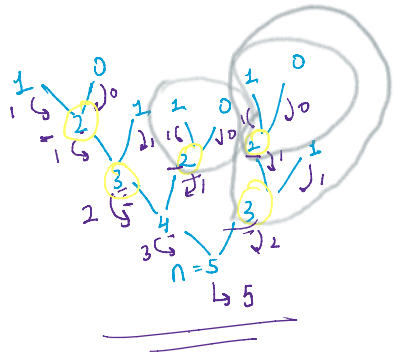
23rd Feb

23 February 2022 19:11

Dynamic Prog.s-

Fibonacci

0, 1, 1, 2, 3, 5, 8, 13, 21, 34 . . . .

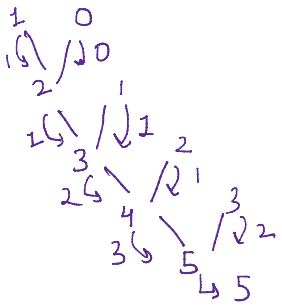


n = 5

```
fib(n) {
if(n==0 || n==1) return n;
return fib(n-1)+fib(n-2);
}
```

↓ T.C => 2^n

```
int fibDP(n, dp[]){ = T.C -> O(n)
                      S.C -> O(n)
if (n==0 || n==1), return n;
if (dp[n] != -1) return dp[n];
return dp[n]= fibdp(n-1, dp) + fibdp(n-2, dp);
}
```



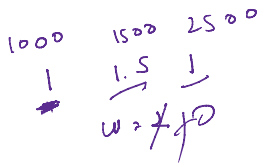
n = 5

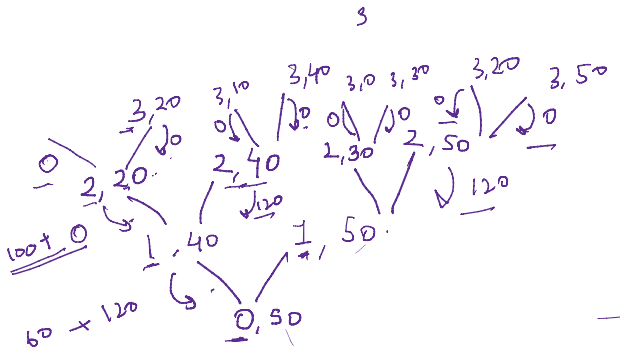
dp[n+1] = [-1, -1, 1, 2, 3, -1]

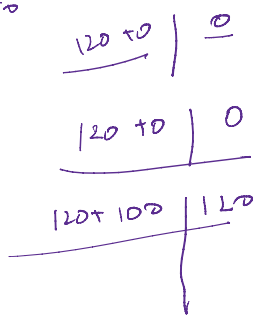
0-1 Knapsack

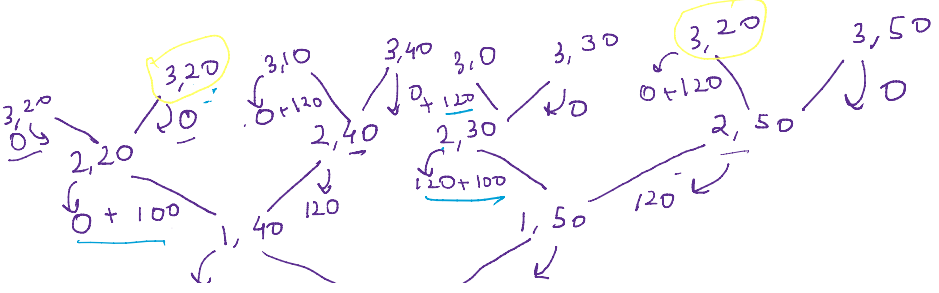
value[] = {60, 100, 120}; W = 50

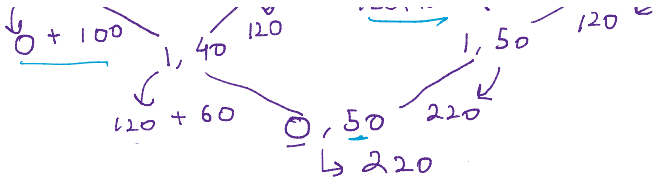
weight[] = {10, 20, 30};












```
value[] = {60, 100, 120};
weight[] = {10, 20, 30};
```

```
int knapsack(wt[], val[], W, i){
    if(i == wt.length) return 0;
    sel = W ≥ wt[i] ? knapsack(wt, val,
                W - wt[i], i+1) + val[i] : 0;
    rej = knapsack(wt, val, W, i+1)
    return max(sel, rej);
}
```

```
n = 3
W += 50
dp[][] = new int[n+1][W+1]
ksDP(wt[], val[], W, i, dp[][]){
if(i == wt.length) return 0;
if(dp[i][W] != 0) return dp[i][W];
sel = W ≥ wt[i] ? kspd(wt, val,
                W - wt[i], i+1, dp) + val[i] : 0;
rej = kspD(wt, val, W, i+1, dp)
return dp[i][W] = max(sel, rej);
}
```

Matrix chain Multiplications:-

arr = [10, 30, 5, 60]

A = 10 × 30

A_{ij} , B_{ij}

A[j] == B[i]

A × B = C[A[i]][B[j]]

$A = 10 \times 30$

$B = 30 \times 5$

$C = 5 \times 60$

$A \times B = C\,[A[i]]\,[B[j]]$

$A \times B = D = 10 \times 5$

$A \times B = C = [10 \times 5 \times 30]$

$C \times D = [10 \times 60 \times 5]$] $(AB)C$

$A(BC)$ 30×60

$BC = D = (30 \times 60 \times 5)$

$A \times D = (10 \times 60 \times 30)$