DYNAMIC PROGRAMMING

DYNAMIC PROGRAMMING

- REMEMBER!!!
- Overlapping Subproblems
  - divide the problem into small problems
  - solve subproblems recursively
  - store solutions to use it again instead of recalculate it
- Optimal Substructure

REMEMBER SOLUTIONS

- Memoization (top down)
- Tabulation (bottom up)

FIBONACCI SERIES

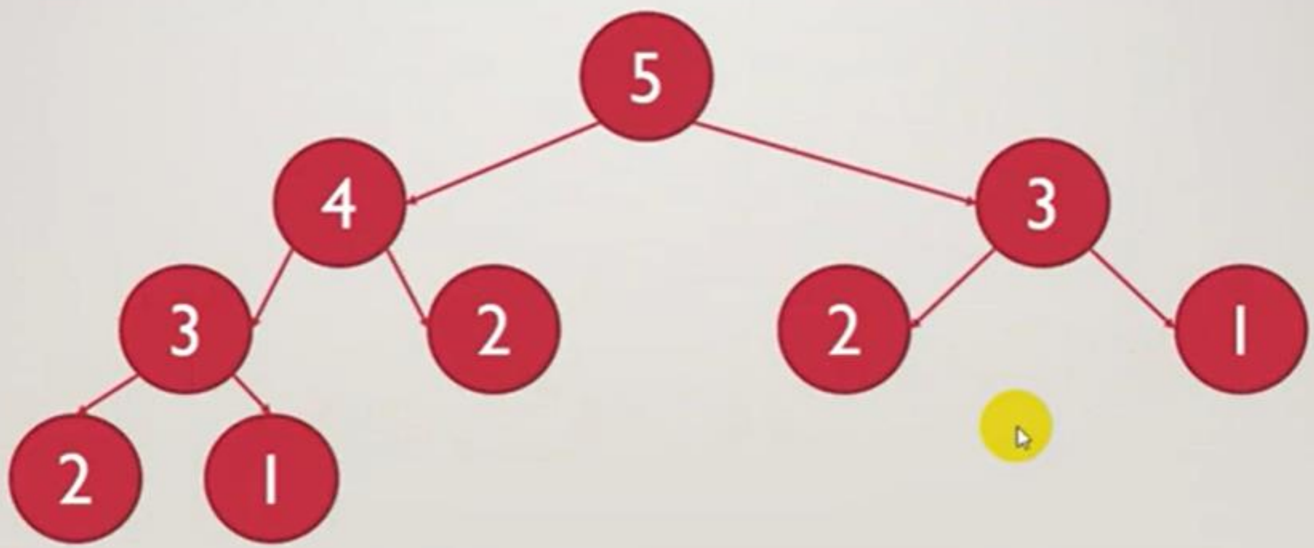
FIBONACCI SERIES - RECURSION

- Pseudo code

```
fib(n)
    if (n ==1) || (n==2) return 1
 return (fib(n-1) + fib(n-2))
```

O(2^n)

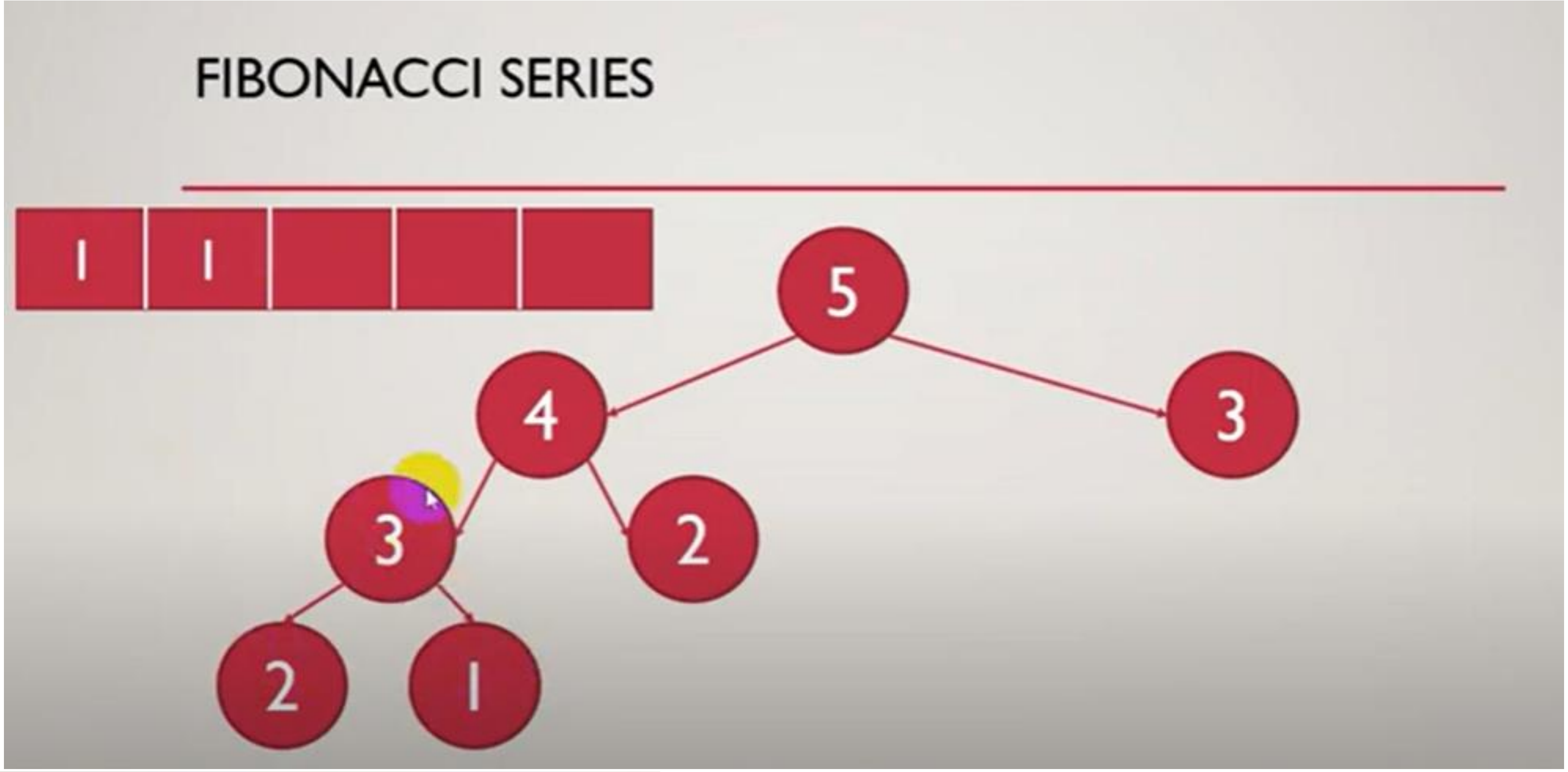
FIBONACCI SERIES



FIBONACCI SERIES - MEMOIZATION

```
arr[max]

fib(n)
    if n == 1 or n == 2
        return 1
    if arr[n] is null
        arr[n] = fib(n-1) + fib(n-2)
    return arr[n]
```

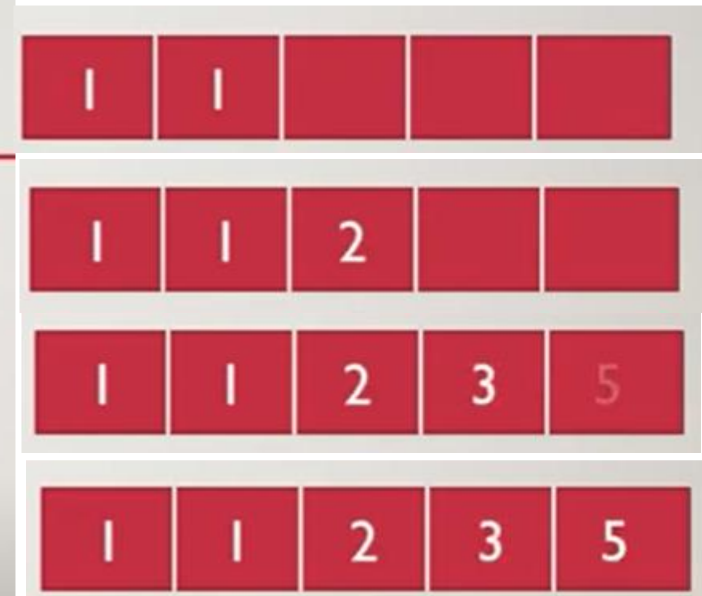

FIBONACCI SERIES
1 1
5
4
3
3
2
2
1


1 1 2
1 1 2 3
1 1 2 3 5

FIBONACCI SERIES - TABULATION

```
arr[max]

fib(n)
    arr[1] = 1
    arr[2] = 1
    for i = 3 to n
        arr[i] = arr[i-1] + arr[i-2]
    return arr[n]
```

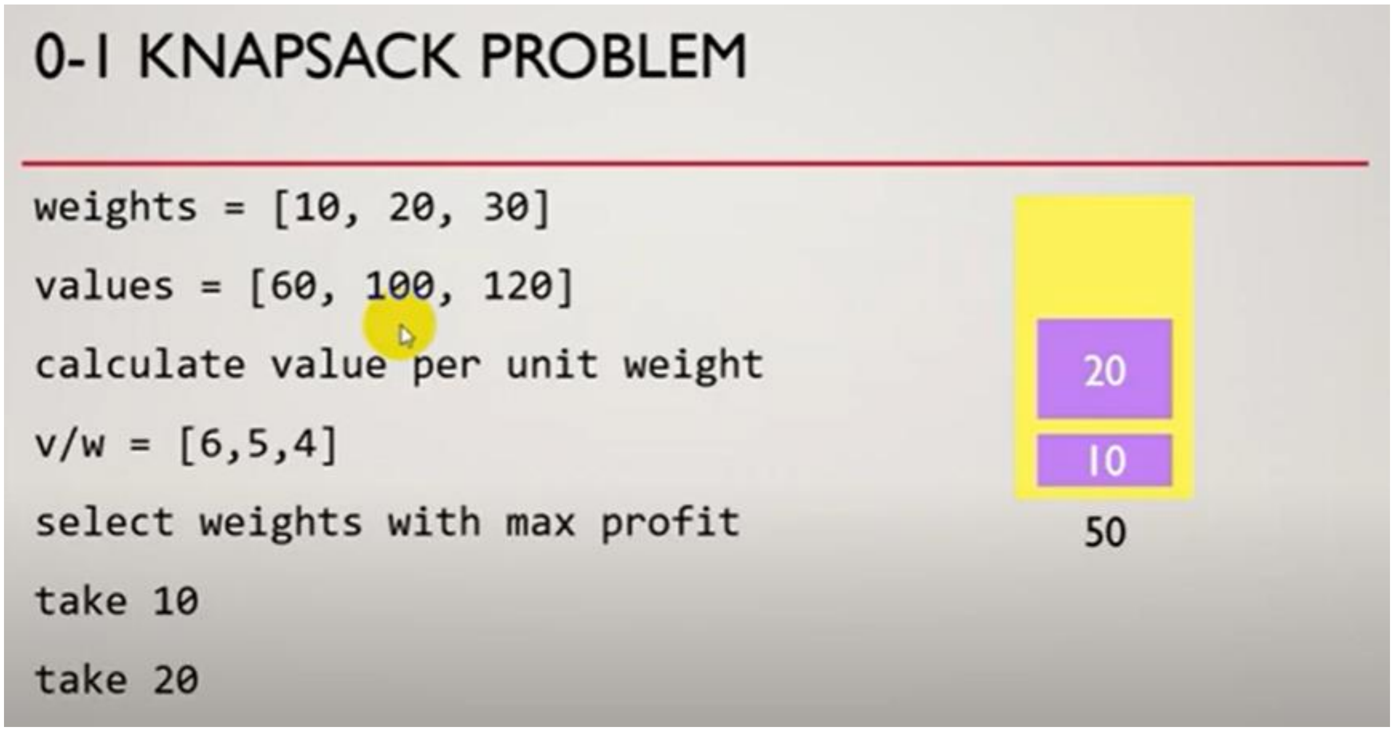


0-1 KNAPSACK PROBLEM

0-1 KNAPSACK PROBLEM

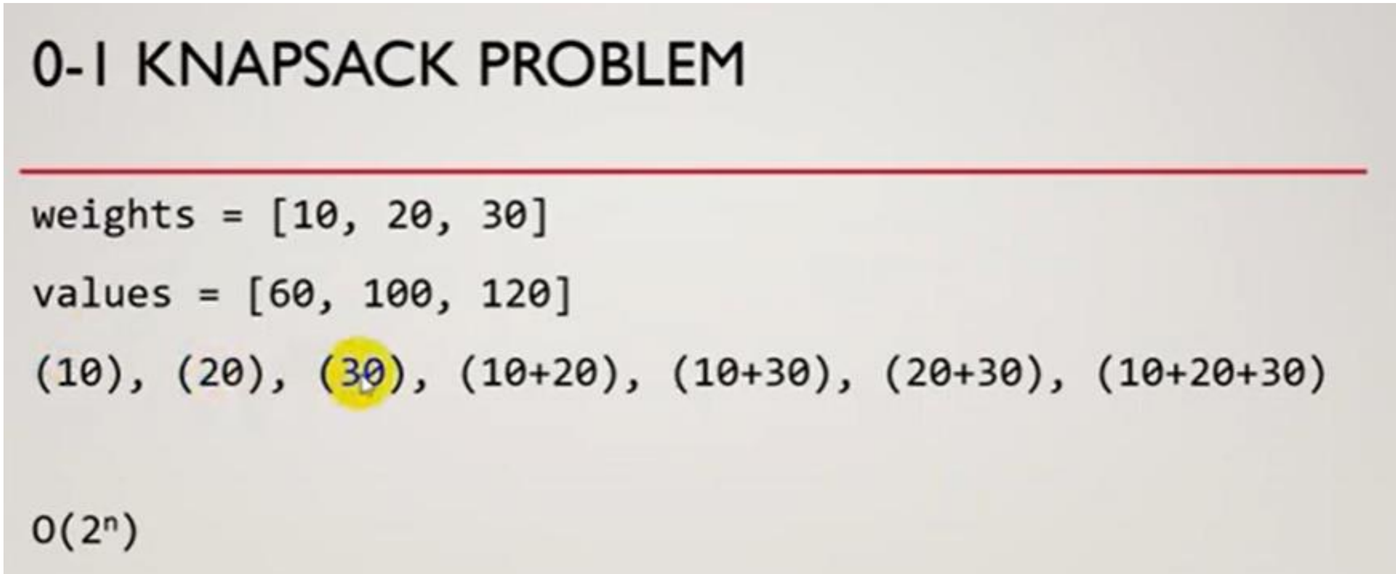
- given “n” items have weights and values
- you want to fill a bag of capacity “c” with items to get maximum value

Using greedy algorithm is not efficient with 0-1 knapsack problem



0-1 KNAPSACK PROBLEM

- consider all possible subsequences
- calculate value for each subsequence
- take maximum value



The number of probability will be very large as the input increase such that it will be 2^n

So this method cannot be used and we will use dynamic programming

When we use dynamic programming in optimization problems like this it is preferable to use tabulation