Tips 💡

Data Structures and Algorithms (DSA) Tips for LeetCode

1. Strings

Note

1. Always return a modified string rather than modifying in-place (which doesn't change the string at all).

```
void helper(String s){
    s = s + "hello";    // doesn't change the string at all
}

String helper(String s){
    return s + "hello"; // this modifies the string as we returning it
}
```

2. Arrays.sort() won’t work on Strings as it isn’t an array

3. You can lexicographically sort the strings if they are stored in an array

```
List<String> list = new ArrayList<>();
// add elements to the list

Collections.sort(list);

// the first one will be the lexiographically the smallest string present
```

Basic String operations

1. To convert a number/character to a string:
This applies to number or character array as well

```
String string = String.valueOf(number);
```

```
char[] array = string.toCharArray();

String s = String.valueOf(array);
```

2. To convert string to a number:

```
Integer val = Integer.parseInt(String s)
```

3. To convert string into an array of characters

```
for(char c : string.toCharArray()){
    // your code
```

```
}
```

4. For comparing 2 strings use

```
if(s1.equals(s2)){
   // your code
}
```

5. Substring method

```
s.substring(int s,int e)

// e is exclusive
```

Important String Builder Methods

1. To set a character

```
StringBuilder sb = new StringBuilder("Navneett");
sb.setCharAt(7,'h');
```

2. To add new character/string/number in the end

```
StringBuilder sb = new StringBuilder("Navneet");
sb.append('h')
```

3. To delete a character

```
StringBuilder sb = new StringBuilder("Navneeth K SS");
sb.deleteCharAt(12);
```

4. To reverse a string

```
// How to the built in reverse method
StringBuilder sb = new StringBuilder("Navneeth K S");
sb.reverse();
```

2. Character

Basic Character Operations

1. To convert a character to a number

```
int value = Character.getNumericValue(character);
```

2. To check if a character is a digit or not

```
Character.isDigit(ch) // return true if yes
```

3. To convert number to a character

```
int number = 9;
Character ch = (char) (number + '0');
```

4. To convert ascii number to character

```
int number = 67;
char ch = (char) number;
```

5. To check if a character is number or a letter

```
Character.isLetterOrDigit(c) // return true if yes
```

6. ASCII starting values

```
a : 97
A : 65
```

7. To check if a character is a space

```
Character.isSpaceChar(char); // return a boolean value
```

3. Arrays

Basic Array Operations

1. Sort a array

```
int[] nums = new int[10];
Arrays.sort(nums);

// method overloaded looks like
Arrays.sort(nums,start,end);
// it sorts from start to end - 1
```

2. To copy the array

```
int[] nums = new int[100];

int[] newArray = Arrays.copyOfRange(nums, fromIndex, toIndex);
```

3. To fill an array

```
int[] nums = new int[size];
```

```
Arrays.fill(nums,0);
```

4. Directly return an array

```
public int[] fill(int n){
    // logic
    return new int[] {1,2,3};
}
```

5. Check if 2 arrays are equal

```
boolean ans = Arrays.equals(a,b)
```

6. To clone an array

```
public void fun(int[] nums){
    int[] newClone = nums.clone();
}
```

4. Hash Maps

Note

1. It doesn't guarantee to the store the key-value pairs in a particular order

1. How to iterate through a hash map ?

```
for(Map.Entry<Integer,Integer> entry : map.entrySet()){
    System.out.println(entry.getKey(), entry.getValue())
}
```

2. If a map contains a element or not

```
if(map.containsKey(key)) // returns true if yes
if(map.containsValue(value)) // returns true if yes
```

3. To reset the HashMap

```
HashMap<Integer,Integer> map = new HashMap<>();

map.clear();
```

4. To remove a character from the map

```
HashMap<Integer,Integer> map = new HashMap<>();
map.put(1,2);
```

```
map.remove(1);    // Removes 1
```

5. To compare if 2 maps are equals

```
map1.equals(map2) // return true;
```

5. Lists

1. To set a object inside the list

```
List<Integer> list = new ArrayList<>();
list.add(0);
list.add(1);
list.add(2);
list.set(2,4);
```

2. To remove an element it is

```
list.remove(i) // i is the index
```

3. To clear the elements of the list

```
list.clear() // Makes the list empty
```

4. To retrieve the size of the list

```
list.size() // return the size
```

5. To convert the list into array

```
list.toArray() // to convert this into array
```

6. To check if the list contains a contains an element

```
list.contains(e) // return either true or false
```

6. Stacks

1. Add a new element

```
Stack<Integer> stack = new Stack<>();
stack.push(10);
```

2. Removing an element

```
stack.pop(); // return the element
```

7. Queue using Linked List implementation

1. To add a new element

```
Queue<Node> queue = new LinkedList<>();

queue.offer(root);
```

2. To remove the element

```
queue.poll(); // Return it
```

3. To peek

```
queue.peek()
```

8. Set

In Java, a **Set** is a collection that **does not allow duplicate elements**. It is part of the **Java Collections Framework** and is useful when we need to store unique values and perform operations like **union, intersection, and difference** efficiently.

1. Adding elements:

```
Set<Integer> numbers = new HashSet<>();
numbers.add(10);
numbers.add(20);
numbers.add(20);  // Duplicate ignored
System.out.println(numbers);  // Output: [10, 20]
```

2. Removing element:

```
Set<String> fruits = new HashSet<>();
fruits.add("Apple");
fruits.add("Banana");

fruits.remove("Apple");
System.out.println(fruits);  // Output: [Banana]

fruits.clear();
System.out.println(fruits);  // Output: []
```

3. Checking elements

```
Set<String> names = new HashSet<>();
names.add("Alice");
names.add("Bob");

System.out.println(names.contains("Alice"));  // Output: true
System.out.println(names.contains("Eve"));    // Output: false
```

4. Set operations

```
Set<Integer> set1 = new HashSet<>(Arrays.asList(1, 2, 3, 4));
Set<Integer> set2 = new HashSet<>(Arrays.asList(3, 4, 5, 6));

// Union
set1.addAll(set2);
System.out.println(set1);  // Output: [1, 2, 3, 4, 5, 6]

// Intersection
set1.retainAll(set2);
System.out.println(set1);  // Output: [3, 4, 5, 6]

// Difference
set1.removeAll(set2);
System.out.println(set1);  // Output: []
```

ADDITIONAL TIPS

1. If there a input present in the null, how do you read it ?

```
// example:

inp = 1 2 3 null 4 null 5

// go for String input intead of int

String inp = sc.nextLine()

if(inp.equals("null")){
   // do smthg
}else{
   //
}
```