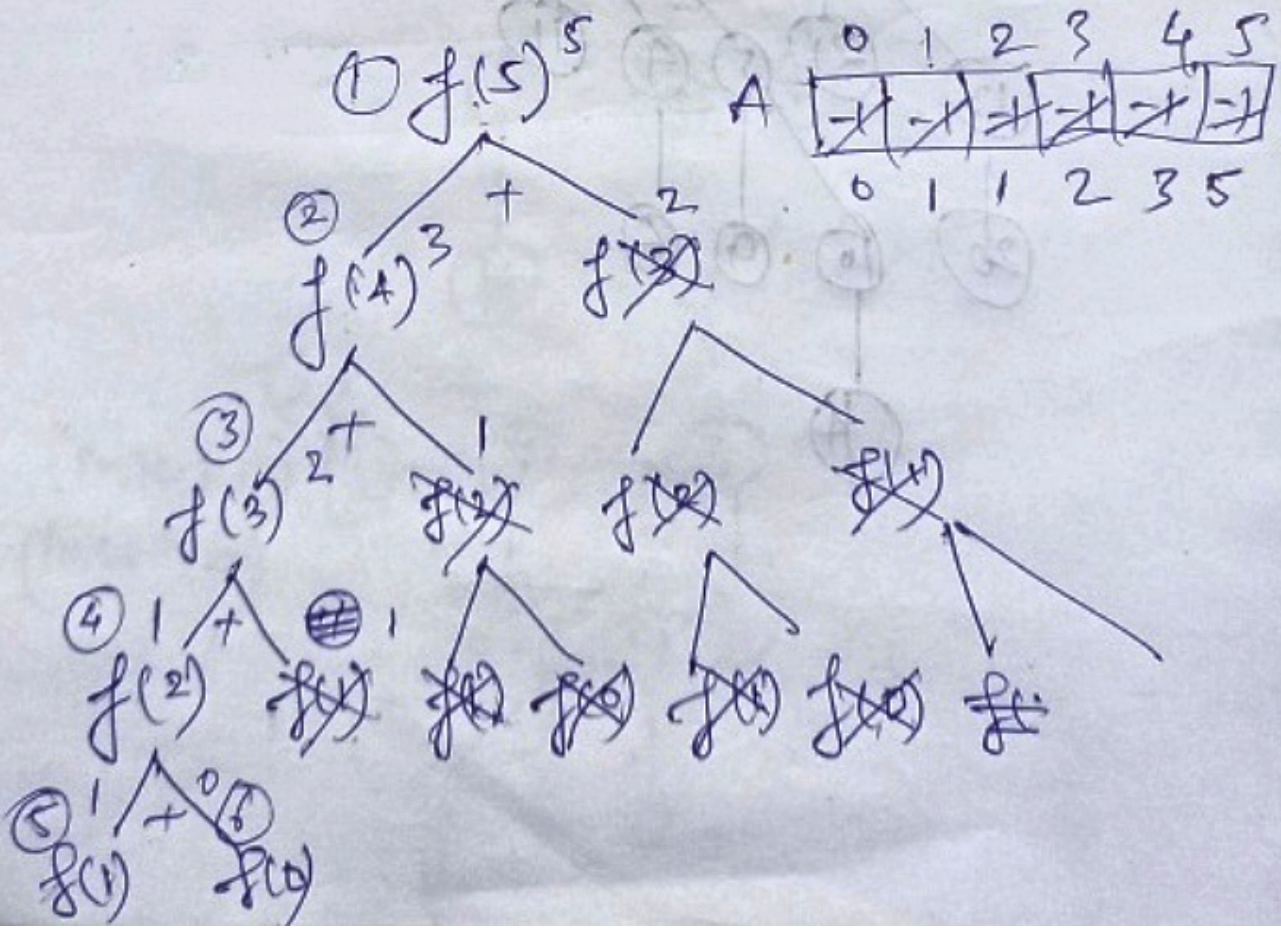
Dynamic programming

1. Optimal substructure
2. Overlapping subproblems.

fibonacci series

f(n)	0	1	1	2	3	5	8	13
n	0	1	2	3	4	5	6	7

$$f(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ f(n-1)+f(n-2) & \text{if } n>1 \end{cases}$$



Brute force method.

Approx $\frac{2^n}{2}$ = $O(2^n)$ time complexity (exponential time)

No of func call for fibonacci of n

f(n) memoization (Top down DP)

⇓ = $O(n)$ polynomial time

n+1

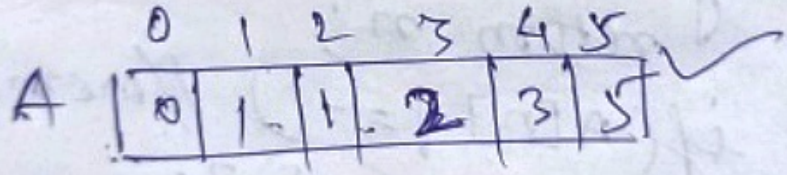
Recursive func of memoization

```
f(n)
{ if (n<=1)
    return n;
  if(A[n] != -1)  //memoization
    return A[n];
  else
    return A[n] = f(n-1) + f(n-2);
}
```

Bottom-up-DP

```
f(n) → int A[n+1]={-1};
{ A[0]=0;
  A[1]=1;
  for(i=2; i<=n; i++)
  {
    A[i]= A[i-1]+ A[i-2];
  }
  return A[n];
}
```

⟹ O(n)



- Tabulation

0/1 Knapsack problem

$KS(n, W) = \begin{cases} 0 & n=0 \text{ or } W=0 \\ KS(n-1, W) & wt(n) > W \\ \max \begin{cases} KS(n-1, W-wt[n]) + P[n] \\ KS(n-1, W) \end{cases} \end{cases}$

(n → no of objects/elements; W → capacity of beg)

Object	1	2	3	4
profit	1	4	5	7
weight	1	3	4	5

W=7

(n+1) (W+1)

	wt	P	W=0	W=1	W=2	W=3	W=4	W=5	W=6	W=7
n=0	0	0	0	0	0	0	0	0	0	0
n=1	1	1	0	1	1	1	1	1	1	1
n=2	3	4	0	1	1	4	5	5	5	5
n=3	4	5	0	1	1	4	5	6	6	9
n=4	5	7	0	1	1	4	5	7	8	9

9 → Tot profit

K(0,0)

KS(1,1) → KS(0,0)+1 ; K(0,1)

$KS(1,2) = KS(0,1) + 1$ (0) $\quad KS(0,2)$ (0)

$KS(2,1) = KS(1,1)$ (1)

$KS(2,2) = KS(1,2)$ (1)

$KS(2,3) = KS(1,0) + 4$ (0) $\quad KS(1,3)$ (1)

$K(2,4) = KS(1,1) + 4$ (1) $\quad KS(1,4)$

$K(3,4) = KS(2,0) + 5$ (0) $\quad KS(2,4)$ (5)

$K(3,5) = KS(2,1) + 5$ (1) $\quad KS(2,5)$ (5)

$KS(3,6) = KS(2,2) + 5$ (1)

$= KS(2,6)$ (5)

$KS(3,7) = KS(2,3) + 5$ (4)

$KS(2,7)$ (5)

$KS(4,5) = KS(3^{0},0) + 7$

$KS(3^{5},5)$

$KS(4,6) = KS(3^{1},1) + 7$

$KS(3^{6},6)$

$KS(4,7) = KS(3,2) + 7$

$KS(3^{9},7)$

~~to~~ get final answer How many objects are included

1	2	3	4
0	1	1	1

$9 - 5 = 4 - 4 = 0$

$4 + 5 = 9$ Ans

$= (n+1)(w+1)$

$= O(nw)$