


Recursion ↳ when a function calls itself

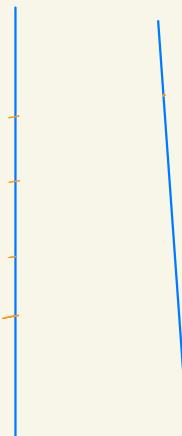
- 1) steps
- ⇒ solve it for the smallest problem.
 - ⇒ Assume that it will work properly for smaller problem (faith)
 - ⇒ Solve it for a smaller solution (your work)

Ques Take an input x , print numbers from x to 1.

⇒ public static void fun(int x) {
 if ($x == 1$) {
 base case
 System.out.println(x);
 return;
 }

step ① ↓
 {
 if ($x == 1$) {
 base case
 System.out.println(x);
 return;
 }
 }
 ↓
 Sys.out(x);

step ② ↓
 fun(x-1);
 ↓
 {



$x-1$

Expectation

it will print numbers from x to 1
properly

work

✓ Faith

it will print numbers from $(x-1)$ to 1

$x=5$

$\frac{5}{4}$
3
2
1

faith $(x-1)$

{ 4
3
2
1

Ques Print numbers from 1 to x

work

Expectation (x) $x=5$ fact

It will print numbers from 1 to x

property

I to well print numbers from 1 to $x-1$ property.

{
1
2
3
4
5

psv print (int x) {

- 1) print ($x-1$);
- 2) System.out.println (x);

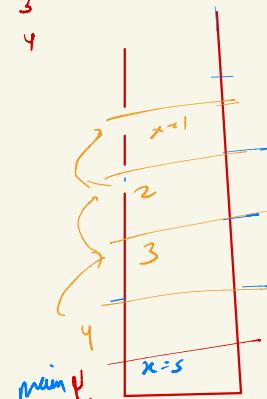
if ($x == 1$) {

System.out.println (x);

return;

}

{
1
2
3
4



1
2
3
4
5

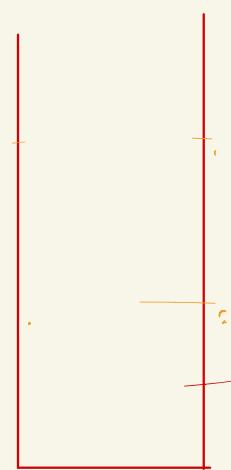
psv print (int x) {

1) if ($x == 1$) {
System.out.println (x);
return;

}

2) print ($x-1$);
3) System.out.println (x);

}



1
2
3
4

Ques Print numbers from n to 1 using recursion.

print decreasing

Expectation

It will print numbers from n to 1

5
4
3
2
1

$psv \quad pd(\text{int } n)$ of

1) Sprint (n)
2) pd(n-1)

3)
 $y(n=1)$ of
Sprint (n)
return;

Does $n=5$
first print
decreasing them
Inversely

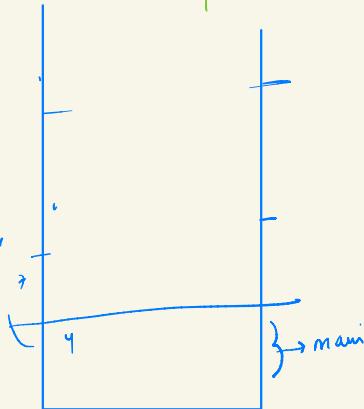
5
4
3
2
1
2
3
4
5

$n=4$)

Fact

It will print numbers from $(n-1)$ to 1

4
3
2
1



4
3
2
1

4
3
2
1
2
3
4

expectation

(n=5)

It will print decreasing
from $n \rightarrow 1$, then $1 \rightarrow n$

faith

(n=4)

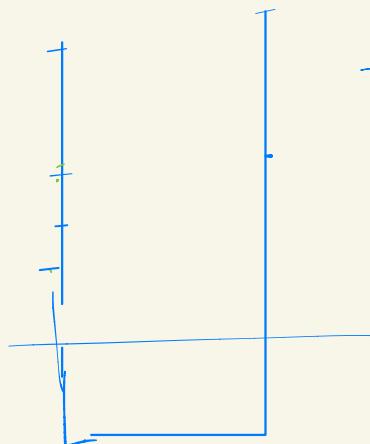
It will print $(n-1)$ to 1
and then 1 to $(n-1)$

5
4
3
2
1
2
3
4
5

```
public static void printDecreasingIncreasing(int x){
    if(x==1){
        System.out.println(x);
        return;
    }
    System.out.println(x);
    printDecreasingIncreasing(x-1);
    System.out.println(x);
```

4
3
2
1
2
3

$\{$
 $\text{if } (x==1)$
 $\text{System.out.println(x)}$
 return



3
2
1
2
3

Ques Write a function to find factorial $n!$

expectation

(n=5)

It will return 120
 $n!$

$5 \times 4 \times 3 \times 2 \times 1$

public static int fac (int n) {
 if ($n == 0$) return 1;

1) int f = fac (n-1);
2) int ans = n * f;
return ans;

faith

It will return
 $(n-1)!$

4
 \downarrow
 $4 \times 3 \times 2 \times 1$
 \downarrow
24

Ques Write function to calculate a raise to power b

Expectation $a=2$
 $b=4$

It will calculate and
return a^b

$$\begin{matrix} 2^4 \\ \downarrow \\ 2 \times 2 \times 2 \times 2 \end{matrix}$$

ps int power(int a, int b){
1) if ($b==0$) return 1;
2) int f = power(a, b-1);
3) int ans = a * f
return ans;

Fault

It will calculate
 a^{b-1}

$O(b)$

}

Expectation $a=2, b=4$

It will calculate a^b .

$\overbrace{2 \times 2 \times 2 \times 2}$

$$ans = f \times f$$

ps int powerWith (a, b){
1) if ($b==0$) return 1;
2) int f = pb(a, b/2);
3) int ans = f * f;
if ($b \cdot 1.2 == 1$)
ans = a * ans;
return ans

Fault

It will calculate $a^{\frac{b}{2}}$

$\overbrace{2 \times 2}$

$O(\log b)$

}

$$f=9$$

$$ans = 9 \times 9 \times 81$$

$$ans = 3 \times 81 \rightarrow 243$$

main {

$a=2, b=7$

$\overbrace{1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7}$

$$b \rightarrow \frac{b}{2} \rightarrow \frac{b}{4} \rightarrow \frac{b}{8} \dots \dots \dots \textcircled{1}$$

$$\frac{b}{2^1} \rightarrow \frac{b}{2^2} \rightarrow \frac{b}{2^x} \dots \dots \dots \left(\frac{b}{2^x} \right)$$

$$\frac{b}{2^x} = 1 \Rightarrow b = 2^x$$

$$\Rightarrow \frac{\log b}{\log 2} = x \log 2$$

$$\boxed{x = \frac{\log b}{\log 2}}$$

Ques Print array using recursion

$$\text{arr} = [10, 29, 11, 5, 43]$$

Expectation
Print elements from index 0

to $(n-1)$:

10

29

11

5

43

Faith

Print elements from index
1 to $n-1$

\downarrow ($\text{idn} = \text{arr.length})$ return

psv print_array (int arr, int idn) {
 1) System.out.println (arr[idn]);
 2) print_array (arr, idn+1);

\downarrow Recursion on the way up

10

29

11

5

43

main {
 arr, 0 } \downarrow

$0 \rightarrow n-1$ reverse

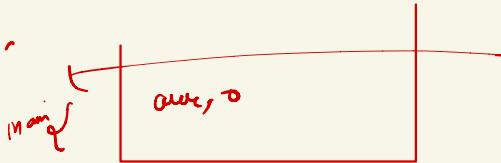
93
5
11
29
10

10, 29, 11, 5, 4? for (arr, idn) {
 for (arr, idn+1);
 System.out.println(idn);
}

$1 \rightarrow n-1$ reverse

93
5
11
- 29

93
5
11
29
10

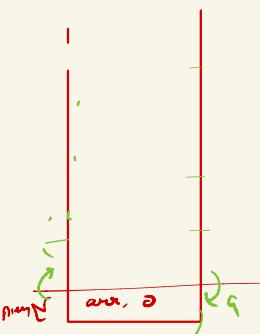


Ques Find max of an array using recursion.

arr = { 0 1 2 3 4
 10, 9, 4, 7, 6 } → [Integer MIN Value]

Intuition

It will find the maximum element b/w (0 - 4) index.



Faith

It will find the maximum element b/w (1-4) index.

9

for max (int[] arr, int `idx`){

- 1) int max-age = max (arr, idx+1)
- 2) int ans = Math.max (arr[idx], max-age);
return ans

Ques Find at which index, tar exists -
 arr = {1, 2, 11, 9, 45}
 0, 2, 3, 4
 tar = 9, ans = 3
 tar = 12, ans = -1

Expectation

It will find idx of tar b/w
 0 and 4

```
// tar=7 0 1 2 3 4 5 6
// arr={2,3,5,7,5,7,24};
public static int find_rec(int[] arr, int idx, int tar){
    1) if(idx==arr.length){
        return -1;
    }

    2) int f=find_rec(arr, idx+1,tar);

    3) if(f==-1){
        if(arr[idx]==tar){
            return idx;
        }
    }

    4) return f;
}
```

Fault

It will find idx of tar
 b/w 1 and 4

⇒ last occurrence

⇒ first occurrence

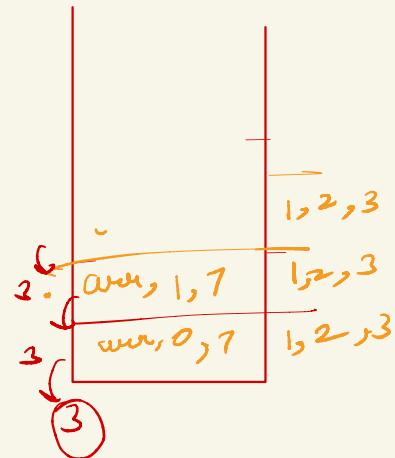
↖ ↘
 Recursion
 on the
 way up recursion
 on the
 way down

0 1 2 3 4 5
2, 3, 5, 7, 5, 7, 24

```
public static int find_fo(int[] arr, int idx, int tar){  
    1) if(idx==arr.length){  
        return -1;  
    }  
    2) int f=find_fo(arr, idx+1,tar);      f= 3  
    3) if(arr[idx]==tar){  
        return idx;  
    }  
    4) return f;  
}
```

⇒ first occurrence
using recursion on
the way down

```
// first occurrence using recursion on the way up  
public static int find_fo_better(int[] arr, int idx, int tar){  
    1) if(idx==arr.length){  
        return -1;  
    }  
    2) if(arr[idx]==tar){  
        return idx;          ans=3  
    }  
    3) int ans=find_fo_better(arr, idx+1, tar);  
        return ans;  
}
```



Ques Find sum of array using recursion

Intuition

Find sum of elements from
0-n

```
public static int sum(int[] arr, int idx){  
    if(idx==arr.length){  
        return 0;  
    }  
  
    int sum_aage=sum(arr, idx+1);  
  
    int ans=sum_aage+arr[idx];  
  
    return ans;  
}
```

Recursion
find sum of elements
from 1-n