1) PD ✓
2) PI ✓
3) PDI :

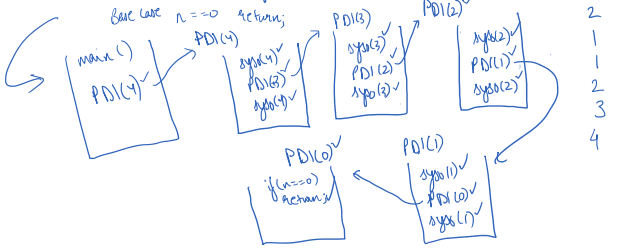
PDI(4): 4 3 2 1 2 3 4 (PDI(3))

PDI(3): 4 3 2 1 1 2 3 4

B.P = PDI(n)

S.P = PDI(n-1)

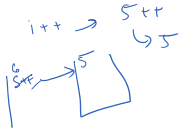
self work: sysо(n) ✓
PDI(n-1) →
sysо(n) ✓



4) PID(S, e)
↳ PID(1,4) :

B.P = PID(s,e)
S.P = PID(s+1,e)

PID(3,4) = PID(3,4)
PID(4,4) = 4
s==e
sysо(s)
return s

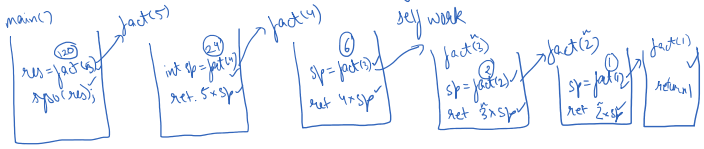


---

1) factorial ⇒ n*(n-1)*(n-2)*(n-3)*.... 3.2.1.
fact(n)

B.P = fact(n)
S.P = fact(n-1) * n

Base case
↳(n==1){
return 1
}



---

2) pow : $a^n \Rightarrow a \cdot a^{n-1}$
↳ (a,n)
2^{10}

B.P = a^n ⇒ pow(a,n)
S.P = pow(a, n-1) * a;

Base case ?? $a^0 = 1$

if(n==0){
return 1;
}

if(n==1){
return a;

if(n==2)
return a*a;
else if(n==1){
return a;
}

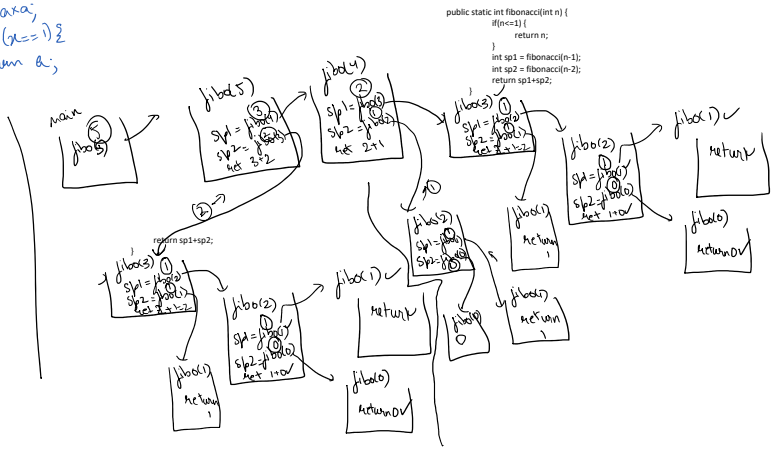
3) fibonacci (n):

res →	0	1	1	2	3	5	8	13	21	34	55
n →	0	1	2	3	4	5	6	7	8	9	10

B.P = fibonacci(n) = SP1 + SP2
S.P1 = fibonacci(n-1) ✓
S.P2 = fibonacci(n-2) ✓

Base case??
if(n<=1){
return n
}

```
public static int fibonacci(int n) {
        if(n<=1) {
            return n;
        }
        int sp1 = fibonacci(n-1);
        int sp2 = fibonacci(n-2);
        return sp1+sp2;
```



Recursion on Arrays

1) Print all elements

↳ print (arr int[], idx);

{10, 20, 30, 40, 50}

syso(arr[idx]);

print(arr, idx+1);

10 ← 0
20
30
40
50

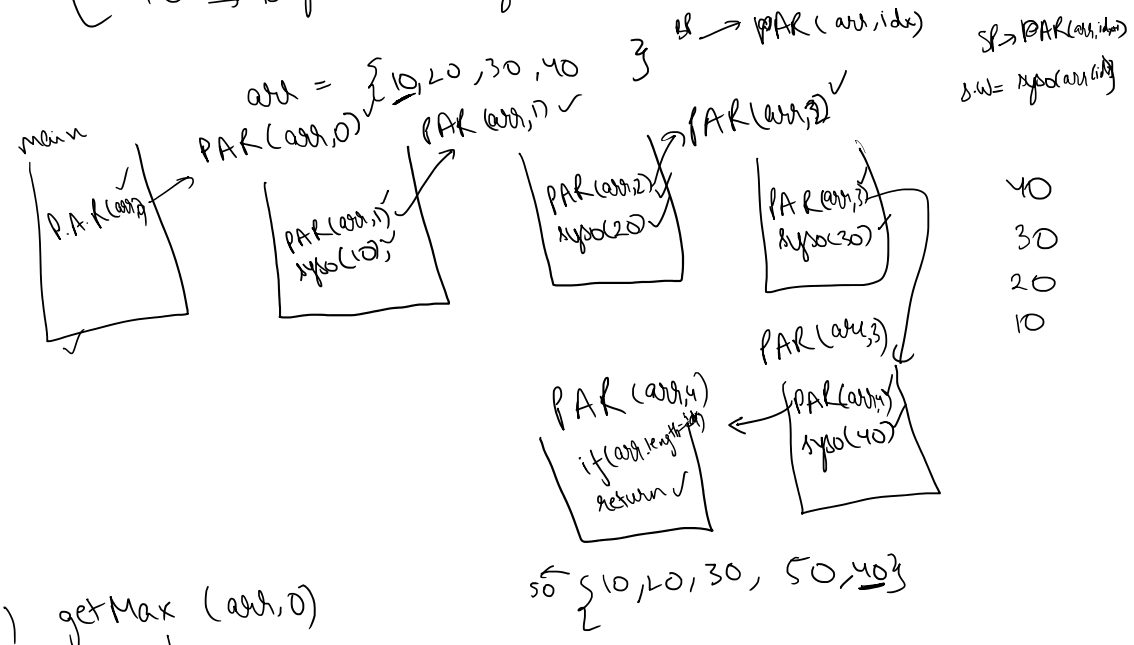
B.P ⇒

Base case : (idx == arr.length) return;

2) Print Reverse (arr, idx) {10, 20, 30, 40, 50}

B.P { 50, 40, 30, 20 } SP → P.Rev(arr, idx+1);

10 → self work syso(arr[idx])



3) getMax (arr, 0)

↳ getMax (arr, idx)

B.P = getMax(arr, idx);

SP = getMax(arr, idx+1)

S.W = return Max (arr[idx], SP)

4) first occurrence

↳ (arr, val, 0)

first_idx / -1

{10, 50, 20, 10, 40, 80, 10, 20} val = 20

0 1 2 3 4 5 6 7

B.P (arr, val, idx)

S.P (arr, val, idx+1);

S.W

↳ check if(arr[idx]==val){ return idx } (arr, val, idx+1);

} else { int sp = firstOcc(arr, val, idx+1);
return sp;

Base case = ? ⇒ -1

5) last occurrence (arr, val, 0)

{10, 50, 20, 10, 40, 80, 10, 70}; val = 10

B.P = LO(arr, val, idx)

↳ S.P = LO(arr, val, idx+1)

sp = LO(arr, val, idx+1)

if(arr[id] == val) {
if(sp == -1) return idx
else sp
} else {
return sp
}

Base case ⇒ ??

X

6) return All Occurrences in Array

int[]

(arr, val, 0, 0)

idx, count

{10, 50, 20, 10, 40, 80, 10, 70}; {0 3 6}

{3,6}

3 extra array

0 to idx

Kitni occurrences

0, {3,6}

B.P

↳ (arr, val, idx, c)

↳ arr[idx] == val → sp = (arr, val, idx+1, c+1)

else

↳ sp = (arr, val, idx+1, c)

Base Case

↳ (idx == arr.length){

// count size ki array return krna

}