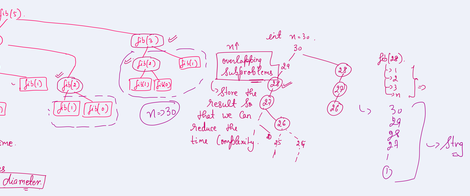
Lecture27

Saturday, 30 May 2020 4:55 PM

Dynamic Programming → Remember the past

```
Q:- fib(n)
{if(int n){
    if (n==0 || n==1){
        return n;
    }
    int fm1 = fib(n-1);
    int fm2 = fib(n-2);
    int fn = fm1 + fm2;
    return fn;
}
```



use extra space to reduce the time.

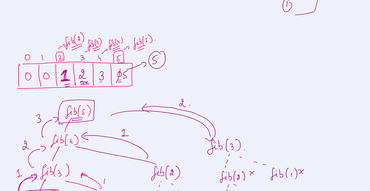
array, hashmap, list, any ds, tree

↳ Trees

↳ D.P → dimension

```
public static int fibRS(int n, int[] Strg){
    if (n==0 || n==1){
        return n;
    }
    if (Strg[n] != 0){
        return Strg[n];
    }
    int fm = fibRS(n-1, Strg);
    int fm2 = fibRS(n-2, Strg);
    int fn = fm1 + fm2;
    Strg[n] = fn;
    return fn;
}
```

Top down approach



↳ $O(n)$

```
public static int fibTS(int n){
    int[] Strg = new int[n+1];
    Strg[0] = 0;
    Strg[1] = 1;
    for (int i=2; i<n; i++){
        Strg[i] = Strg[i-1] + Strg[i-2];
    }
    return Strg[n];
}
```

Seeding

0	1	2	3	4	5
0	1	1	2	3	5

↳ $O(n)$

↳ $O(n)$

↳ time

↳ less space as compared to recursive Strategy.

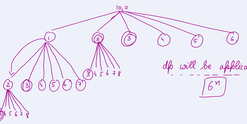
Q:- cfp

via rolling a dice.

How many ways are there?

```
public static int cfp(int end, int curr){
    if (curr > end){
        return 0;
    }
    return 1;
    int count = 0;
    for (int dice = 1, dice <= 6; dice++){
        count = count + cfp(end, curr + dice);
    }
    return count;
}
```

cfp?




```
public static int cfpRS(int end, int curr){
    if (Strg[curr] != 0){
        return Strg[curr];
    }
    for (  ) {
    }
    Strg[curr] = count;
    return count;
}
```

```
public static int cfpTS(int end, int curr){
```

0	1	2	3	4	5	6	7	8	9	10
149	249	125	63	32	16	8	4	2	1	1

```
}
```

Q:- cmp

dp?

(2^{mn})

↳ RS

	0	1	2
0	6	3	1
1	3	2	1
2	1	1	1

↳ $O(mn)$

Q:- Reduce to 1 $\boxed{n}$

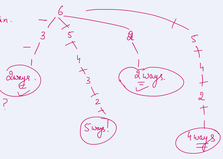
$n \% 2 == n/2 \rightarrow 3$

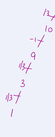
$n \rightarrow n \% 2 \rightarrow n/2 \rightarrow 1$

$n \rightarrow n \% 3 \rightarrow n/3 \rightarrow 1$

$n-1 \rightarrow \rightarrow 1$

min. ways to reduce it to 1?






```
if (ec == cc && ec == cc){
    return 1;
}
```

```
public static int minSteps(int num){
    if (num==1){
        return 0;
    int ans = ∞;
    if (num%3==0){
        int f1 = 1 + minSteps(num/3);
        ans = min(ans, f1);
    }
    if (num%2==0){
        int f2 = 1 + minSteps(num/2);
        ans = min(ans, f2);
    }
    int f3 = 1 + minSteps(num-1);
    ans = min(ans, f3);
    return ans;
```


$6 \% 3 = 1 \quad 5 \% 3 \rightarrow$

$f1 = \infty$

$f2 = \infty$

$f3 = 2 + 1 = 3$

$3 \% 3 = 1$

$f1 = 1$

$f2 = \infty$

$f3 = 2 + 1$

$\% 2 = 1 \rightarrow$

$4 \% 3 \times$

$f2 = 1 + 1/2 \Rightarrow 2$

$f3 = 1$

$2 \% 2 = 1$

∞

$2 \% 2 == 0$

$f2 = 0$

$f3 = 0$

$(\infty, 0, 0)$

$\rightarrow 0 + 1$

0	1	2	3	4	5	6
0	0	1	1	2	3	2

```
    int[] Strg = new int[num+1];
    Strg[1] = 0;
    for (int i=2; i<=num; i++){
        f1 = ∞
        f2 = ∞
        f3 = ∞
        if (i%3==0){
            f1 = Strg[i/3];
        }
        if (i%2==0){
            f2 = Strg[i/2];
        }
        f3 = Strg[i-1];
        Strg[i] = min(f1, f2, f3) + 1;
    }
    return Strg[num];
```

Basic Questions

Advance Questions ⇒ D.P ⇒ Monday → Recursion

↳ Recursive

monday → 3 papers