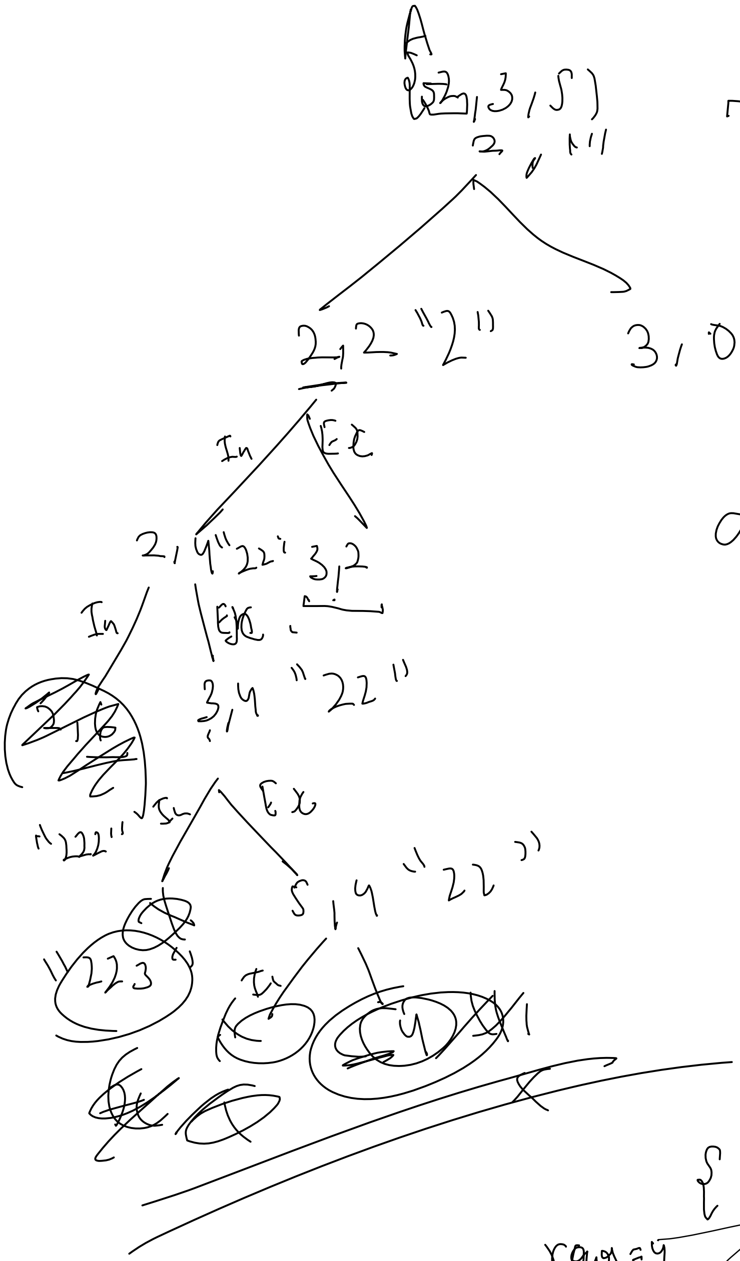
Sunday, August 21, 2022 11:45 AM



0, rows   0 — A

r   C

(idx, A) → ans

Hash < idx , Hash < A, ans >>

dP[idx][A]

{2, 3, 5}, A=5

rows=4 , A=6

Coin	A	0	1	2	3	4	5
2	0	1	0	1	1	1	2
3	1	1	0	0	1	0	1
5	2	1	0	0	0	0	1
	3	0	0	0	0	0	0

0

1 ← id+x

2, -1

```
for (int idx = Coins.length - 1; idx >= 0; idx--)
{
            for (int A = 0; A <= Amount; A++) {
                if(A==0) {
                    dp[idx][A] = 1;
                    continue;
                }
                int sp1 = 0;
                if (A - Coins[idx] >= 0) {
                    sp1 = dp[idx][A - Coins[idx]];
                }
                int sp2 = dp[idx + 1][A];
            dp[idx][A] = sp1 + sp2;
        }
    }
```

D-1 { 10, 20, 5, 15 }

D-2d 20, 40, 10, 50 }

D-3rd 30, 60, 15, 45 }

```
D1 = 2,3 ,5  ,1  ,4          Sell bottle 2
D2 = 4,6 ,10,2  ,8           Sell bottle 6
D3 = 6,9 ,15,3  ,12         Sell bottle 12
D4 = 8,12,20,4 ,16          Sell bottle 4
D4 = 8,15,25,4 ,16
```

s=1, e=4

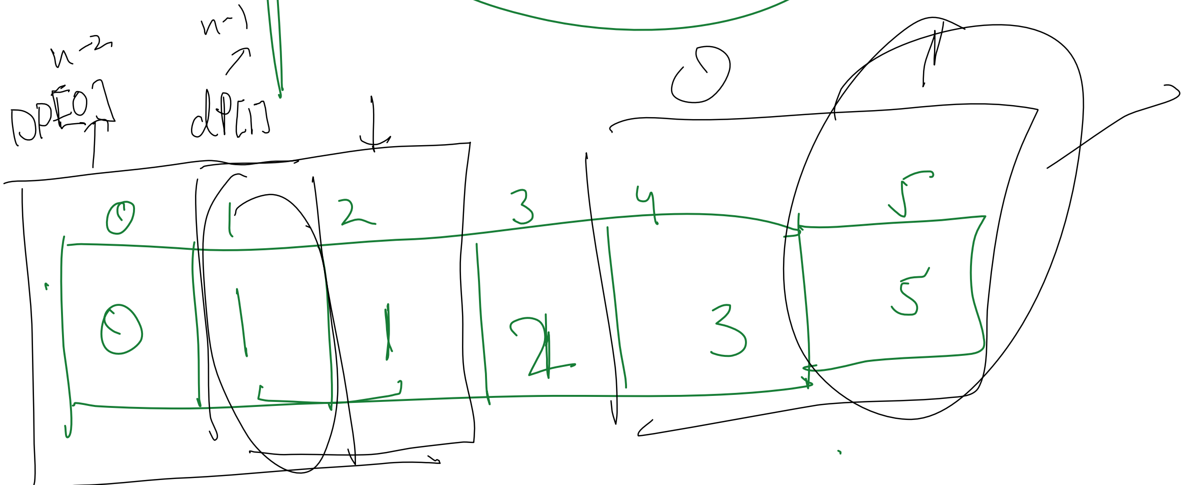
s=0, e=3

Day = sold + 1

s=0, e=4

curr.len - (e-s+1)

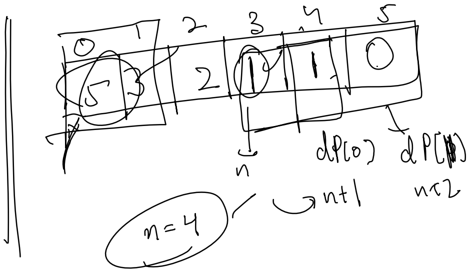
5-(e-s+1)



dP[1] ← curr

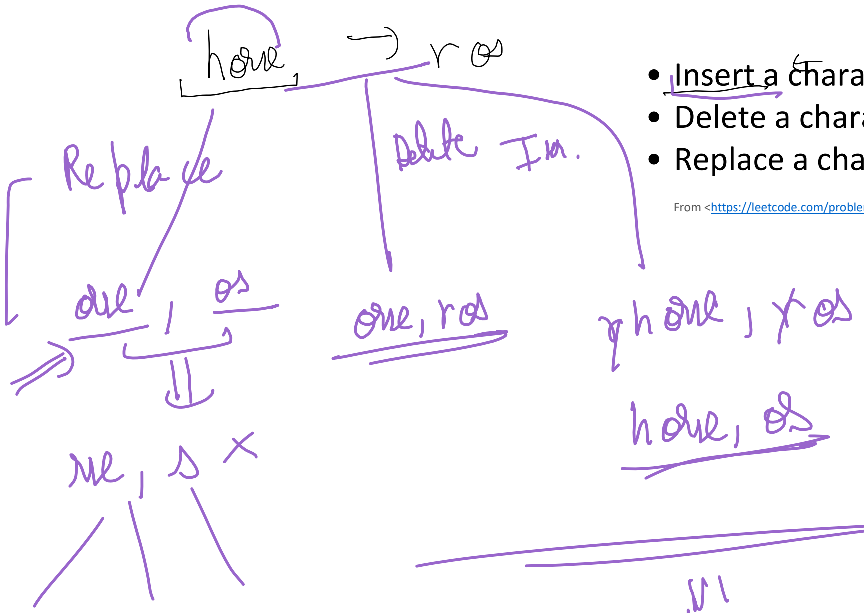
dP[0] ← dP[1]

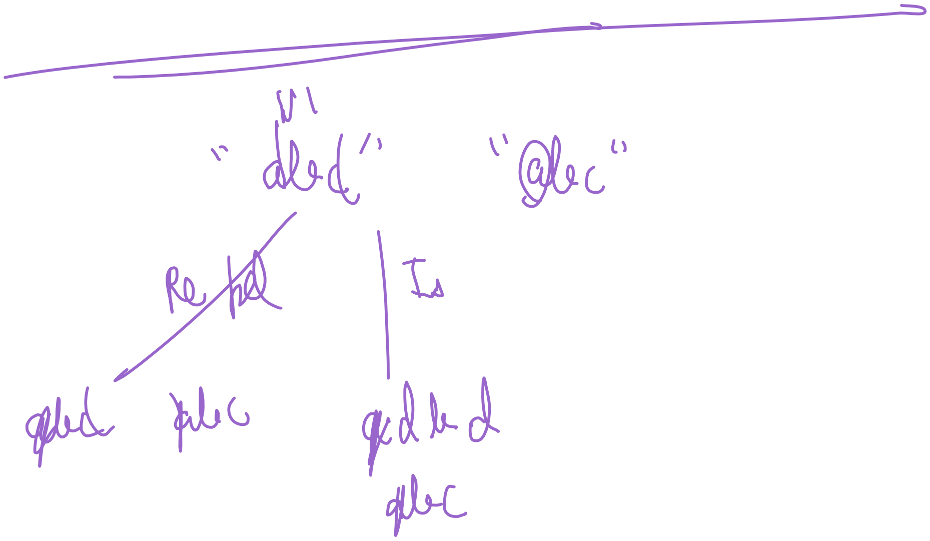
```
public static int climb_BU(int n) {
        int[] dp = new int[n+2];
        dp[n]=1;
        for (int curr = n-1; curr >= 0; curr--) {
//          climb(curr)
                int sp1 = dp[curr + 1];
                int sp2 = dp[curr + 2];
                dp[curr] = sp1 + sp2;
        }
        return dp[0];
    }
```

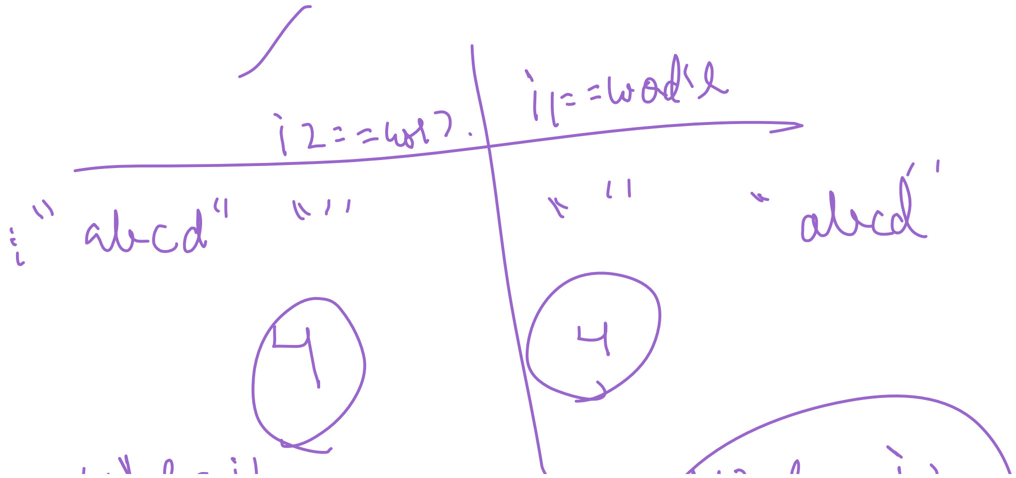


- Insert a character
- Delete a character
- Replace a character

From <https://leetcode.com/problems/edit-distance/>

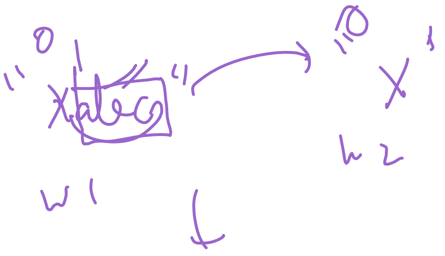






w1.h - i1

w2.len - i2



$(500)^2$

5×10^2

25×10^4

2,50,000

(1,1)

4-1

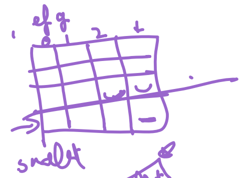
```
for (int i1 = w1.length(); i1 >= 0; i1--) {
                    for (int i2 = w2.length(); i2 >= 0; i2--) {
//                                  solve(i1,i2)
                          if (i1 == w1.length() || i2 == w2.length()) {
                                int ans1 = w1.length() - i1;
                                int ans2 = w2.length() - i2;
                                dp[i1][i2]= Math.max(ans2, ans1);
                                continue;
                          }
                          if (w1.charAt(i1) == w2.charAt(i2)) {
                                int ans = solve(w1, i1 + 1, w2, i2 + 1);
                                dp[i1][i2] = ans;

                          } else {
                                int rep = 1 + dp[i1 + 1][i2 + 1];
                                int del = 1 + dp[i1 + 1][i2];
                                int ins = 1 + dp[i1][i2 + 1];
                                dp[i1][i2] = Math.min(rep, Math.min(del, ins));
                          }
                    }
```

BU SE

SE

HR-1
HR-2

3,3
abc , efg



smallest

for(smallest

TD array

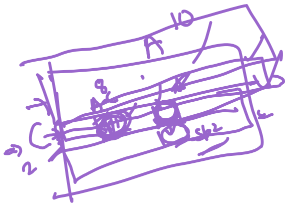
Rec 4-7

Sat 4-7

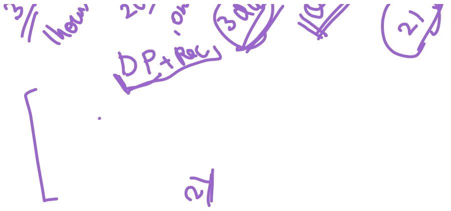
Rec
Coin change

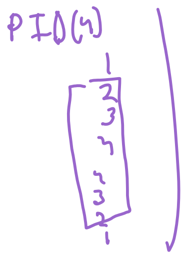
(idx A)

(idx A C



3/ 1hour 20min 10min DP+Rec 3Qs 10Qs 21





BP→ PID(1,4)

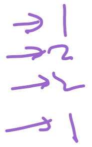
SP→ PID(2,4)

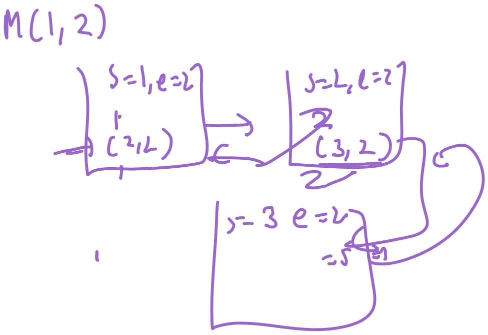
PID ✓

Fib()

a^b

```
public static void PID(int s, int e) {
        if(s>e) {
                return;
        }
        System.out.println(s);
        PID(s+1,e);
        System.out.println(s);
}
```





BP ⇒ (a^b)

sp → a^{b-1} = sp

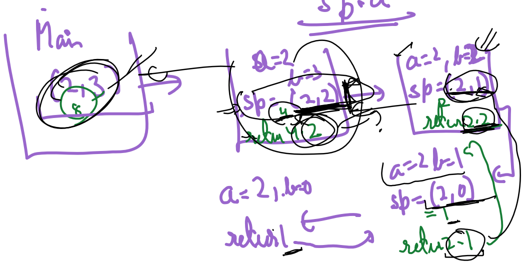
a^{b-1} = sp a^b ?

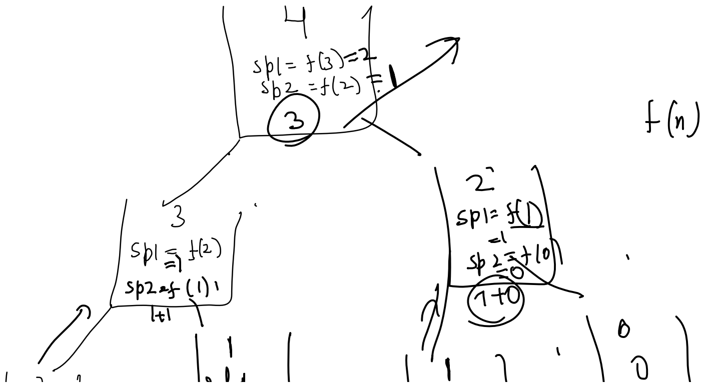
sp·a

return

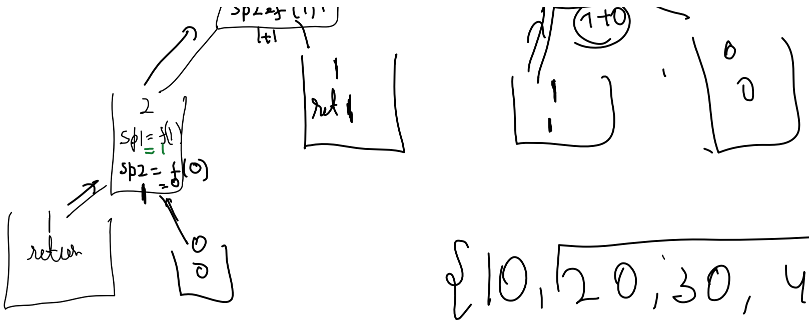
```
public int pow(int a, int b) {
        if(b==0) {
                return 1;
        }
        int sp = pow(a, b- 1);
        return sp*a;
}
```

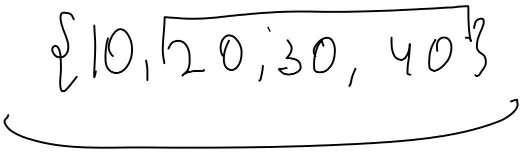
4





$$f(n) = f(n-1) + f(n-2)$$
$$= 2f(n-2) + f(n-3)$$

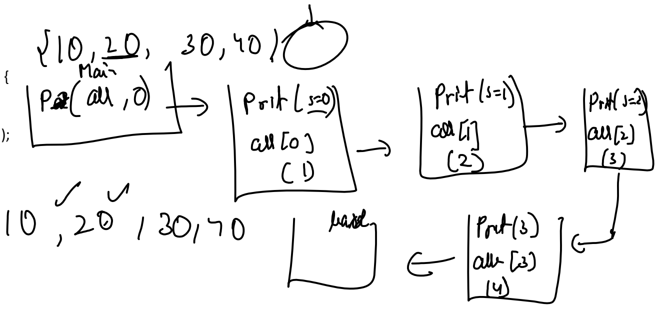




BP ⇒ Prit(all, 0)

SP ⇒ Prit(all, 1)

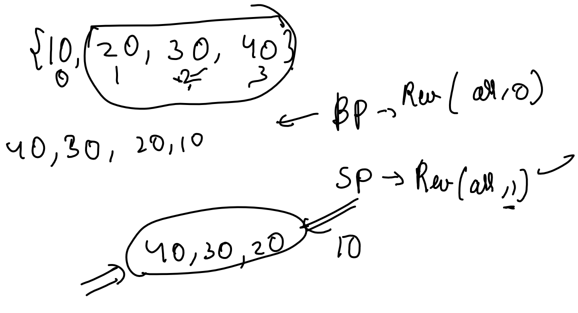
```
public static void Print(int[] arr, int s) {
//                  BP : Print(arr,s)
//                  SP : Print(arr,s+1)
              System.out.println(arr[s]);
              Print(arr, s+1);


       }
```



P (all, 0) {

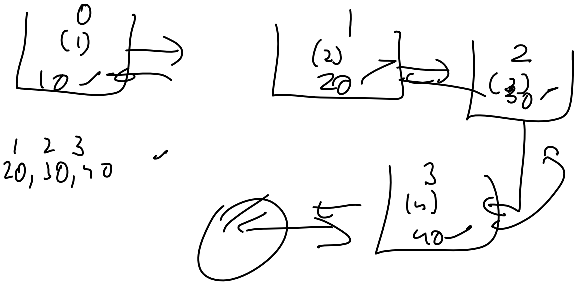
Spo(all[0])

P(all, 1)

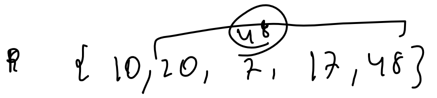


```
public static void Rev(int[] arr, int s) {
              if(s==arr.length) {
                     return ;
              }
//                  BP : Rev(arr,s)
//                  SP : Rev(arr,s+1)
              Rev(arr,s+1);
              System.out.println(arr[s]);
       }
```

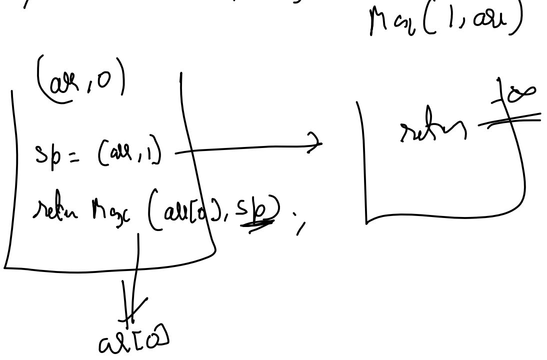
0 1 2 3
10, 20, 30, 40



Max

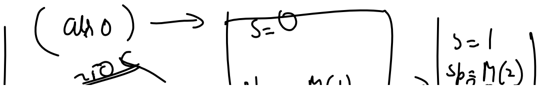


Max(1, arr)

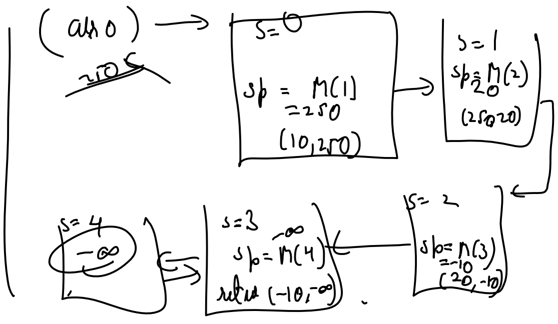


0 1 2 3

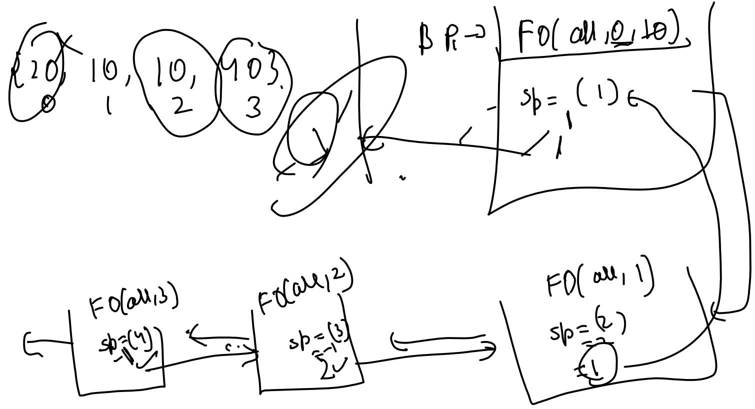
```
int[] arr = { 10, 250, 20, -10 };
public static int Max(int[] arr, int s) {
```



```
              0  1   2   3
int[] arr = { 10, 250, 20, -10 };
public static int Max(int[] arr, int s) {
                if (s == arr.length) {
                        return Integer.MIN_VALUE;
                }
//                      Bp: Max(arr,s)
//                      SP : Max(arr,s+1)
                int sp = Max(arr, s + 1);
                return Math.max(arr[s], sp);
        }
```



Find.



S

{15, 10, 20, 10, 30}
0 1 2 3 4

BP → LO(arr,0,di)

SP → LO(arr,1,di)

3

Q {15, 10, 11, 10, 10, 50}
0 1 2 3 4 5

{1, 3, 4}

int[]