

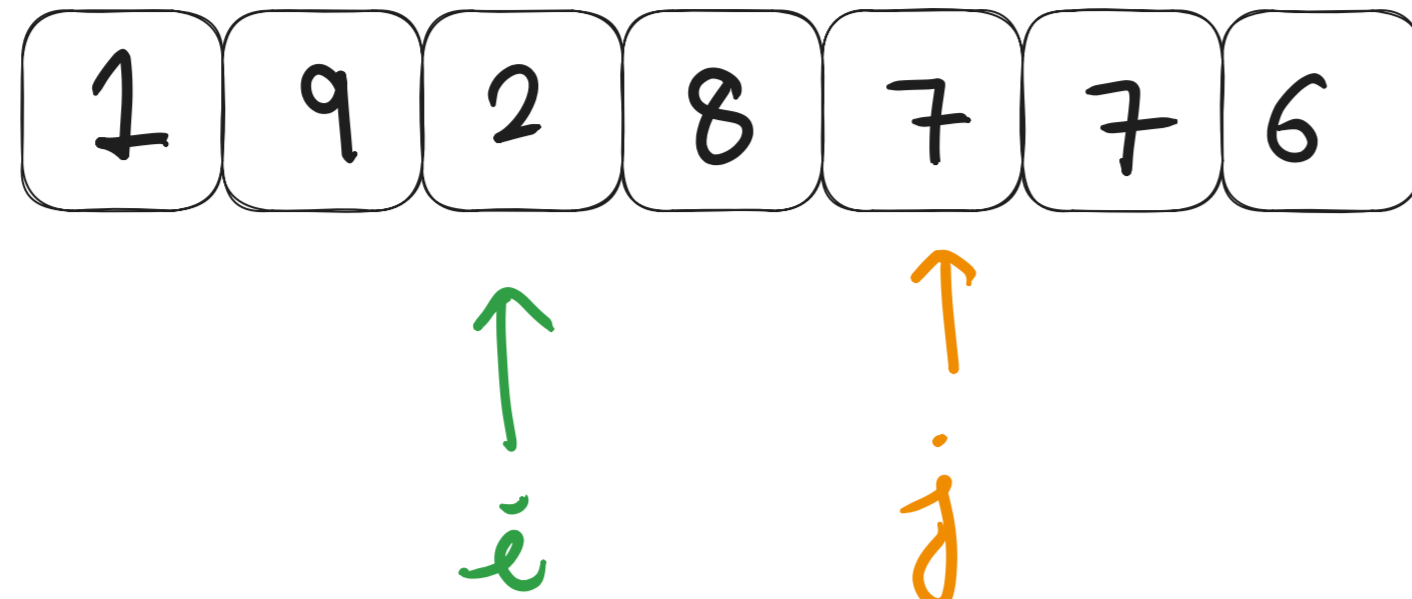
Babua DSA Patterns Course

Day 2/90

Comment "Done"

Two Pointer Pattern

Linear Data structure (Array, String, Linked List)
Two variables (or) Two Pointers



125. Valid Palindrome

LeetCode Easy

344. Reverse String

LeetCode Easy

977. Squares of a Sorted Array

LeetCode Easy

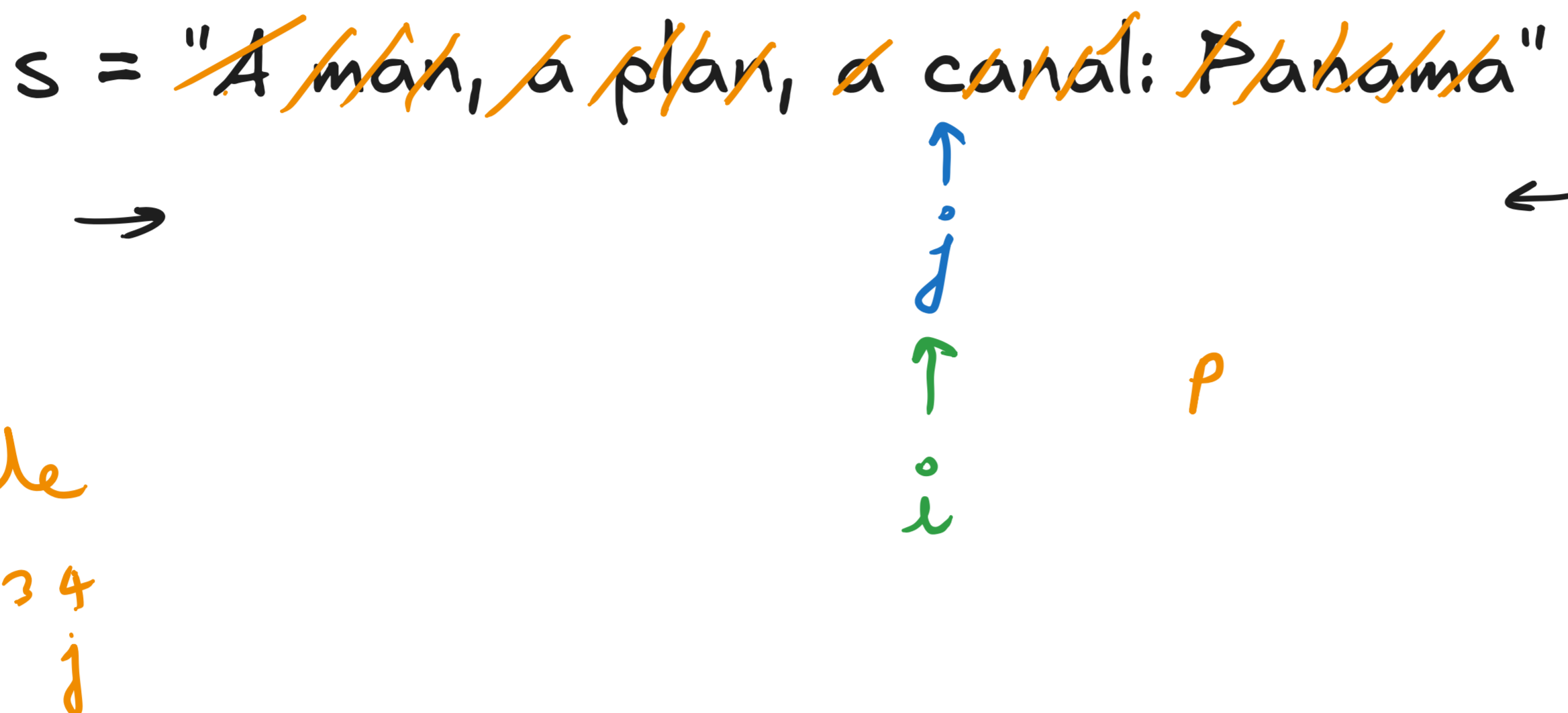
680. Valid Palindrome II

LeetCode Easy

Valid Word Abbreviation

NeetCode Easy

125. Valid Palindrome



T: O(n)
S: O(1)

```
class Solution {
    public boolean isPalindrome(String s) {
        int i = 0;
        j = s.length() - 1;

        // skip spaces
        // left = a
        // right = p

        // T: O(n), S: O(1)
        while (i < j) {
            char left = s.charAt(i);
            char right = s.charAt(j);

            if (!Character.isLetterOrDigit(left)) {
                i++;
                continue;
            }

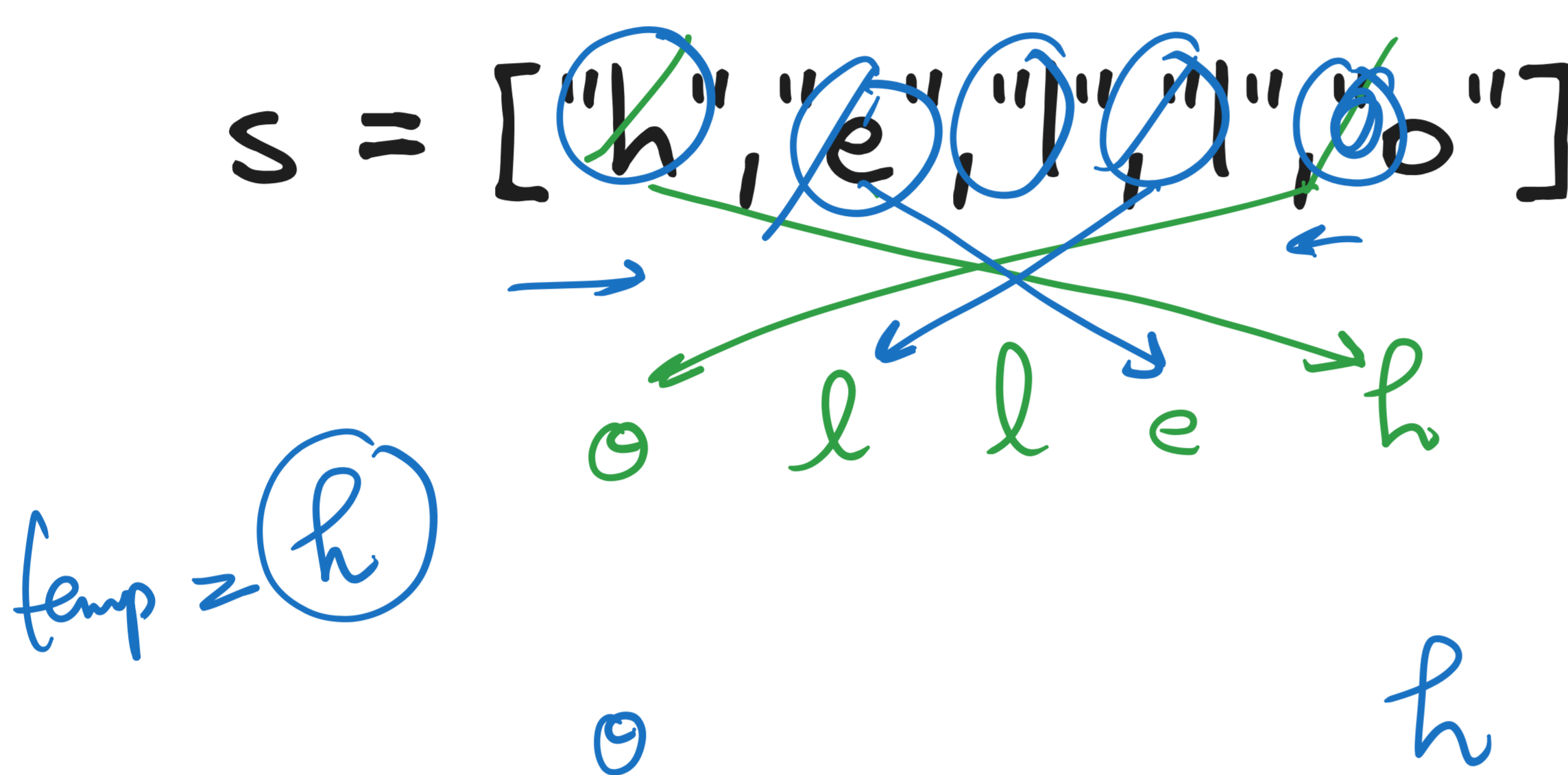
            if (!Character.isLetterOrDigit(right)) {
                j--;
                continue;
            }

            if (Character.toLowerCase(left) != Character.toLowerCase(right)) {
                return false;
            }

            i++;
            j--;
        }

        return true;
    }
}
```

344. Reverse String



T: O(n)
S: O(1)

```
class Solution {
    public void reverseString(char[] s) {
        int i = 0;
        j = s.length - 1;

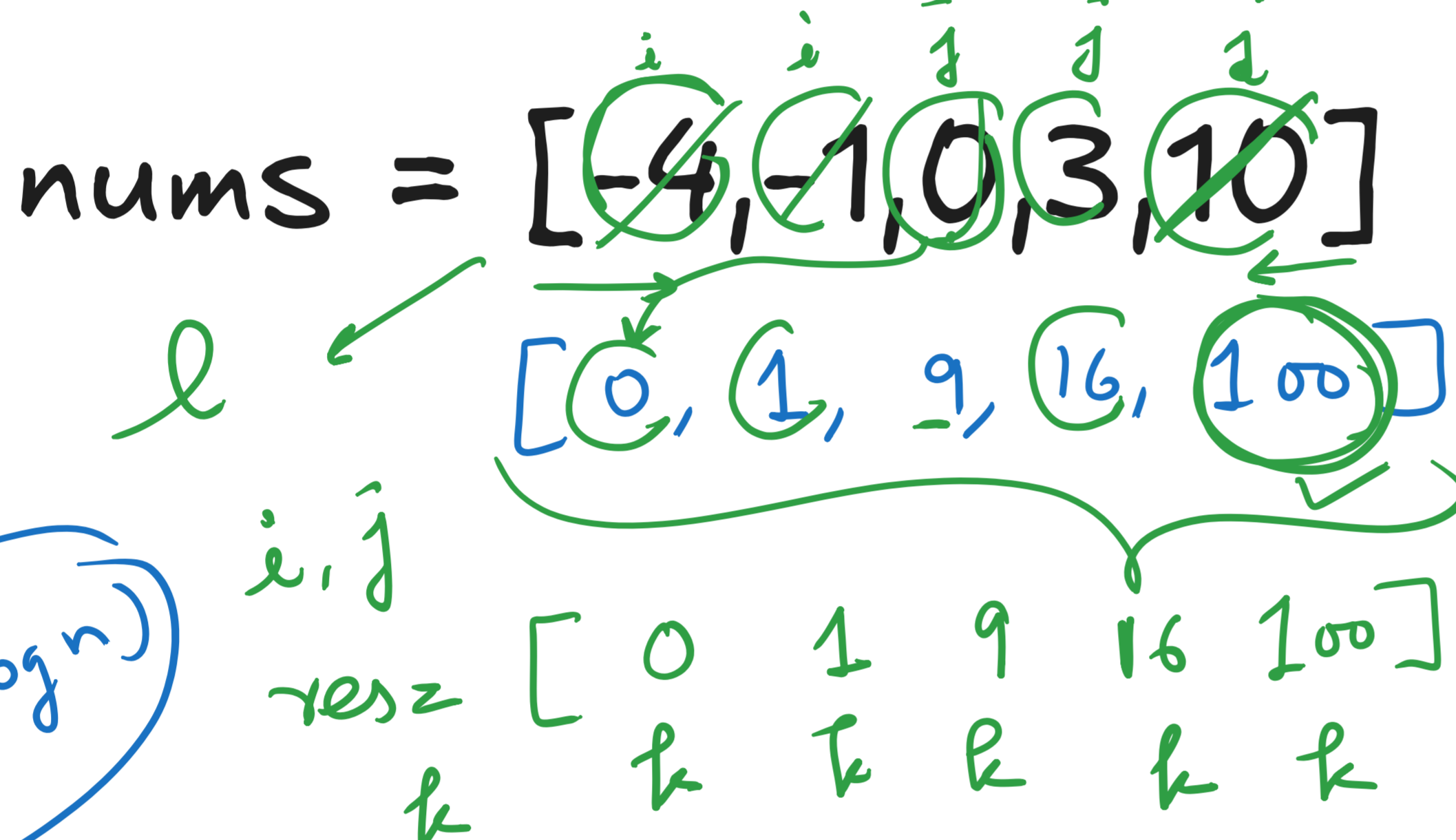
        // o l l e h
        // i
        // j
        // temp = e

        while (i < j) {
            char temp = s[i];
            s[i] = s[j];
            s[j] = temp;

            i++;
            j--;
        }

        // T: O(n)
        // S: O(1)
    }
}
```

977. Squares of a Sorted Array



T: O(n log n)
S: O(1)

```
class Solution {
    public int[] sortedSquares(int[] nums) {
        int i = 0;
        j = nums.length - 1;

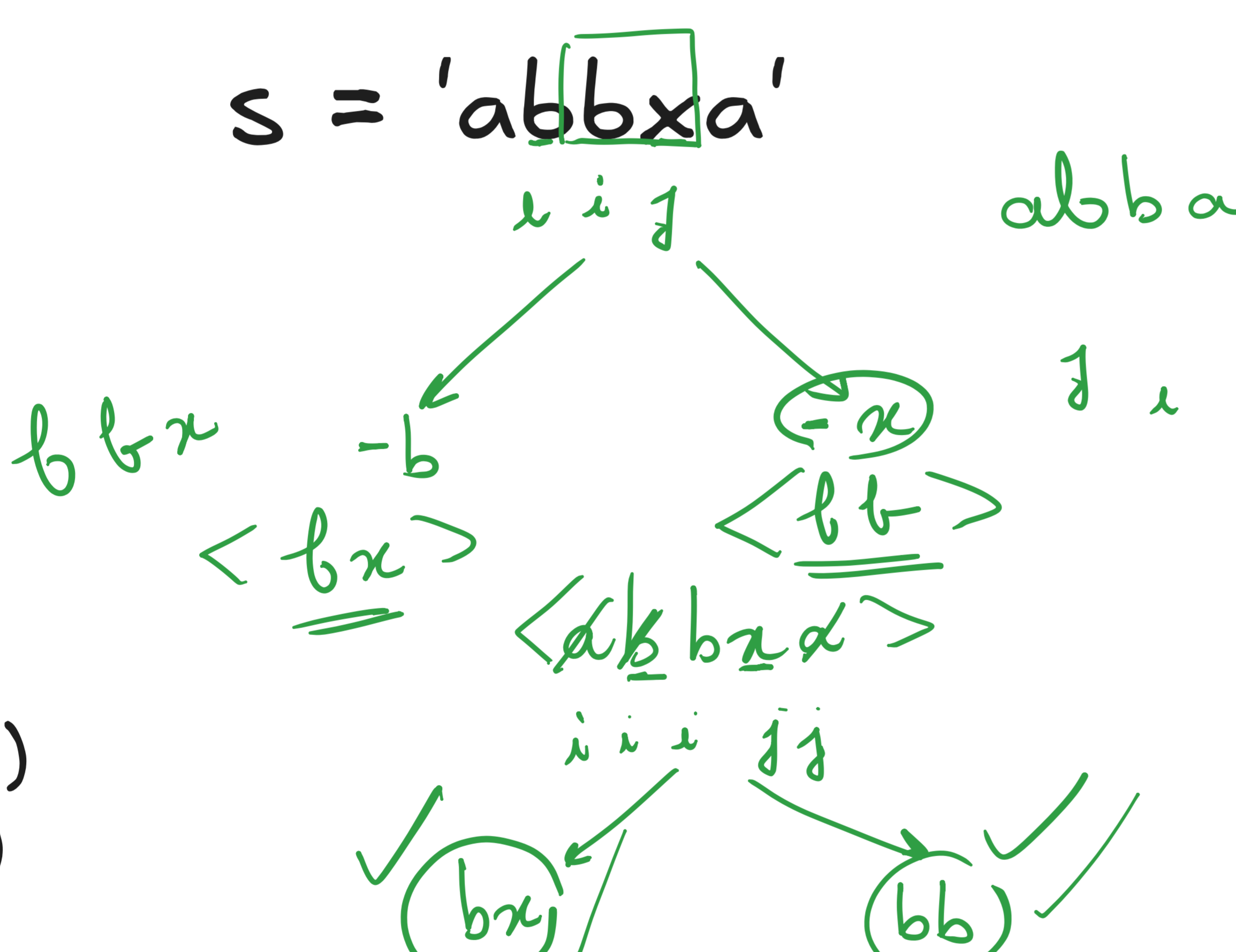
        int k = nums.length - 1;

        // [-3, 2, 4]
        // i
        // j
        // [ 4 9 16]
        // k

        // T: O(n), S: O(1)
        while (i <= j) {
            if (Math.abs(nums[i]) > Math.abs(nums[j])) {
                res[k] = nums[i] * nums[i];
                i++;
            } else {
                res[k] = nums[j] * nums[j];
                j--;
            }
            k--;
        }

        return res;
    }
}
```

680. Valid Palindrome II



T: O(n)
S: O(1)

```
class Solution {
    public boolean palindromeHelper(int i, int j, String s) {
        while (i < j) {
            if (s.charAt(i) != s.charAt(j)) {
                return false;
            }

            i++;
            j--;
        }

        return true;
    }

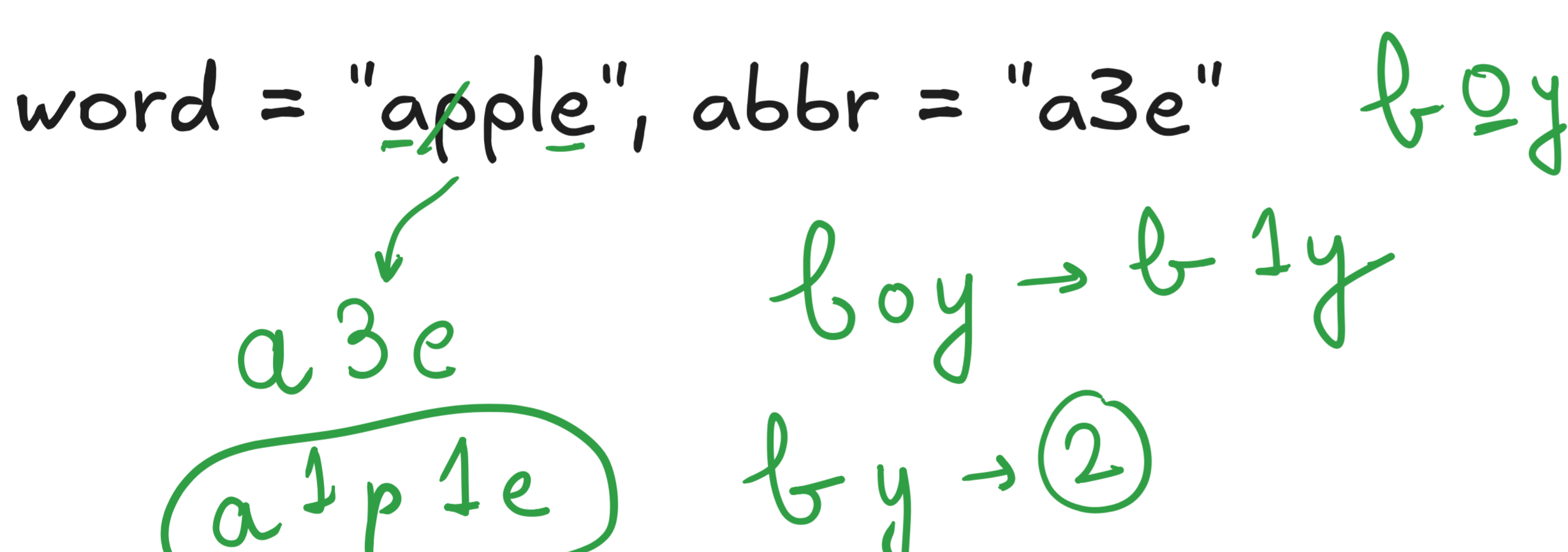
    // T: O(n), S: O(1)
    // skip
    // i
    // j
    public boolean validPalindrome(String s) {
        int i = 0;
        j = s.length() - 1;

        while (i < j) {
            char left = s.charAt(i);
            char right = s.charAt(j);

            if (left != right) {
                // use our helper
                return palindromeHelper(i + 1, j, s) || palindromeHelper(i, j - 1, s);
            } else {
                i++;
                j--;
            }
        }

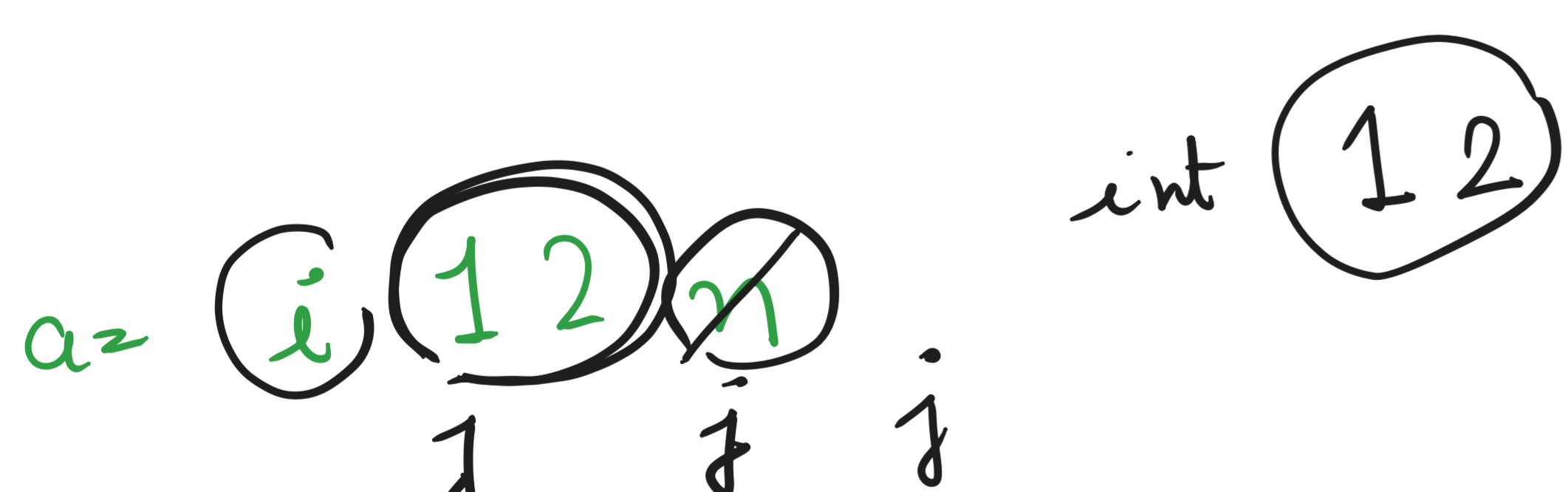
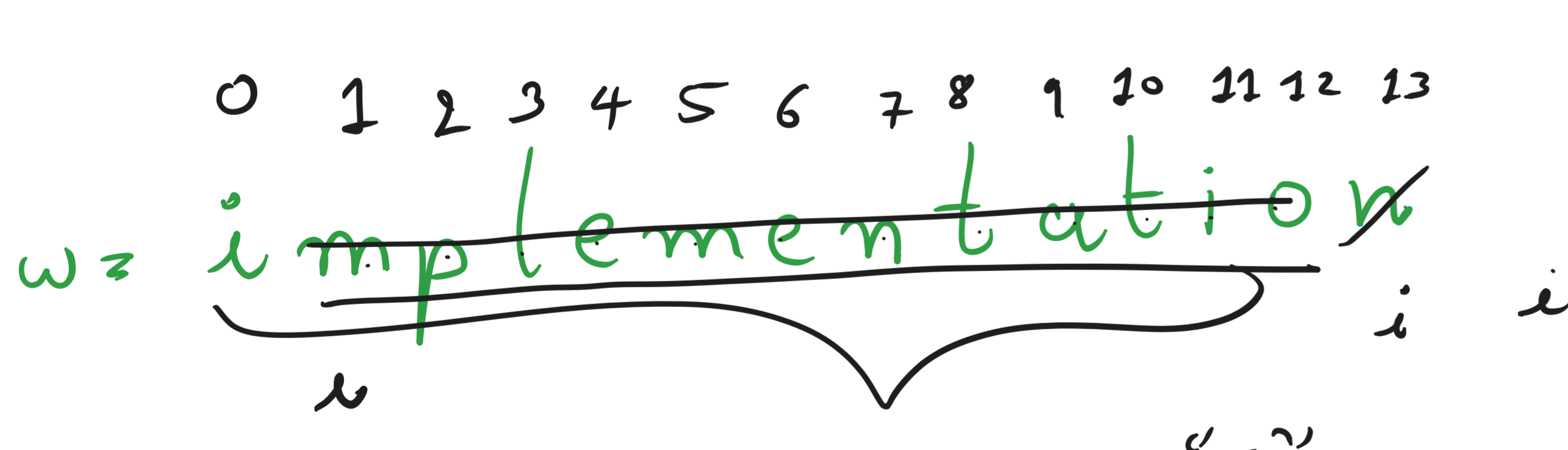
        return true;
    }
}
```

Valid Word Abbreviation



word = "international", abbr = "i9l"

word = "international", abbr = "i12l"



T:

S:

0 1 2 3 4

a p p l e

a 3 e

a 1 p 1 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

a 4 e

false

(4 1)

n = 40

(4 1)

(4 1)

(4 1)

(4 1)

(4 1)

(4 1)

(4 1)

(4 1)

(4 1)

(4 1)

(4 1)

(4 1)

```
class Solution {
    public boolean validWordAbbreviation(String word, String abbr) {
        int i = 0;
        j = 0;

        // apple
        // a34
        // i
        // j
        // w_c = p
        // a_c = 3
        // curr = 3
        // T: O(n + a), S: O(1)
        while (i < word.length() && j < abbr.length()) {
            char w_c = word.charAt(i);
            char a_c = abbr.charAt(j);

            if (Character.isDigit(a_c)) {
                if (a_c == '0') {
                    return false;
                }

                int curr = 0;

                while (j < abbr.length() && Character.isDigit(abbr.charAt(j))) {
                    curr = curr * 10 + (abbr.charAt(j) - '0');
                    j++;
                }

                i += curr;
            } else {
                if (w_c != a_c) {
                    return false;
                }
            }

            i++;
            j++;
        }

        return i == word.length() && j == abbr.length();
    }
}
```

// T: O(m + n), S: O(1)