Lecture 16

Thursday Nov. 2

$$P(n) = \frac{(1+n)n}{2}$$

$P(n)$ = sum of the first n integers from 1

Base Cases

$P(1)$ $P(2)$

Recursive / Inductive Cases

Assume: $P(n-1) = \frac{(1+(n-1))(n-1)}{2}$

Prove: $P(n)$ (derive)

Problem : $n!$

$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$

$4!$

$7! = 7 * 6!$

size of problem

$$n! = \begin{cases} 1 & n = 0 \\ n \cdot [(n-1) \cdot (n-2) \cdots 1] & n > 0 \end{cases}$$

$(n-1)!$

$n! = n * (n-1)!$

① Base Cases

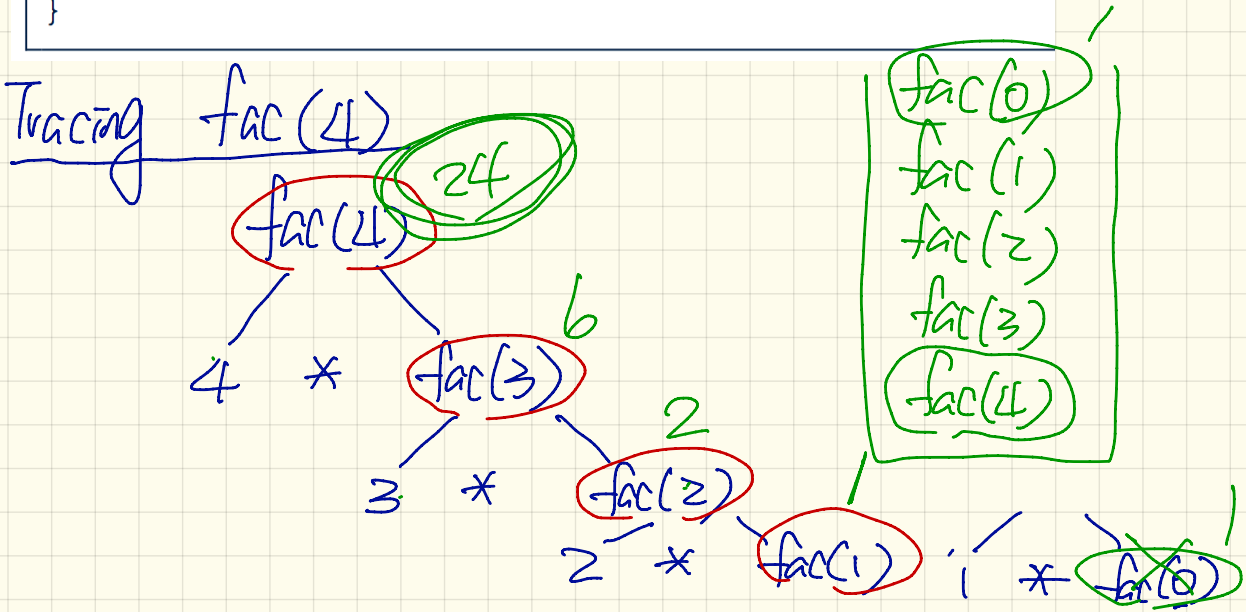
$0! = 1$ $\quad$ $1! = 1$

② Recursive Cases

Assume we have $(n-1)!$ — Solution to a strictly smaller problem.

Derive: $n! = (n-1)! * n$ — $(n-1)!$: sub-problem; $n!$: original problem

```
int factorial (int n) {
  int result;
  if(n == 0) { /* base case */ result = 1; }
  else { /* recursive case */
    result = n * factorial (n - 1);
  }
  return result;
}
```



```
int factorial (int n) {
  int result;
  if(n == 0) { /* base case */ result = 1; }
  else { /* recursive case */
    result = n * factorial (n - 1);
  }
  return result;
}
```

Tracing fac(3)

1. Activate fac(3), push (fac(3))
2. Execute fac(3)
   ↳ 3 * fac(2)
3. Activate fac(2), push (fac(2))
4. Execute fac(2)
   ↳ 2 * fac(1)
5. Activate fac(1)
   ↳ push (fac(1))
6. Execute fac(1)
   ↳ 1 * fac(0)
7. Activate fac(0)
   push (fac(0))
8. fac(0) returns 1
   pop()
   → fac(1) = 1 * 1
   = 1
9. fac(1) returns 1
   pop()
10. fac(2) returns 2

11. fac(3) returns 6
    pop()

Fib. Seq.

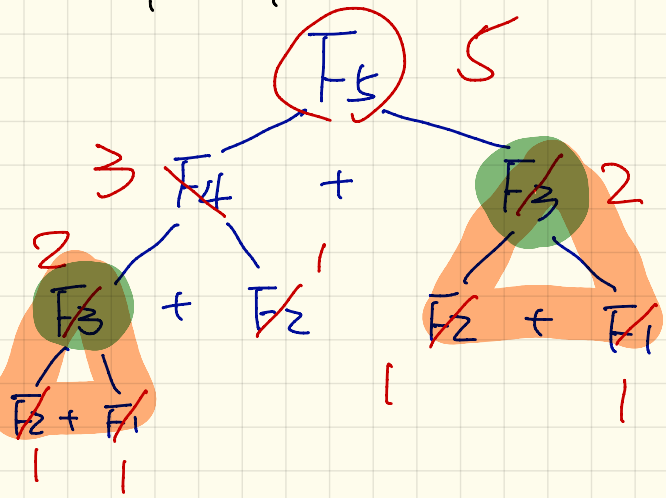
1 1 2 3 5 . . .

F_1 F_2 $F_1 + F_2$

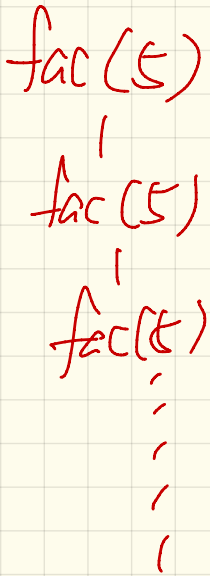
$$F_n = \begin{cases} 1 & n = 1 \\ 1 & n = 2 \\ F_{n-1} + F_{n-2} & n > 2 \end{cases}$$

F_{n-1}: solution to a strictly smaller problem

F_{n-2}: solution to smaller problem



```
int factorial(int n) {
  int result;
  if(n == 0) { /* base case */ result = 1; }
  else { /* recursive case */
    result = n * factorial(n-1);
  }
  return result;
}
```



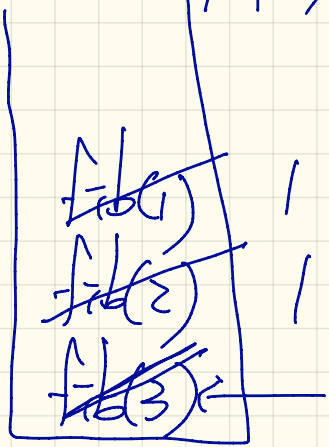
```
int fib(int n) {
  int result;
  if (n == 1) { /* base case */ result = 1; }
  else if (n == 2) { /* base case */ result = 1; }
  else { /* recursive case */
    result = fib(n - 1) + fib(n - 2);
  }
  return result;
}
```

3 2 1

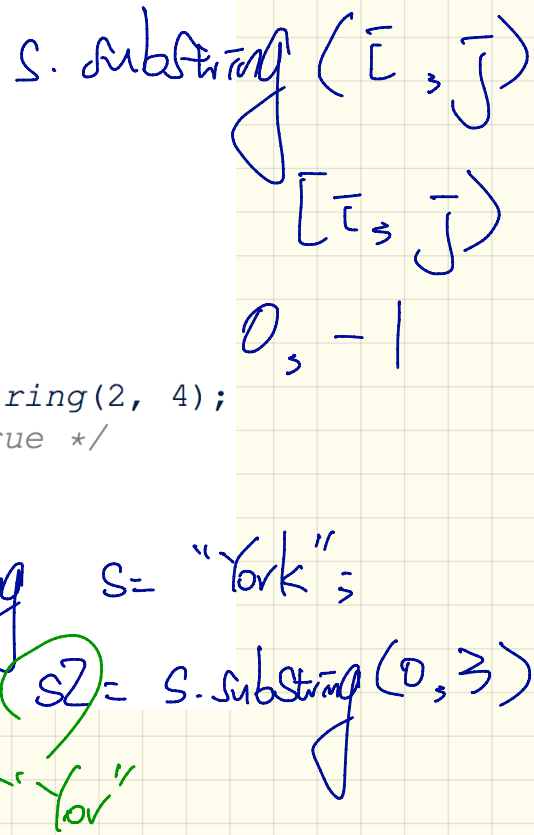
2 1

fib(3)

1. Activate fib(3)
   ↳ push(fib(3))
2. Execute top of stack fib(3)
   ↳ fib(2) + fib(1)
3. Activate fib(2)
   push
4. Execute fib(2)
   ↳ return 1
   ↳ pop
5. Activate fib(1)
   ↳ push
6. Execute fib(1)
   return 1
   pop
7. Execute fib(3)
   return
   1+1 = 2
   pop



```
public class StringTester {
  public static void main(String[] args) {
    String s = "abcd";
    System.out.println(s.isEmpty()); /* false */
    /* Characters in index range [0, 0) */
    String t0 = s.substring(0, 0);
    System.out.println(t0); /* "" */
    /* Characters in index range [0, 4) */
    String t1 = s.substring(0, 4);
    System.out.println(t1); /* "abcd" */
    /* Characters in index range [1, 3) */
    String t2 = s.substring(1, 3);
    System.out.println(t2); /* "bc" */
    String t3 = s.substring(0, 2) + s.substring(2, 4);
    System.out.println(s.equals(t3)); /* true */
    for(int i = 0; i < s.length(); i ++) {
      System.out.print(s.charAt(i));
    }
    System.out.println();
  }
}
```



String S = "..."

$(0 \le i < S.length() - 1)$

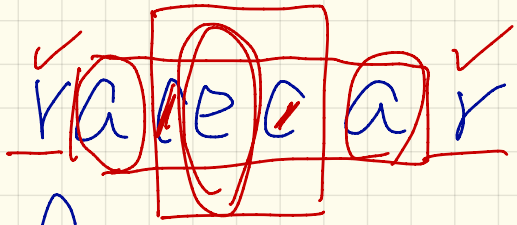
S.substring(0, i) + S.substring(i, S.length() ~~-1~~)

S = "York" S.length() == 4

S.substring(0, 2) + S.substring(2, 4)

0 1 2 3

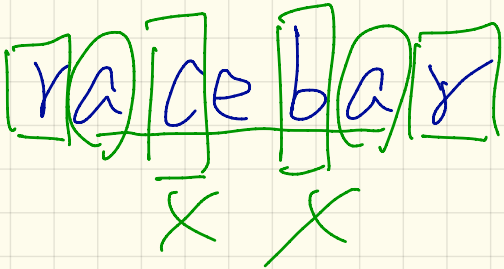
"Yo" + "rk"



Base Cases:

"" is P

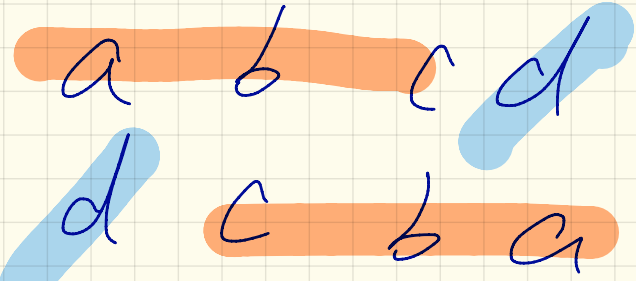
"_" is P



Recursive Cases:

reverse

input: a b c d



reverse(b c d) + a

output: d c b a

reverse ("") ""

reverse ("_") "_"