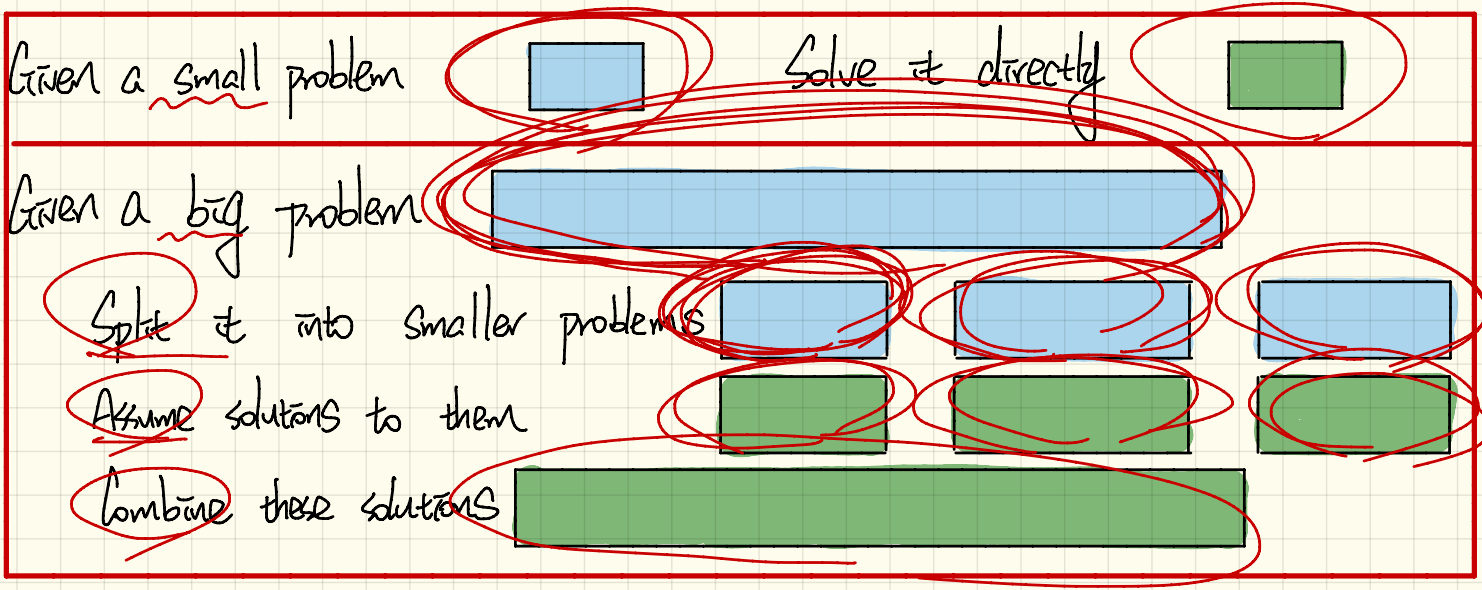
Wednesday Nov. 21

Lecture 21

Solving a Problem Recursively




```
m(i) {
  if(i == ...) { /* base case: do something directly */ }
  else {
    m(j);/* recursive call with strictly smaller value */
  }
}
```

subproblem $j < i$

Factorial

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n-1)! & \text{if } n \geq 1 \end{cases}$$

recursive solution to a smaller problem

small problem

original problem

combine

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

factorial (4)

Runtime Stack

V1

```
int fac(int n){
    int result;
    result = n * fac(n-1);
    return result;
}
```

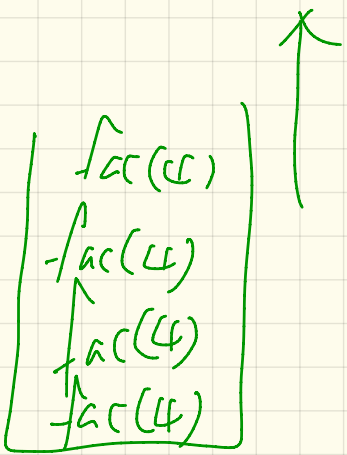
fac(4) fac(-2) fac(-1) fac(0) fac(1) fac(2) fac(3) fac(4)

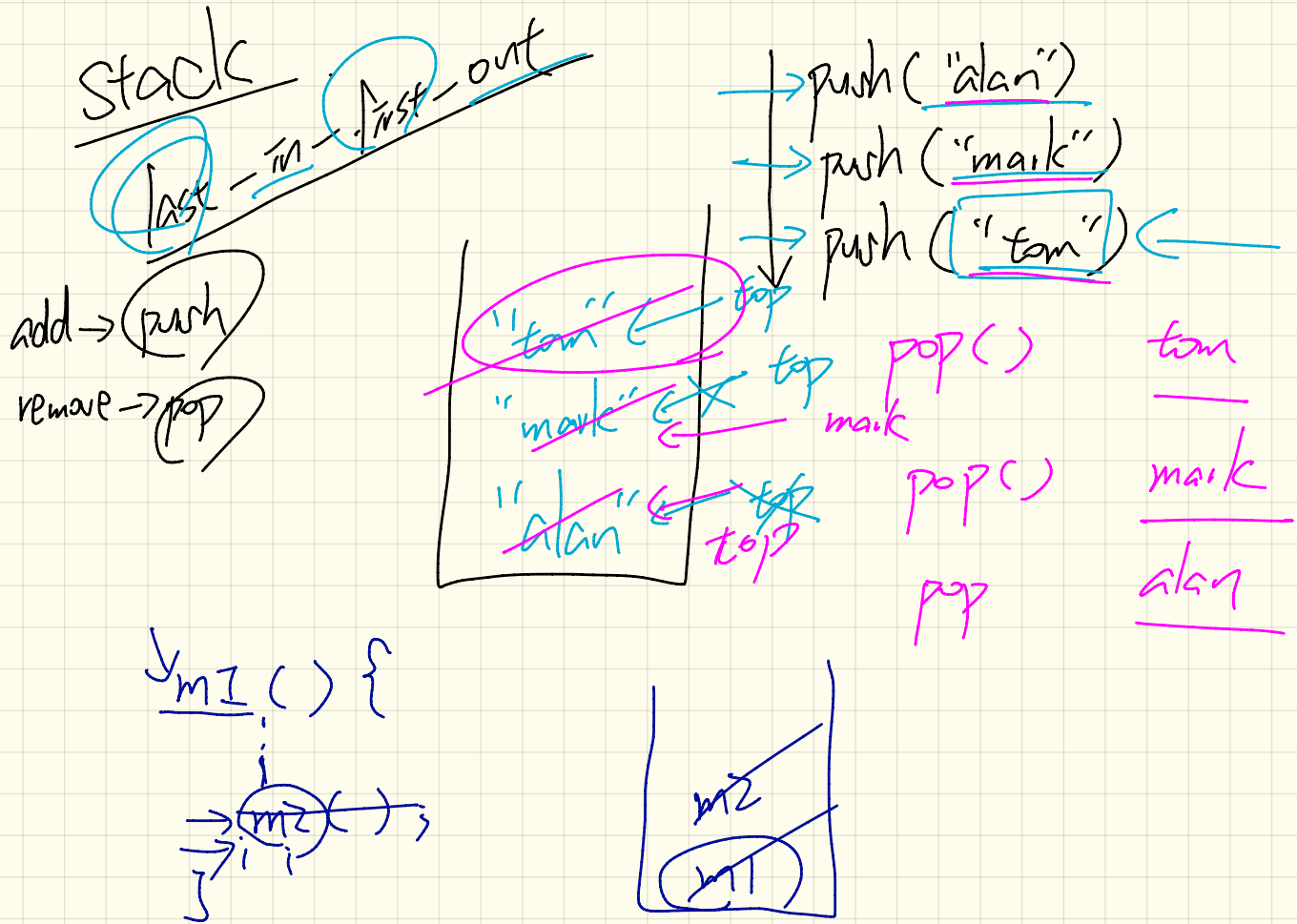
lack of base case

V2

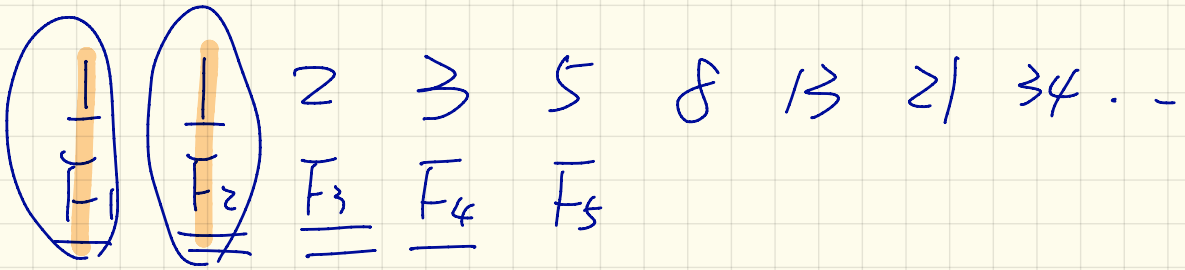
```
int fac(int n){
    int result;
    if(n == 0){ result = 1; }
    else { n * fac(n); }
}
```

fac(4) fac(4) fac(4) fac(4) fac(4)




stack
last - in - first - out
add -> push
remove -> pop
push("alan")
push("mark")
push("tom")
"tom" top
"mark" top
"alan" top
pop()
mark
pop()
pop
tom
mark
alan
void m1() {
m2();
}
m2
m1

F_n



F_n

F_n =

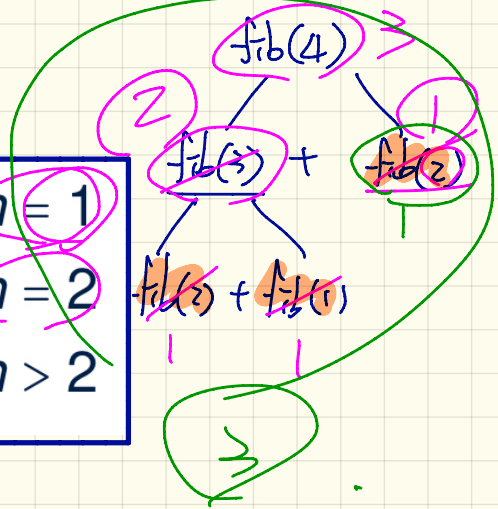
$$\begin{cases} 1 & \text{if } n == 1 \\ 1 & \text{if } n == 2 \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2 \end{cases}$$

F_{n-1} F_{n-2}

F_{n-2}

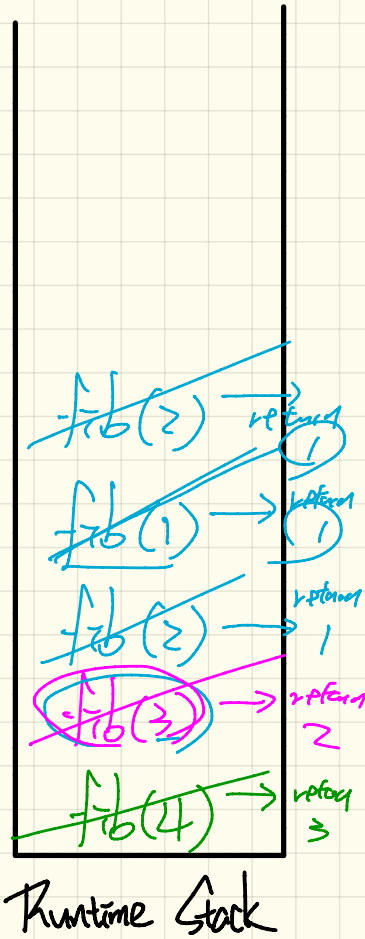
Fibonacci Number

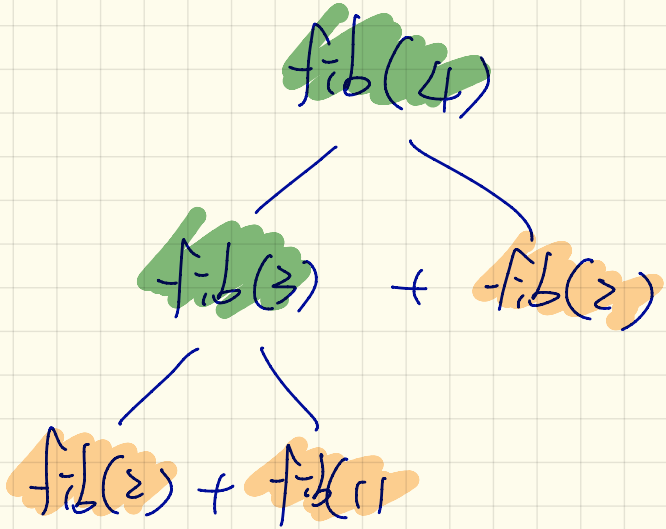
$$F_n = \begin{cases} 1 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \\ F_{n-1} + F_{n-2} & \text{if } n > 2 \end{cases}$$



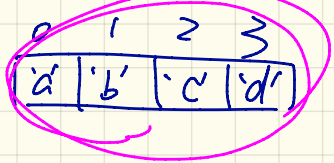

```
int fib (int n) {
  int result;
  if (n == 1) { /* base case */ result = 1; }
  else if (n == 2) { /* base case */ result = 1; }
  else { /* recursive case */
    result = fib (n - 1) + fib (n - 2);
  }
  return result;
}
```

fib(4)

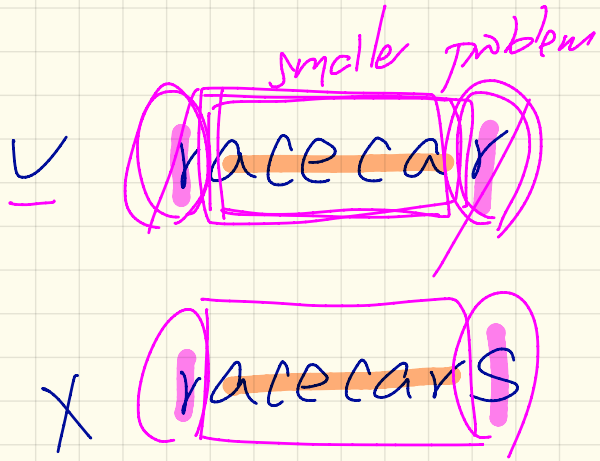



fib(4)
fib(3)
+
fib(2)
fib(2)
+
fib(1)

Use of String




```
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();
  }
}
```



✓

✗

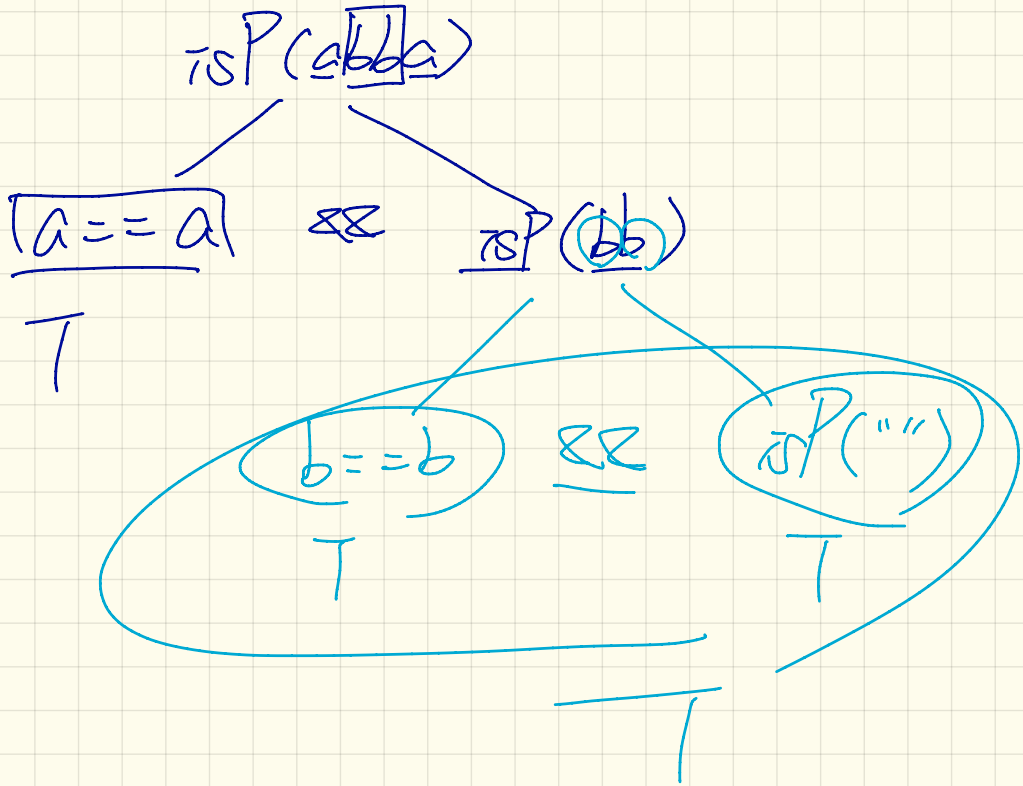
s

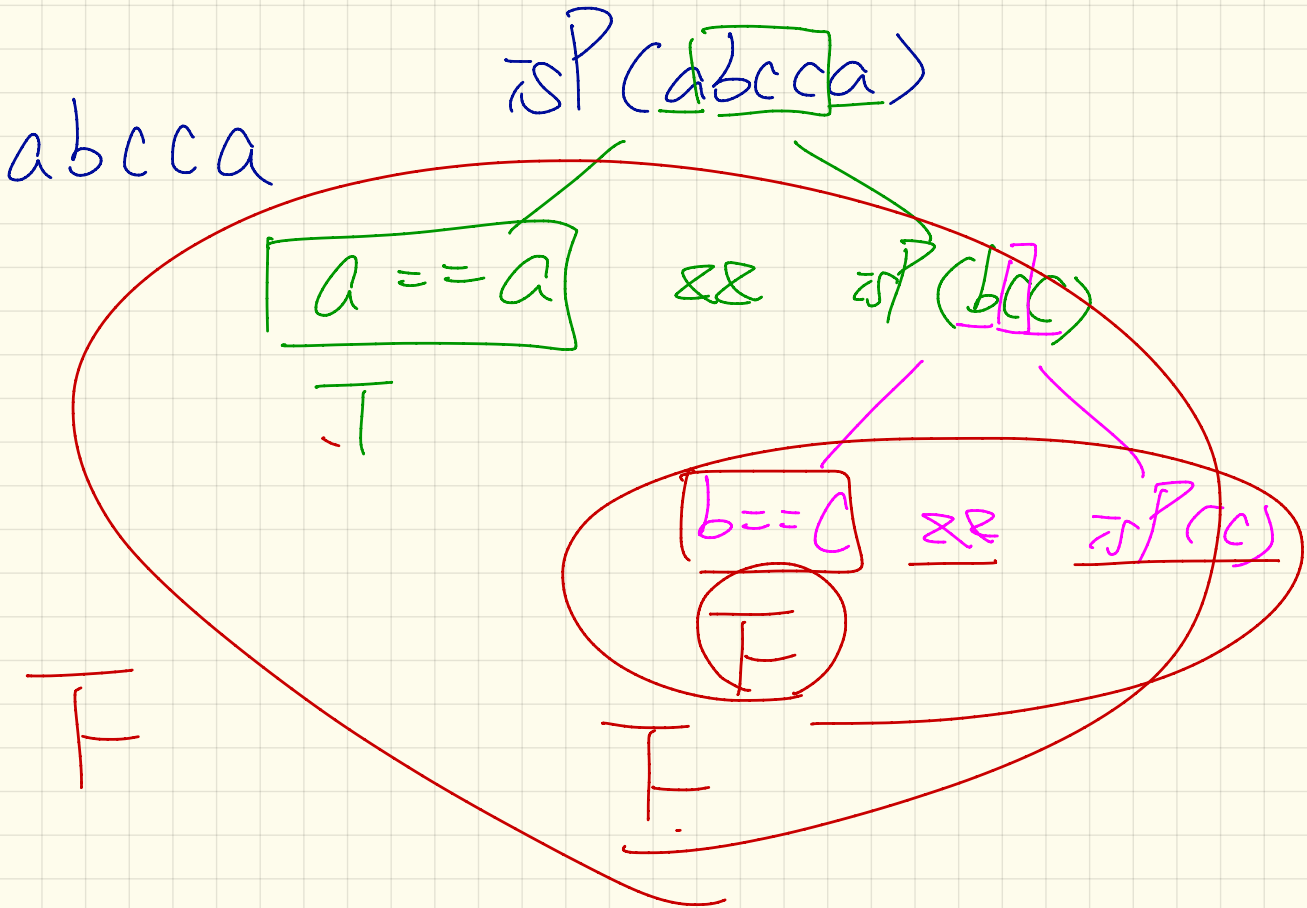
fc(s) == lc(s)

&&

isP(substring(———))

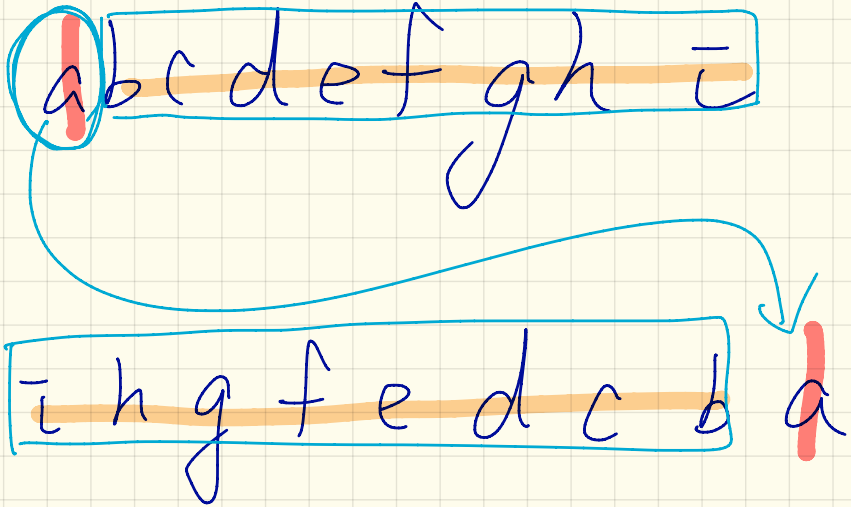
abba

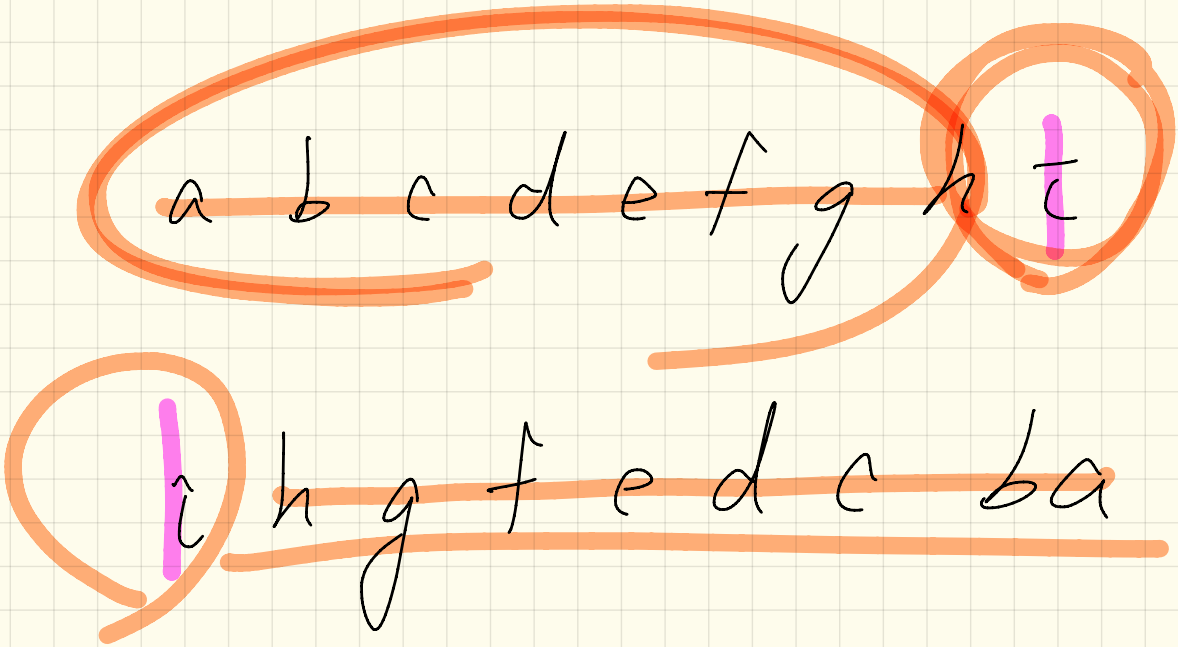



isP(abcca)
abcca
a==a
&&
isP(bcc)
T
b==c
&&
isP(c)
F
F
F

Palindrome

```
boolean isPalindrome(String word) {
  if(word.length() == 0 || word.length() == 1) {
    /* base case */
    return true;
  }
  else {
    /* recursive case */
    char firstChar = word.charAt(0);
    char lastChar = word.charAt(word.length() - 1);
    String middle = word.substring(1, word.length() - 1);
    return
        firstChar == lastChar
        /* See the API of java.lang.String.substring. */
        && isPalindrome(middle);
  }
}
```


a b c d e f g h i
i h g f e d c b a


a b c d e f g h i
i h g f e d c b a