Lecture 23 – Nov 26

Recursion

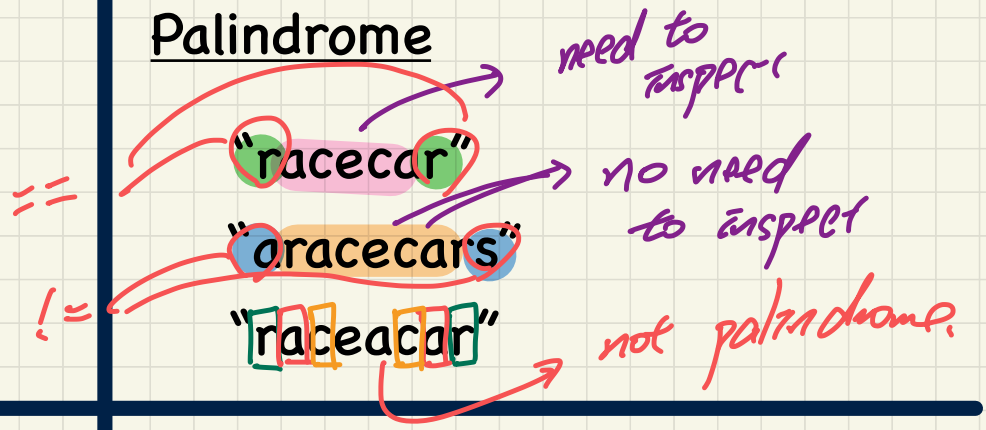
Examples: reverseOf, occurrencesOf
Recursion on Arrays: Creating Sub-Arrays
Recursion on Arrays: Call by Value

Announcements/Reminders

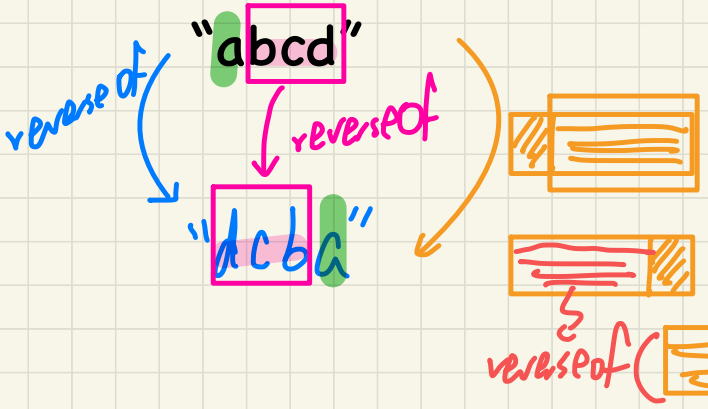
- Today's class: notes template posted
- Lab5 released (deadline: Dec 2; grace period until Dec 9)
  + Required background study: abstract classes & interfaces

Recursions on Strings

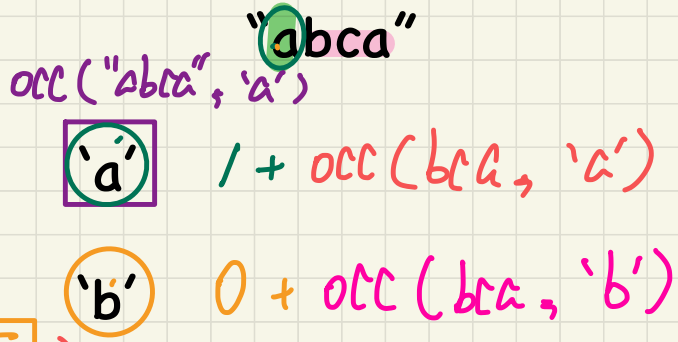
Palindrome



Reversal



Number of Occurrences



Problem: Palindrome

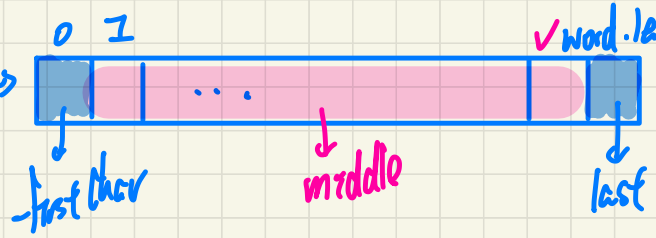
```
boolean isPalindrome (String word) {
  if(word.length() == 0 || word.length() == 1) {
    /* base case */
    return true;
  }
  else {
    /* recursive case */
    char firstChar = word.charAt(0);
    char lastChar = word.charAt(word.length() - 1);
    String middle = word.substring(1, word.length()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

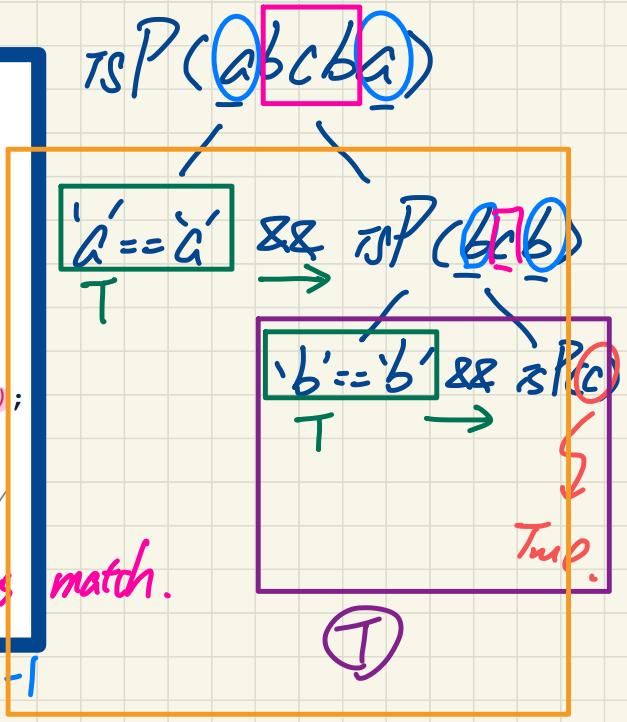
↳ [1, word.length()-2]

1. LHS is false
↳ skip RHS

2. LHS is true
↳ eval RHS

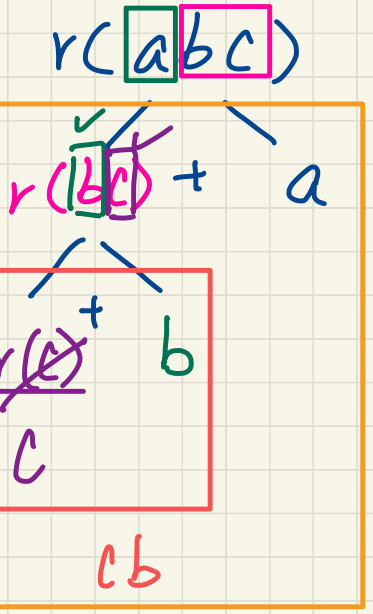
↳ RC. made only if first & last chars match.

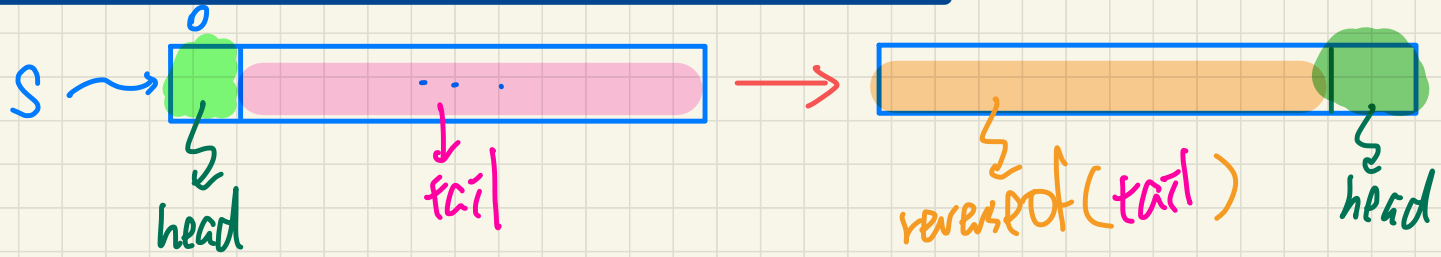




Problem: Reverse of a String

```
String reverseOf (String s) {
  if (s.isEmpty()) { /* base case 1 */
    return "";
  }
  else if (s.length() == 1) { /* base case 2 */
    return s;
  }
  else { /* recursive case */
    String tail = s.substring(1, s.length());
    String reverseOfTail = reverseOf (tail);
    char head = s.charAt(0);
    return reverseOfTail + head;
  }
}
```





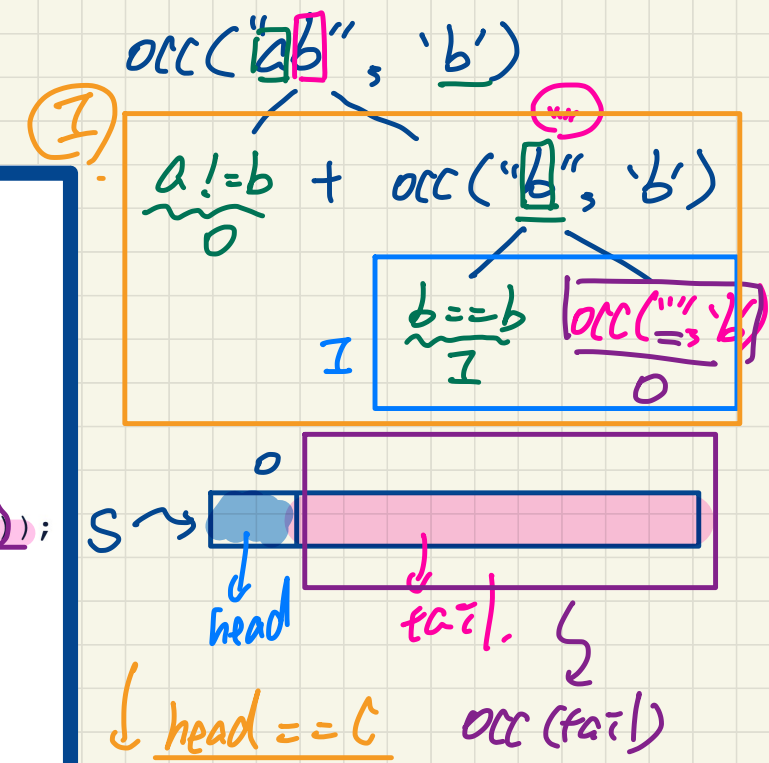
Exercise

a b c d

d c b a

Problem: Number of Occurrences

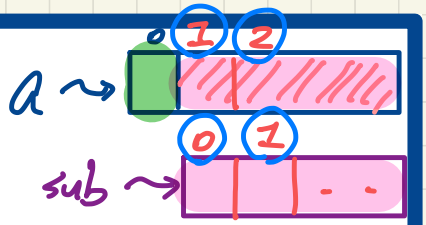
```
int occurrencesOf(String s, char c) {
  if(s.isEmpty()) {
    /* Base Case */
    return 0;
  }
  else {
    /* Recursive Case */
    char head = s.charAt(0);
    String tail = s.substring(1, s.length());
    if(head == c) {
      return 1 + occurrencesOf(tail, c);
    }
    else {
      return 0 + occurrencesOf(tail, c);
    }
  }
}
```



"b"
head
tail: substring(1, 1)
""

Recursion on an Array: Passing new Sub-Arrays

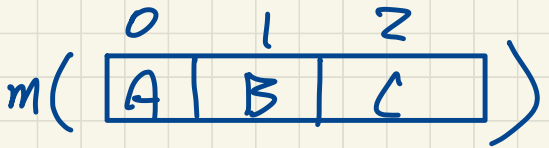
```
void m(int[] a) {
  if(a.length == 0) { /* base case */ }
  else if(a.length == 1) { /* base case */ }
  else {
    int[] sub = new int[a.length - 1];
    for(int i = 1; i < a.length; i ++) { sub[i - 1] = a[i]; }
    m(sub) } }
```



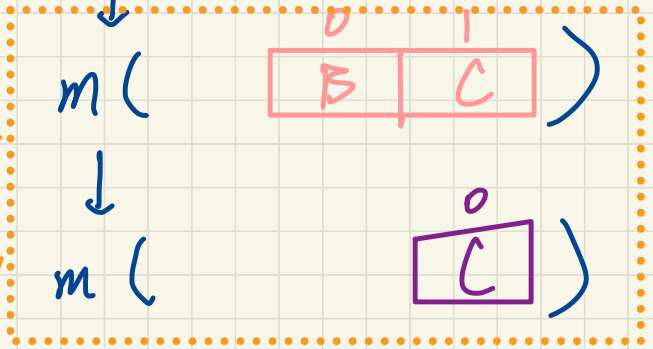
Copy every thing except the head.

Say a1 = {}, consider m(a1)

↳ base case # 1



Say a2 = {A, B, C}, consider m(a2)



time & space wasted in copying some contents over & over