# How to improve

Uwi Tenpen

https://www.hackerrank.com/uwi?hr\_r=1

"Many coders ask me how to improve their own performances. I cannot say anything except solve and review and prepare your library".

# standard maven directory layout

<https://stackoverflow.com/questions/41638654/java-project-folder-structure-in-intellij-idea>

That is the basic folder structure of a maven project. IntelliJ usually recognizes this and sets up sensical defaults for you. If it didn't (or if it did but you modified them afterwards), you need to set up your java folder as the sources folder (i.e. the folder that contains the source code).

[] .idea

[] src

[] main

[] java

com.simpleproject

SimpleClass.java

[] resources

[] test

[] java

com.simpleproject

SimpleClassTest.java

[] resources

# Guide to naming conventions on groupId, artifactId and version

<https://maven.apache.org/guides/mini/guide-naming-conventions.html>

groupId will identify your project uniquely across all projects, so we need to enforce a naming schema. It has to follow the package name rules, what means that has to be at least as a domain name you control, and you can create as many subgroups as you want. Look at More information about package names.

eg. org.apache.maven, org.apache.commons

A good way to determine the granularity of the groupId is to use the project structure. That is, if the current project is a multiple module project, it should append a new identifier to the parent's groupId.

eg. org.apache.maven, org.apache.maven.plugins, org.apache.maven.reporting

artifactId is the name of the jar without version. If you created it then you can choose whatever name you want with lowercase letters and no strange symbols. If it's a third party jar you have to take the name of the jar as it's distributed.

eg. maven, commons-math

version if you distribute it then you can choose any typical version with numbers and dots (1.0, 1.1, 1.0.1, ...). Don't use dates as they are usually associated with SNAPSHOT (nightly) builds. If it's a third party artifact, you have to use their version number whatever it is, and as strange as it can look.

eg. 2.0, 2.0.1, 1.3.1

# Project related setting that need to be in sink

Changes to the pom.xml file may result in the following error messages.

- Error message "Error:(31, 40) java: diamond operator is not supported in -source 1.5

(use -source 7 or higher to enable diamond operator)

- Error message "Error:java: javacTask: source release 1.8 requires target release 1.8"

Sometimes happens when change scope in pom file between compile and test.

<scope>compile</scope>

<scope>test</scope>

Do both #1 and #2 to fix the above error messages.

1. <project>.iml

- <component name="NewModuleRootManager" LANGUAGE\_LEVEL="JDK\_1\_5" inherit-compiler-output="false">

LANGUAGE\_LEVEL =" JDK\_1\_8" (not "JDL\_1\_5")

2. Intellij IDE | Preferences | Build , Execition, Deployment | Compiler | Java Compiler

- Target bytecode version = 1.8 (not 1.5)

3. pom.xml

<dependencies>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.11</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.json/json -->

<dependency>

<groupId>org.json</groupId>

<artifactId>json</artifactId>

<version>20170516</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.5.3</version>

</dependency>

</dependencies>

<properties>

<maven.compiler.target>1.8</maven.compiler.target>

<maven.compiler.source>1.8</maven.compiler.source>

</properties>

**testing v6.11 wants**

- com.beust:jcommander:1.64

- org.yaml:snakeyaml:1.17

**json v20170516**

- org.json:json:20170516

**httpclient v4.5.3**

- org.apache.httpcomponents:httpclient:4.5.3

- org.apache.httpcomponents:httpcore:4.4.6

- commons-codec:commons-logging:1.2

- commons-codec:commons-codec:1.9

*Note: Reported but I didn’t need to do following: If scope is set to "test" and problem, try "compile" and see if go away.*

4. File | Project Structure | Modules | Dependencies

- If scope is set to "test" might be a problem. Try "compile" and see if go away.

- testing Scope = Compile (not Test)

*Note: For #4, it was reported but I didn't need to do it.*

# Project files should not be checked in

<https://softwareengineering.stackexchange.com/questions/202051/what-should-i-include-in-my-repositiory-from-ide-projects>

When more people are involved, I don't put these things in source control. On my team, we have a mix of IntelliJ, Sublime Text, and Eclipse being used. IDE files just add clutter, and result in pulling in commits to those files from other people for an IDE you don't use.

Also, your project shouldn't be dependent on the IDE anyway. A build server won't be booting up Eclipse to compile your product, so it should already be IDE-free. A more minor point: it eliminates personal organization within the project. For example, in IntelliJ I like to use many modules within our project. No one else using IntelliJ worries about this because we don't store the .iml (module) files.

For Java, there is such a thing: Maven. It imposes strong conventions on the project's structure and allows you to specify project metadata (such as library dependencies) in one point (a file called pom.xml which is in the project's main directory and, of course, in source control). There are plugins which then create IDE project files from the Maven configuration, and others which can automate the project build process, or do just about anything. Sometimes it feels like it makes things unneccessarily complex and it takes some time to learn, but the benefits are generally worth it.

# How to create .ignore file

<https://stackoverflow.com/questions/10744305/how-to-create-gitignore-file>

MacOS / Linux one-liner

An easy way to get a default git ignore without messing about with create/copy/paste is to use the curl command from the terminal. First cd into your projects root directory and then run the command by replacing MY\_API\_NAME with your API name from one of the following two sources:

gitignore.io

curl -o .gitignore https://www.gitignore.io/api/MY\_API\_NAME

You can find your API name by searching from the list here and clicking Generate. (<https://www.gitignore.io/> )

Java Example:

curl -o .gitignore https://www.gitignore.io/api/java

GitHub

Alternatively you can use the ones at GitHub. Find the filename for your API here.

curl -o .gitignore https://raw.githubusercontent.com/github/gitignore/master/MY\_API\_NAME.gitignore

Java Example:

curl -o .gitignore https://raw.githubusercontent.com/github/gitignore/master/Java.gitignore

# number -> string -> array -> string -> number

package practice;

import static org.apache.commons.lang3.StringUtils.join;

import java.util.Arrays;

public class Sandbox

{

public static void main(String[] arguments)

{

// declaration and initialization

int number = 1024;

String string = "1024";

String[] array = new String[] {"1", "0", "2", "4"};

// print

System.out.println(number);

System.out.println(string);

System.out.println(Arrays.toString(array));

System.out.println(join(array, ""));

// length

System.out.println(Integer.SIZE);

System.out.println(string.length());

System.out.println(array.length);

// assignment

number = 5678;

string = "5678";

array[0] = "5"; array[1] = "6"; array[2] = "7"; array[3] = "8";

// get the 4th character

System.out.println(string.charAt(3));

System.out.println(array[3]);

// set the 4th character

array[3] = "9";

// number = 2,147,483,647 = 2^31 - 1 = MAX\_VALJUE;

// number -> string -> string array -> string -> number

/\* number \*/ int num = 2147483647;

/\* number -> string \*/ String str = Integer.toString(num);

/\* string -> string array \*/ String[] arr = str.split("");

/\* string array -> string \*/ str = join(arr, "");

/\* string -> number \*/ num = Integer.parseInt(str);

// number -> string -> char array -> string -> number

/\* number \*/ int num2 = 2147483647;

/\* number -> string \*/ String str2 = Integer.toString(num2);

/\* string -> char array \*/ char[] arr2 = str.toCharArray();

/\* char array -> string \*/ str2 = String.valueOf(arr2);

/\* string -> number \*/ num2 = Integer.parseInt(str2);

}

}

# Imports

import java.util.\*;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

import static org.apache.commons.lang3.StringUtils.\*;

import org.testng.annotations.\*;

import static org.testng.Assert.\*;

or

import static org.testng.AssertJUnit.\*;

# String

*String is a Java class.*

String greeting = "Hello World!";

# String Comparison

== tests for reference equality (whether they are the same object).

.equals() tests for value equality (whether they are logically "equal").

Objects.equals() checks for nulls before calling .equals() so you don't have to (available as of JDK7, also available in Guava).

Consequently, if you want to test whether two strings have the same value you will probably want to use Objects.equals().

// These two have the same value

new String("test").equals("test") // --> true

// ... but they are not the same object

new String("test") == "test" // --> false

// ... neither are these

new String("test") == new String("test") // --> false

// ... but these are because literals are interned by

// the compiler and thus refer to the same object

"test" == "test" // --> true

// ... but you should really just call Objects.equals()

Objects.equals("test", new String("test")) // --> true

Objects.equals(null, "test") // --> false

You almost always want to use Objects.equals(). In the rare situation where you know you're dealing with interned strings, you can use ==.

# Array

*array is a language construct.*

int[] year = new int[] {2, 0, 1, 4};

# Array: One Dimensional

**int[] myIntArray = new int[]{1,2,3};**

**int length = myIntArray.length;**

**System.out.println("myInteArray: " + Arrays.toString(myIntArray));**

// Declaration of an array, initializing with default values

int[] myIntArray1 = new int[3];

// Declaration of array, initializing with given values: method 1.

int[] myIntArray2 = {1,2,3};

// Declaration of array, initializing with given values: method 2.

// Also shows the property of anonymous array-object creation

int[] myIntArray3 = new int[]{1,2,3};

// Shows 3 step declaration and initialization.

**int**[] myIntArray4; // step 1:

myIntArray4 = **new** **int**[3]; // step 2: not declared until here

myIntArray4[0] = 1; // step 3: initialization start

myIntArray4[1] = 2; // step 3: initialization continuing

myIntArray4[2] = 3; // step 3: initialization completed

# Array: Multidimensional

// Declaration

int[][] num = new int[5][2];

// or int num[][] = new int[5][2];

// or int[] num[] = new int[5][2];

// Initialization

num[0][0]=1;

num[0][1]=2;

num[1][0]=1;

num[1][1]=2;

num[2][0]=1;

num[2][1]=2;

num[3][0]=1;

num[3][1]=2;

num[4][0]=1;

num[4][1]=2;

// or

int[][] num2 = { {1,2}, {1,2}, {1,2}, {1,2}, {1,2} };

# Ragged Array (or Non-rectangular Array)

So here we are defining columns explicitly.

int[][] num = new int[5][];

num[0] = new int[1];

num[1] = new int[2];

num[2] = new int[5];

num[3] = new int[2];

num[4] = new int[3];

Another Way:

int[][] num = { {1}, {1,2}, {1,2,3,4,5} ,{1,2}, {1,2,3} };

For accessing

for (int i = 0; i < (num.length); i++ ) {

for (int j = 0; j < num[i].length; j++)

System.out.println(num[i][j]);

}

Alternatively:

for (int[] a : num) {

for (int i : a) {

System.out.println(i);

}

}

# Lists

List<String> string = new ArrayList<String>();

List<Integer> integer = new ArrayList<Integer>();

List<Integer> integer = new ArrayList<Integer>(4, 7, 1, 6, 3, 5, 4);

or

List<Integer> integer = Arrays.asList(4, 7, 1, 6, 3, 5, 4);

# HashTable

Type arguments cannot be of primitive type.

Incorrect:

Hashtable<char, int> ht = new Hashtable<>();

Correct:

Hashtable<Character, Integer> ht = new Hashtable<>();

# Map

map = hash = associated array = dictionary = key-value store

Map<Integer, String> counts = new TreeMap<>();

Iterator<Integer> keys = counts.keySet().iterator();

*// Retrieves is same order of insertions*

Map<Integer, String> counts = new LinkedHashMap<>();

Iterator<Integer> keys = counts.keySet().iterator();

*// Thread safe for simultaneous reads and writes*

Map<Integer, String> counts = new ConcurrentHashMap<>();

Iterator<Integer> keys = counts.keySet().iterator();

# Set

Set <String> set = new HashSet<>();

# Iterator

package practice;

import java.util.\*;

public class IteratorDemo {

public static void main(String args[]) {

// Create an array list

ArrayList<String> al = new ArrayList<String>();

// add elements to the array list

al.add("C");

al.add("A");

al.add("E");

al.add("B");

al.add("D");

al.add("F");

// Use iterator to display contents of al

System.out.print("Original contents of al: ");

Iterator<String> itr = al.iterator();

while(itr.hasNext()) {

Object element = itr.next();

System.out.print(element + " ");

}

System.out.println();

// Modify objects being iterated

ListIterator<String> litr = al.listIterator();

while(litr.hasNext()) {

Object element = litr.next();

litr.set(element + "+");

}

System.out.print("Modified contents of al: ");

itr = al.iterator();

while(itr.hasNext()) {

Object element = itr.next();

System.out.print(element + " ");

}

System.out.println();

// Now, display the list backwards

System.out.print("Modified list backwards: ");

while(litr.hasPrevious()) {

Object element = litr.previous();

System.out.print(element + " ");

}

System.out.println();

}

}

# Java String objects are immutable

<http://stackoverflow.com/questions/1552301/immutability-of-strings-in-java>

String str = new String();

str = "Hello";

System.out.println(str); //Prints Hello

str = "Help!";

System.out.println(str); //Prints Help!

Now, in Java, String objects are immutable. Then how come the object str can be assigned value "Help!". Isn't this contradicting the immutability of strings in Java? Can anybody please explain me the exact concept of immutability?

Edit:

Ok. I am now getting it, but just one follow-up question. What about the following code:

String str = "Mississippi";

System.out.println(str); // prints Mississippi

str = str.replace("i", "!");

System.out.println(str); // prints M!ss!ss!pp!

Does this mean that two objects are created again ("Mississippi" and "M!ss!ss!pp!") and the reference str points to a different object after replace() method?

str is not an object, it's a reference to an object. "Hello" and "Help!" are two distinct String objects. Thus, str points to a string. You can change what it points to, but not that which it points at.

Take this code, for example:

String s1 = "Hello";

String s2 = s1;

// s1 and s2 now point at the same string - "Hello"

Now, there is nothing1 we could do to s1 that would affect the value of s2. They refer to the same object - the string "Hello" - but that object is immutable and thus cannot be altered.

If we do something like this:

s1 = "Help!";

System.out.println(s2); // still prints "Hello"

Here we see the difference between mutating an object, and changing a reference. s2 still points to the same object as we initially set s1 to point to. Setting s1 to "Help!" only changes the reference, while the String object it originally referred to remains unchanged.

If strings were mutable, we could do something like this:

String s1 = "Hello";

String s2 = s1;

s1.setCharAt(1, 'a'); // Fictional method that sets character at a given pos in string

System.out.println(s2); // Prints "Hallo"

''Edit to respond to OP's edit:\*\* If you look at the source code for String.replace(char,char) (also available in src.zip in your JDK installation directory -- a pro tip is to look there whenever you wonder how something really works) you can see that what it does is the following:

If there is one or more occurrences of oldChar in the current string, make a copy of the current string where all occurrences of oldChar are replaced with newChar.

If the oldChar is not present in the current string, return the current string.

So yes, "Mississippi".replace('i', '!') creates a new String object. Again, the following holds:

String s1 = "Mississippi";

String s2 = s1;

s1 = s1.replace('i', '!');

System.out.println(s1); // Prints "M!ss!ss!pp!"

System.out.println(s2); // Prints "Mississippi"

System.out.println(s1 == s2); // Prints "false" as s1 and s2 are two different objects

Your homework for now is to see what the above code does if you change s1 = s1.replace('i', '!'); to s1 = s1.replace('Q', '!'); :)

1Actually, it is possible to mutate strings (and other immutable objects). It requires reflection and is very, very dangerous and should never ever be used unless you're actually interested in destroying the program.

# Why is String.length() a method?

<http://stackoverflow.com/questions/8720220/why-is-string-length-a-method>

Java is a standard, not just an implementation. Different vendors can license and implement Java differently, as long as they adhere to the standard. By making the standard call for a field, that limits the implementation quite severely, for no good reason.

Also a method is much more flexible in terms of the future of a class. It is almost never done, except in some very early Java classes, to expose a final constant as a field that can have a different value with each instance of the class, rather than as a method.

The length() method well [predates](http://www.tns.lcs.mit.edu/manuals/java-1.1.1/api/java.lang.String.html#length%28%29) the CharSequence interface, probably from its first version. Look how well that worked out. Years later, without any loss of backwards compatibility, the CharSequence interface was introduced and fit in nicely. This would not have been possible with a field.

So let's really inverse the question (which is what you should do when you design a class intended to remain unchanged for decades): What does a field gain here, why not simply make it a method?

# Why is array length a property?

<http://stackoverflow.com/questions/9297899/arrays-length-property>

The reason is, that an array is a language construct.

Arrays are special objects in java; they have a simple attribute named length which is final. There is no "class definition" of an array (you can't find it in any .class file), they're a part of the language itself.

# BufferedReader vs Scanner vs Console

BufferedReader input = new BufferedReader(new InputStreamReader(System.in));

Scanner reader = new Scanner(System.in);

Console console = System.console();

BufferedReader

* Since Java 1.1
* Throws checked exceptions
* Can read chars, char arrays, and lines
* Fast
* Buffer size is 8,192.

Scanner

* Since Java 1.5
* Doesn't throw checked exceptions
* Can read lines, whitespace-delimited tokens, regex-delimited tokens, and numbers
* Difficult to [read single characters](http://stackoverflow.com/a/2597939)
* Buffer size is 1,024
* Can pass a BufferedReader to a scanner as the source of characters to parse.

Console

* Since Java 1.6
* Doesn't throw checked exceptions
* Can read lines
* [Underlying reader](http://docs.oracle.com/javase/7/docs/api/java/io/Console.html#reader%28%29) can read chars and char arrays (stops at line bounds)
* Not always available (e.g. [Eclipse](https://bugs.eclipse.org/bugs/show_bug.cgi?id=122429))
* Can read passwords (i.e. read without displaying the characters)

Recommendation: Scanner

* The methods for reading numbers are very useful.
* The exceptions are unchecked, so you do not have to write boilerplate try/catch blocks.
* Use the Scanner if you want to parse the file, use the BufferedReader if you want to read the file line by line.

# JSON

* JSON stands for JavaScript Object Notation.
* The format was specified by Douglas Crockford.
* It was designed for human-readable data interchange.
* It has been extended from the JavaScript scripting language.
* The filename extension is **.json**.
* JSON Internet Media type is **application/json**.
* The Uniform Type Identifier is public.json.

Simple Example in JSON

{

"book": [

{

"id":"01",

"language": "Java",

"edition": "third",

"author": "Herbert Schildt"

},

{

"id":"07",

"language": "C++",

"edition": "second",

"author": "E.Balagurusamy"

}

]

}

# Mapping between JSON and Java entities

<http://json.org/javadoc/org/json/JSONObject.html>

|  |  |
| --- | --- |
| **JSON** | **Java** |
| string | java.lang.String |
| number | java.lang.Number |
| true|false | ava.lang.Boolean |
| null | null |
| array | java.util.List |
| object | java.util.Map |

# How to Add JARs to Project Build Paths in Eclipse (Java)

<http://www.wikihow.com/Add-JARs-to-Project-Build-Paths-in-Eclipse-(Java>)

# Random Number

<http://stackoverflow.com/questions/363681/generating-random-integers-in-a-range-with-java>

**import** java.util.Random;

**class** RandomExample {

**public** **static** **void** main (String[] arguments) {

Random rn = **new** Random();

// Using the Random class

// Produces a 32-bit integer random number

**int** value1 = rn.nextInt();

**int** value2 = rn.nextInt();

**int** value3 = rn.nextInt();

**int** value4 = rn.nextInt();

// Using the Random class

// Produces a double value with a positive sign

// grater than or equal to 0.0 and less than 1.0.

**double** dvalue1 = rn.nextDouble();

**double** dvalue2 = rn.nextDouble();

**double** dvalue3 = rn.nextDouble();

**double** dvalue4 = rn.nextDouble();

// Using the Math class

// Produces a double value with a positive sign

// grater than or equal to 0.0 and less than 1.0.

**double** dmValue1 = Math.*random*();

**double** dmValue2 = Math.*random*();

**double** dmValue3 = Math.*random*();

**double** dmValue4 = Math.*random*();

// Using Math class

// Formula to produce a random number in the range Min to Max inclusive

// Min + (int)(Math.random() \* ((Max - Min) + 1))

// Random integer in the range 5 to 20

**int** Min = 5;

**int** Max = 20;

**int** Answer = Min + (**int**)(Math.*random*() \* ((Max - Min) + 1));

}

}

# Counting Characters

Can use a boolean array to count duplicate characters in ASCII (128) and Unicode (65,536) character sets in Java. (Note: The latest version of Unicode contains a repertoire of more than 128,000 characters.)

boolean[] ascii\_char\_set = new boolean[128];

boolean[] unicode\_char\_set = new boolean[65536]

# Anagram

An **anagram** is direct word switch or word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once. Examples: dog/god, act/cat, pea/ape, tab/bat tar/rat.

<http://www.manythings.org/anagrams/>

# public String[] split(String regex)

Learning's

• Multiple delimiters in a row get convert to empty strings.

Need to have characters between separators, else limit argument of zero applies.

String[] words1 = "anBoxCart".split("[A-Z]"); // words1: {"an", "ox", "art"}

String[] words2 = "anBCart".split("[A-Z]"); // words2: {"an", "", "art" }

String[] words3 = "ABC".split("[A-Z]"); // words3: {}

String[] words4 = "Big dogs.".split(" "); // words4: {"Big", "", "", "dogs."}

# public String[] matches(String regex)

Learning's

• Order of letters in RegEx expression [...] does not matter.

• matches() uses regular expression.

• contains() does not use regular expression.

String str = "meme";

System.out.println(str.matches("[a-z].\*")); // true

System.out.println(str.matches("[a-zA-z].\*")); // true

System.out.println(str.matches("[A-za-z].\*")); // true

System.out.println(str.matches("[a-mn-z].\*")); // true

System.out.println(str.matches("[n-za-m].\*")); // true

System.out.println(str.matches("[a-ln-z].\*")); // false

System.out.println(str.contains("[a-z].\*")); // false

# newline, optional

Avoid the concatenation of newline if the String is empty.

myString += myString.isEmpty() ? "" : "\n";

if(!myString.isEmpty()) myString += "\n";

myString += myString.isEmpty() ? result : result + '\n';

if (stringBuilder.length() == 0) stringBuilder.append('\n');

# Coding examples

<https://gist.github.com/Jocelyn9>

* Recover Binary Search Tree
* 1.3 Given two strings, write a method to decide if one is a permutation of the other.
* 1.1 Implement an algorithm to determine if a string has all unique characters. Whatif you cannot use additional data structures?

<http://algorithms.tutorialhorizon.com/>

* Many different problems

Java Coding Interview Rotate an Array by N elements

* <https://www.youtube.com/watch?v=E2ZSmN7qsmI#t=293.248483>

Cracking The Coding Interview: Q1.7 - Method to Rotate a Matrix by 90 Degrees

* <https://www.youtube.com/watch?v=DnVWLZpg7sM> - video python solution
* <http://stackoverflow.com/questions/25882480/rotating-a-nxn-matrix-in-java> java explanation

# join() with separator

https://codereview.stackexchange.com/questions/93840/add-a-newline-to-a-string-if-not-empty-in-java

This is a common pattern - only apply the separator if both parts are substantial.

public static String join(String a, String j, String b) {

if (a == null) {

// No a - just becomes b.

return b;

}

if (b == null) {

// No b - just becomes a.

return a;

}

if (a.length() == 0) {

// No a - just becomes b.

return b;

}

if (b.length() == 0) {

// No b - just becomes a.

return a;

}

return a + j + b;

}

public void test() {

System.out.println(join(null, ",", "Hello"));

System.out.println(join("Hello", ",", "Hello"));

String s = "";

for (String append : new String[]{"Hello", "a", "", "", null, "", "b"}) {

s = join(s, ",", append);

}

System.out.println(s);

}

# RegEx: Match a word

Learning's

• Pattern matching

static String hackerrankInString(String[] messages) {

Pattern p = Pattern.compile(".\*h.\*a.\*c.\*k.\*e.\*r.\*r.\*a.\*n.\*k.\*");

String answer = "";

for (String message : messages) {

// Check if string contains 'hackerrank' with

// 0 or more other chars spaced around each char

Matcher m = p.matcher(message);

if (m.matches())

answer += answer.isEmpty() ? "YES" : "\nYES";

else

answer += answer.isEmpty() ? "NO" : "\nNO";

}

return (answer);

}

# RegEx: Check for all letters in the alphabet

Pangrams

<https://www.hackerrank.com/challenges/pangrams>

Pangrams are sentences constructed by using every letter of the alphabet at least once.

Learning's

• Pattern matching

static String pangram(String sentence) {

String answer = "";

// A regex that works with String.matches(). This uses one look ahead

// for each letter, each of which asserts that the letter is present.

// The (?i) switch turns on case insensitivity.

Pattern p = Pattern.compile("(?i)(?=.\*a)(?=.\*b)(?=.\*c)(?=.\*d)(?=.\*e)"

+ "(?=.\*f)(?=.\*g)(?=.\*h)(?=.\*i)(?=.\*j)(?=.\*k)(?=.\*l)(?=.\*m)"

+ "(?=.\*n)(?=.\*o)(?=.\*p)(?=.\*q)(?=.\*r)(?=.\*s)(?=.\*t)(?=.\*u)"

+ "(?=.\*v)(?=.\*w)(?=.\*x)(?=.\*y)(?=.\*z).\*");

Matcher m = p.matcher(sentence);

if (m.matches())

answer = "pangram";

else

answer = "not pangram";

return (answer);

}

# Verify substring repeats in larger string

Mars Exploration

<https://www.hackerrank.com/challenges/mars-exploration>

Learning's

• Character arithmetic

1. Can use index and modulo of index

• Modulo

1. modulo result is zero based
2. modulo range is 0...n-1
3. 0 modulo <anything> is 0
4. n modulo n is 0

static int alteredLetters(char[] message) {

String sos = "SOS";

int count = 0;

for (int i = 0; i < message.length; i++) {

if (message[i] != sos.charAt(i % 3)) {

count++;

}

}

return count;

}

# Count camel case words

CamelCase

<https://www.hackerrank.com/challenges/camelcase>

Learning's

• Character arithmetic

• RegEx

• String methods

• Java8 Streams example

public class Solution extends InputData {

static boolean DEBUG = true;

static boolean TEST = true;

static int camelCase(String str) {

int count = 1;

for (int i = 0; i < str.length(); i++) {

char ch = str.charAt(i);

if ('A' <= ch && ch <= 'Z')

count = count + 1;

}

return (count);

}

/\*

>>>

>>> Alternate solution #1: Using regex

>>>

Use a regex matching to split the string on capital letters the resulting

array contains contiguous sections of lowercase letters this works because

the problem states that each word has at least 2 characters, and we know

that the first character of each word is always capitalized.

String[] words = str.split("[A-Z]");

return (words.length);

>>>

>>> Alternate solution #2: Using replace() method

>>>

return (str.length() - str.replaceAll("[A-Z]", "").length() + 1);

>>>

>>> Alternate solution #3: Using Java8 Streams.

>>>

return (str.chars().filter(c -> Character.isUpperCase((char)c)).count()+1);

\*/

# Char as a value versus as an index

Weighted Uniform Strings

- Character's numeric value is used as a value from 1..26

- i.e. weight a = 1, b = 2, ... z = 26

- where curr = 'a' && curr - 'a' + 1 = 1

Anagram

- Character's numeric value is used as an index, from 0..25

- i.e. count[a = 0] = x, count[b = 1] = y, count[c = 3] = z, ...

- where curr = 'a' && curr - 'a' == 0

# Calculate all weighted uniform string values

Weighted Uniform Strings

<https://www.hackerrank.com/challenges/weighted-uniform-string>

Learning's

• How to initialize and use "prev" and "curr".

• How to use character arithmetic:

1. returning character's index starting at 1 (versus 0)

• How to use HaskSet.

// Author: Rodney Shaghoulian

// Github: github.com/rshaghoulian

// HackerRank: hackerrank.com/rshaghoulian

import java.util.Scanner;

import java.util.HashSet;

// Traverse the string just 1 time. Save the weights of every

// possible uniform substring in a HashSet.

// Time complexity: O(n)

// Space complexity: O(n)

public class Solution {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

String str = scan.next();

int n = scan.nextInt();

HashSet<Integer> weights = getWeights(str);

while (n-- > 0) {

int x = scan.nextInt();

System.out.println(weights.contains(x) ? "Yes" : "No");

}

scan.close();

}

private static HashSet<Integer> getWeights(String str) {

HashSet<Integer> weights = new HashSet<>();

int weight = 0;

char prev = ' '; // so it doesn't match 1st character

for (int i = 0; i < str.length(); i++) {

char curr = str.charAt(i);

if (curr != prev) {

weight = 0;

}

weight += curr - 'a' + 1;

weights.add(weight);

prev = curr;

}

return weights;

}

}

# How many characters in one string are not in another string.

Anagram

https://www.hackerrank.com/challenges/anagram/problem

Anagram is a word, phrase, or name formed by rearranging the letters of

another, such as cinema, formed from iceman.

Learning**: (Using int array)**

count[0] count[1] count[2] count[3] count[4] ... count[25]

a b c d e ... z

aabc +2 +1 +1 0 0 ... 0

abbd -1 -2 0 -1 0 ... 0

===================================================================

Total 1 -1 1 -1 0 ... 0

abs(1) + abs(-1) + abs(1) + abs(-1) = 4 / 2 = 2

static String anagram(int size, String[] strs) {

String answer = "";

for (String s : strs) {

// Use an int array [0..25] to represent chars a..z to store char counts.

int[] count = new int[26];

if (s.length() %2 != 0) {

answer += "-1\n";

} else {

// For each char in str1, increment count

for(int i = 0; i < s.length() / 2; i++){

count[s.charAt(i) -'a']++;

}

// For each char in str2, decrement count

for(int i = s.length() / 2; i < s.length(); i++){

count[s.charAt(i) - 'a']--;

}

// The abs sum of array give total number of chars that don't match

int sum = 0;

for(int i = 0; i < 26; i++){

sum += Math.abs(count[i]);

}

// Divide by 2 to calculate number of characters to change in

// one string to match the other string.

answer += sum / 2 + "\n";

}

}

return (answer);

}

# How many characters in one string are not in another string

Anagram

https://www.hackerrank.com/challenges/anagram/problem

Anagram is a word, phrase, or name formed by rearranging the letters of

another, such as cinema, formed from iceman.

Learning: **(Using StringBuilder)**

• **Solution:**

- Split string into 2 equal parts: st1 and st2

- Traverse str2 one character at a time. For each character:

- Check str1 for existence and delete one in str1 if it exists.

- Answer is the length of str1 (after all the deletions).

• When the problem definition is ambiguous, review the discussion tab for

clarification.

• To split a string in two:

String str1 = s.substring(0, len / 2));

String str2 = s.substring(len / 2);

• StringBuilder has methods:

int indexOf(String) and deleteCharAt(int);

• When using indexOf(String) and not found, returns -1.

• Can convert a char into string using concatenation.

int x = str1.indexOf(str2.charAt(i) + "");

static String anagram(int size, String[] strs) {

String answer = "";

for (String s : strs) {

int len = s.length();

if (len % 2 != 0) {

answer += "-1\n";

} else {

StringBuilder str1 = new StringBuilder(s.substring(0, len / 2));

String str2 = s.substring(len / 2);

for (int i = 0; i < str2.length(); i++) {

int x = str1.indexOf(str2.charAt(i) + "");

if (x != -1) str1.deleteCharAt(x);

}

answer += str1.length() + "\n";

}

}

return (answer);

}

# Create anagram by deleting characters

Making Anagrams

https://www.hackerrank.com/challenges/making-anagrams

Learning:

• Use int array to count letters

int[] chars = new int[26];

• Two different for loop styles can be used:

for (int j = 0; j < s1.length(); j++) {

chars[s1.charAt(j) - 'a']++;

for (char c1 : s1.toCharArray())

chars[c1 - 'a']++;

• Solution:

count[0] count[1] count[2] count[3] count[4] ... count[25]

a b c d e ... z

cde 0 0 +1 +1 +1 ... 0

abc -1 -1 -1 0 0 ... 0

===================================================================

Total -1 -1 0 +1 +1 ... 0

abs(-1) + abs(-1) + abs(0) + abs(+1) + abs(+1) = 4

anagram = "a"

count[0] count[1] count[2] count[3] count[4] ... count[25]

a b c d e ... z

abcd +1 +1 +1 +1 0 ... 0

abc -1 -1 -1 0 0 ... 0

===================================================================

Total 0 0 0 1 0 ... 0

abs(0) + abs(0) + abs(0) + abs(1) + abs(0) = 1

anagram = "abc"

count[0] count[1] count[2] count[3] count[4] ... count[25]

a b c d e ... z

ab +1 +1 0 0 0 ... 0

cd 0 0 -1 -1 0 ... 0

===================================================================

Total +1 +1 -1 -1 0 ... 0

abs(+1) + abs(+1) + abs(-1) + abs(-1) + abs(0) = 4

anagram = ""

static String makingAnagrams(String s1, String s2) {

String answer = "";

// Use an int array [0..25] to represent chars a..z to store char counts.

int[] chars = new int[26];

// For each char in s1, increment chars

for (int j = 0; j < s1.length(); j++) {

chars[s1.charAt(j) - 'a']++;

}

// For each char in s2, decrement chars

for (int k = 0; k < s2.length(); k++) {

chars[s2.charAt(k) - 'a']--;

}

// The abs sum of array give total number of chars that don't match

int sum = 0;

for (int l = 0; l < 26; l++) {

sum += Math.abs(chars[l]);

}

answer += sum + "\n";

return (answer);

}

static String makingAnagrams\_2nd(String s1, String s2) {

String answer = "";

// Use an int array [0..25] to represent chars a..z to store char counts.

int[] chars = new int[26];

// For each char in s1, increment count

for (char c1 : s1.toCharArray())

chars[c1 - 'a']++;

// For each char in s2, decrement count

for (char c2 : s2.toCharArray())

chars[c2 - 'a']++;

// The abs count of array give total number of chars that don't match

int count = 0;

for (int i: chars)

count += Math.abs(chars[i]);

answer += count + "\n";

return (answer);

}

# Can an anagram of a string also be a palindrome

Game of Thrones - I

https://www.hackerrank.com/challenges/game-of-thrones

Learning:

• Use int array to count letters

int[] chars = new int[26];

• Use Set to track non-letter pairs

if (set.contains(ch)){

set.remove(ch);

} else {

set.add(ch);

}

// Use an int array

static String gameOfThronesOne(String s) {

// Use an int array [0..25] to represent chars a..z to store char counts.

int[] chars = new int[26];

// Count occurrences of each letter

for (int j = 0; j < s.length(); j++) {

chars[s.charAt(j) - 'a']++;

}

// - if each letter count % 2 == 0, then YES

// - if each letter count % 2 == 0, except 1 is count % 2 == 1, then YES

// - else NO

int count = 0;

for (int i = 0; i < 26; i++)

count += chars[i] % 2;

return count <= 1 ? "YES\n" : "NO\n";

}

// Using a Set

static String gameOfThronesOne\_2nd(String str) {

Set<Character> set = new HashSet<>();

for(Character ch : str.toCharArray()){

if(set.contains(ch)){

set.remove(ch);

}else{

set.add(ch);

}

}

return (set.size() <= 1) ? "YES" : "NO";

}

# See if an anagram of a string is also a palindrome

Two Strings

https://www.hackerrank.com/challenges/two-strings

Learning:

• Using int array to count letters

int[] chars = new int[26];

• Using HashSet<>() to find union

a.retainAll(b);

// Using int array

static String twoStrings\_1st(int size, String[] strs) {

String answer = "";

for (int index = 0; index < size \* 2; index+=2) {

String str1 = strs[index];

String str2 = strs[index + 1];

int[] c1 = new int[26];

int[] c2 = new int[26];

boolean found = false;

// Count characters in str1

for (int i = 0; i < str1.length(); i++)

c1[str1.charAt(i) - 'a']++;

// Count characters in str2

for (int i = 0; i < str2.length(); i++)

c2[str2.charAt(i) - 'a']++;

// Compare character counts between str1 and str2

for (int i = 0; i < 26; i++){

if (c1[i] > 0 && c2[i] > 0) {

found = true;

break;

}

}

answer += found ? "YES\n" : "NO\n";

}

return (answer);

}

// Using HashSet

static String twoStrings(int size, String[] strs) {

Set<Character> a;

Set<Character> b;

String answer = "";

for (int index = 0; index < size \* 2; index += 2) {

String str1 = strs[index];

String str2 = strs[index + 1];

a = new HashSet<Character>();

b = new HashSet<Character>();

for (char c : str1.toCharArray()) {

a.add(c);

}

for (char c : str2.toCharArray()) {

b.add(c);

}

// store the set intersection in set 'a'

a.retainAll(b);

answer += a.isEmpty() ? "NO\n" : "YES\n";

}

return (answer);

}

# Count unique characters

String Construction

https://www.hackerrank.com/challenges/string-construction

Learning:

• HashSet solution

for (char c : str.toCharArray()) { chars.add(c); }

• Java 8 solution

for (String str: strs) { answer += str.chars().distinct().count() + "\n"; }

• Array solution (assumes lower case letters)

for (int i = 0; i < str.length(); i++) { counts[str.charAt(i) - 'a' ] = 1; }

// HashSet solution

static String stringConstruction\_1st(int size, String[] strs) {

String answer = "";

for (String str: strs) {

Set <Character> chars = new HashSet<>();

for (char c : str.toCharArray()) {

chars.add(c);

}

answer += chars.size() + "\n";

}

return (answer);

}

// Java 8 solution

static String stringConstruction\_2nd(int size, String[] strs) {

String answer = "";

for (String str: strs) {

answer += str.chars().distinct().count() + "\n";

}

return (answer);

}

// Array solution (assumes lower case letters)

static String stringConstruction(int size, String[] strs) {

String answer = "";

int total = 0;

for (String str: strs) {

int counts[] = new int[26];

for (int i = 0; i < str.length(); i++) {

counts[str.charAt(i) - 'a' ] = 1;

}

total = 0;

for (int count: counts) {

total += count;

}

answer += total + "\n";

}

return (answer);

}

# Rotated every letter in a string by a fixed number K.

Caesar Cipher

<https://www.hackerrank.com/challenges/caesar-cipher-1>

Learning's

• Use StringBuilder and append() to create new string.

• Use Character.isLetter() to ignore non-letter characters.

• Putting encrypt code in own method is better.

• Good example of character arithmetic:

1. returning character
2. independent if lower case or upper case
3. modulo result is zero based

// Author: Rodney Shaghoulian

// Github: github.com/rshaghoulian

// HackerRank: hackerrank.com/rshaghoulian

import java.util.Scanner;

public class Solution {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int length = scan.nextInt();

String str = scan.next();

int K = scan.nextInt();

scan.close();

StringBuilder sb = new StringBuilder();

for (int i = 0; i < length; i++) {

char ch = str.charAt(i);

sb.append(encrypt(ch, K));

}

System.out.println(sb);

}

/\* Encrypts a character using Caesar Cipher \*/

private static char encrypt(char ch, int K) {

if (!Character.isLetter(ch)) {

return ch;

}

char base = Character.isLowerCase(ch) ? 'a' : 'A';

return (char) ((ch - base + K) % 26 + base);

}

}

# Convert a string into a palindrome

The Love-Letter Mystery

<https://www.hackerrank.com/challenges/the-love-letter-mystery>

Given a string, convert it into a palindrome using the minimum number of operations.

Learning's

• By using Math.abs() , you don't have to check which end is larger before dong subtraction.

• Can use the for loop index to traverse from both ends simultaneously.

// Author: Rodney Shaghoulian

// Github: github.com/rshaghoulian

// HackerRank: hackerrank.com/rshaghoulian

import java.util.Scanner;

// We compare ASCII values for pairs of characters. The 1st pair we consider is the 2

// characters at opposite ends of the String. We then move inwards until we consider

// all palindromic pairs.

public class Solution {

public static void main(String [] args) {

Scanner scan = new Scanner(System.in);

int T = scan.nextInt();

while (T-- > 0) {

String str = scan.next();

System.out.println(minimumOperations(str));

}

scan.close();

}

private static int minimumOperations(String str) {

int count = 0;

for (int i = 0; i < str.length() / 2; i++) {

count += Math.abs(str.charAt(i) - str.charAt(str.length() - 1 - i));

}

return count;

}

}

# Delete duplicate consecutive letters

Alternating Characters

<https://www.hackerrank.com/challenges/alternating-characters/>

Learning's

• When the loop index is initialized to 1 (instead of 0), then the loop index can be used for "prev" and "curr".

// Author: Rodney Shaghoulian

// Github: github.com/rshaghoulian

// HackerRank: hackerrank.com/rshaghoulian

import java.util.Scanner;

// Each time we have 2 consecutive characters that are the same, we can delete 1 of them

public class Solution {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int T = scan.nextInt();

while (T-- > 0) {

String str = scan.next();

System.out.println(minimumDeletions(str));

}

scan.close();

}

private static int minimumDeletions(String str) {

int deletions = 0;

for (int i = 1; i < str.length(); i++) {

if (str.charAt(i-1) == str.charAt(i)) {

deletions++;

}

}

return deletions;

}

}

# Counting frequency using HashMap and HashSet

Gemstones

<https://www.hackerrank.com/challenges/gem-stones>

Learning's

• Querying HashMap with key and then retrieving value.

• Iterating over a HashSet to get information for updating a HashMap.

• Learning methods of HashSet and HashMap

private static int solve(int size, String[] rocks) {

HashMap<Character, Integer> allElements = new HashMap<>();

int count = 0;

for (String rock : rocks) {

Set<Character> elements = new HashSet<>();

for (int i = 0; i < rock.length(); i++) {

elements.add(rock.charAt(i));

}

for (Character element : elements) {

Integer frequency = allElements.get(element);

if (frequency == null) {

allElements.put(element, 1);

} else {

allElements.put(element, frequency + 1);

}

}

}

for (Integer frequency : allElements.values()) {

if (frequency == rocks.length)

count = count + 1;

}

return (count);

}

# Delete any pair of adjacent letters using ArrayDeque

Super Reduced String

<https://www.hackerrank.com/challenges/reduced-string>

Learning's

• Learning methods of ArrayDeque

static String super\_reduced\_string\_2nd(String string) {

// Benjamin's 2nd solution using ArrayDeque

if (string == null || string.length() == 0) {

return ("Empty String");

}

ArrayDeque data = new ArrayDeque();

for (int i = 0; i < string.length(); i++) {

if (!data.isEmpty() && (char) data.peek() == string.charAt(i)) {

data.pop();

} else {

data.push(string.charAt(i));

}

}

if (data.size() == 0) {

return ("Empty String");

} else {

String str = "";

for (Object s : data) {

str = s + str;

}

return (str);

}

}

# Convert str into 2 distinct alternating chars using matrix

Two Characters

<https://www.hackerrank.com/challenges/two-characters>

• TODO; Trace/learn this code (which was different from my solution).

// Author: Rodney Shaghoulian

// Github: github.com/rshaghoulian

// HackerRank: hackerrank.com/rshaghoulian

import java.util.Scanner;

/\*

Time Complexity: O(n) with just 1 traversal of our String

Space Complexity: O(1)

There are approximately 26^2 possible combinations of alternating pairs of letters.

Notice that this number remains constant and is not dependent on the length of our

input String. This fact will help us achieve a linear O(n) runtime.

We want to solve this problem with just 1 traversal of our String, so we solve all

26^2 subproblems simultaneously. We use two int[26][26] arrays to keep track of the

26^2 solutions.

As we iterate through our String, we update our two int[26][26] arrays as follows:

- int[26][26] letter ---> This array is used to keep track of which of the alternating

characters we saw last. To achieve this, for the current character "ch", we update the

corresponding row (26 entries) and column (26 entries) with the (ASCII) value of "ch".

- int[26][26] count ----> if we find that no solution exists for a pair of characters

(which happens when the characters don't alternate), we store -1 in this array.

Otherwise, we store he current maximum length of "s" for the pair of characters.

Our final answer is the maximum value in our "int[26][26] count" array.

\*/

public class Solution {

public static final int NUM\_LETTERS = 26;

public static void main(String [] args) {

/\* Save input \*/

Scanner scan = new Scanner(System.in);

int length = scan.nextInt();

String str = scan.next();

scan.close();

/\* Edge case \*/

if (length <= 1) {

System.out.println(0);

return;

}

/\* Create arrays representing the 26^2 subproblems \*/

int [][] pair = new int[NUM\_LETTERS][NUM\_LETTERS];

int [][] count = new int[NUM\_LETTERS][NUM\_LETTERS];

for (int i = 0; i < length; i++) {

char letter = str.charAt(i);

int letterNum = letter - 'a';

/\* Update row \*/

for (int col = 0; col < NUM\_LETTERS; col++) {

if (pair[letterNum][col] == letter) {

count[letterNum][col] = -1;

}

if (count[letterNum][col] != -1) {

pair[letterNum][col] = letter;

count[letterNum][col]++;

}

}

/\* Update column \*/

for (int row = 0; row < NUM\_LETTERS; row++) {

if (pair[row][letterNum] == letter) {

count[row][letterNum] = -1;

}

if (count[row][letterNum] != -1) {

pair[row][letterNum] = letter;

count[row][letterNum]++;

}

}

}

/\* Find max in "count" array \*/

int max = 0;

for (int row = 0; row < NUM\_LETTERS; row++) {

for (int col = 0; col < NUM\_LETTERS; col++) {

max = Math.max(max, count[row][col]);

}

}

System.out.println(max);

}}

# 2 or more numbers incremented by 1

Sequence of two or more positive integers with each incremented by one.

Separate the Numbers

<https://www.hackerrank.com/challenges/separate-the-numbers>

Learning's

• Need to use Long for very long numbers.

• Use index and substring(0, index) to get incrementally larger string.

• Based upon 1st computed value, construct the remaining string.

static String getBeautiful(int size, String[] strs) {

// Another Java solution, building up a test sequence string for each

// possible starting number and then comparing to the original (from

// the Discussions page).

String answer = "";

for (String str : strs) {

boolean valid = false;

long firstX = -1;

// Try each possible starting number

for (int i = 1; i <= str.length()/2; ++i) {

long x = Long.parseLong(str.substring(0,i));

firstX = x;

// Build up sequence starting with this number

String test = Long.toString(x);

while (test.length() < str.length()) {

test += Long.toString(++x);

}

// Compare to original

if (test.equals(str)) {

valid = true;

break;

}

}

if (valid)

answer += answer.isEmpty() ? "YES " + firstX : "\nYES " + firstX;

else

answer += answer.isEmpty() ? "NO " : "\nNO ";

}

return (answer);

}

# Using loop index for start, start + 1, end - 1, and end.

Funny String

<https://www.hackerrank.com/challenges/funny-string>

String is funny when abs((start + 1) - start) equals abs((end - 1) - end).

Learning:

• When test result has multiple states, change test cases to display results

in different order, i.e "Funny,Not Funny" and "Not Funny,Funny". This

catches when a variable is initialized correctly the 1st time, but is

incorrect in subsequent iterations (i.e. boolean isFunny = true;).

• When need to simultaneously look at string's start and start + 1, begin

loop counter at 1 (instead of 0).

• Can use loop index to simultaneously move from start of string and

end of string, as in start, start + 1, end - 1, and end.

static String solve(int size, String[] strs) {

String answer = "";

for (String str : strs) {

boolean isFunny = true;

int len = str.length();

for (int i = 1; i < len; i++) {

if (Math.abs(str.charAt(i) - str.charAt(i - 1)) ==

Math.abs(str.charAt(len - i - 1) - str.charAt(len - i))) {

// no opp

} else {

isFunny = false;

break;

}

}

answer += isFunny ? "Funny\n" : "Not Funny\n";

}

return (answer);

}

# Count number of occurrences of substring in string

Beautiful Binary String

<https://www.hackerrank.com/challenges/beautiful-binary-string>

A string is a Beautiful Binary String when it does not contain "010".

Learning:

• Using Pattern Matching to find substring requires ".\*" at start and end of

expression, i.e. Pattern p = Pattern.compile(".\*010.\*");.

• str.indexOf(str) returns -1 when string not found.

• str.indexOf(str) returns index of the 1st char.

• The following are equivalent:

str.contains("010")

str.indexOf("010") != -1

• Can also use replaceAll() to compute number of occurrences, i.e

(str.length() - str.replaceAll("010", "").length()) / 3

\*/

static int makeBeautifulBinaryString\_1st(int size, String str) {

String badStr = "010";

int count = 0;

while (str.contains("010")) {

int loc = str.indexOf(badStr);

str = str.substring(0, loc + 2) + "1" + str.substring(loc + 3);

count++;

}

return (count);

}

static int makeBeautifulBinaryString(int size, String str) {

return ((str.length() - str.replaceAll("010", "").length()) / 3);

}

# Find the "one" character to delete to create Palindrome.

Palindrome Index

<https://www.hackerrank.com/challenges/palindrome-index/problem>

Given a string S of lowercase letters, determine the index of the character

whose removal will make a palindrome. There is always a solution.

Learning:

• Traverse from both ends (start, end) to middle. When a char doesn't match,

check next two characters beyond start. If they pass, then delete start

char, else delete end character (because solution is always present). Two

character check is required because must handle even and odd palindrome

character size.

• Solution intended working string "in place." Using additional

StringBugger() resulted in some test cases timing out on website, even

though passing locally (although, after long time).

static int palindromeIndex(String str) {

int len = str.length();

int start = 0, end = len - 1;

while (start < end) {

System.out.println(

"start " +

"start + 1 " +

"start + 2 " +

"end - 2 " +

"end - 1 " +

"end "

);

System.out.println(String.format("%-9c %-9c %-9c %-9c %-9c %-9c ",

(char) str.charAt(start),

(char) str.charAt(start+1),

(char) str.charAt(start+2),

(char) str.charAt(end-2),

(char) str.charAt(end-1),

(char) str.charAt(end))

);

if (str.charAt(start) != str.charAt(end)) {

if ((str.charAt(start + 1) == str.charAt(end)) &&

(str.charAt(start + 2) == str.charAt(end - 1)))

return start;

else

return end;

}

start++;

end--;

}

return -1;

}

iSize: 3

iStrings: [aaab, baa, aaa]

start start + 1 start + 2 end - 2 end - 1 end

a a a a a b

start start + 1 start + 2 end - 2 end - 1 end

b a a b a a

start start + 1 start + 2 end - 2 end - 1 end

a a a a a a

3

0

-1

iSize: 1

iStrings: [abczddcba]

start start + 1 start + 2 end - 2 end - 1 end

a b c c b a

start start + 1 start + 2 end - 2 end - 1 end

b c z d c b

start start + 1 start + 2 end - 2 end - 1 end

c z d d d c

start start + 1 start + 2 end - 2 end - 1 end

z d d z d d

3

iSize: 1

iStrings: [abcddzcba]

start start + 1 start + 2 end - 2 end - 1 end

a b c c b a

start start + 1 start + 2 end - 2 end - 1 end

b c d z c b

start start + 1 start + 2 end - 2 end - 1 end

c d d d z c

start start + 1 start + 2 end - 2 end - 1 end

d d z d d z

5

# Fibonacci NonRecursive

static int fibonacci(int number) {

// Fibonacci Series in Java Using Non-recursion

// http://www.java67.com/2016/05/fibonacci-series-in-java-using-recursion.html

if (number <= 0)

return -1;

if (number == 1 || number == 2)

return 1;

int fib1 = 1;

int fib2 = 1;

int fibonacci = 1;

for (int i = 3; i <= number; i++) {

fibonacci = fib1 + fib2;

fib1 = fib2;

fib2 = fibonacci;

}

return fibonacci;

}

# Fibonacci Recursive

static int fibonacci(int number) {

// THE FIBONACCI SEQUENCE, SPIRALS AND THE GOLDEN MEAN - recursion.

// https://math.temple.edu/~reich/Fib/fibo.html

// The first two numbers of Fibonacci series is always 1, 1.

if (number <= 0)

return (-1);

if (number == 1 || number == 2){

return (1);

}

return (fibonacci(number - 1) + fibonacci(number - 2));

}

# Fibonacci Print Sequence

public class Solution {

public static String fibonacci(int number) {

String seq = "";

if (number <= 0) {

throw new IllegalArgumentException("Expected number to be 1 or greater."

+ "Number is " + number + ".");

} else if (number == 1) {

seq = "0";

} else if (number == 2) {

seq = "0 1";

} else { // number > 2

seq = "0 1 ";

int fib1 = 0;

int fib2 = 1;

int fib3 = 1;

for (int i = 3; i <= number; i++) {

fib3 = fib1 + fib2;

fib1 = fib2;

fib2 = fib3;

seq += fib3 + " ";

}

}

return seq;

}

public static void main(String[] args) {

System.out.println(fibonacci(13));

}

}

// 0 1 1 2 3 5 8 13 21 34 55 89 144

# Fibonacci As List

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

// Return value is a List.

public class Solution {

/\*

The Fibonacci sequence is a list of numbers, where the next value in the

sequence is the sum of the previous two. The sequence defines that the

first number is zero, and the next is one.

\*/

public static List<Integer> fibonacci(int number) {

if (number <= 0) {

throw new IllegalArgumentException("Expected parameter to be 1 or greater. "

+ "Actual parameter is " + number + ".");

} else if (number == 1) {

return Arrays.asList(0);

} else if (number == 2) {

return Arrays.asList(0, 1);

} else { // if (number > 2)

List<Integer> seq = new ArrayList<>();

seq.add(0);

number = number - 1;

seq.add(1);

number = number - 1;

while (number > 0) {

int a = seq.get(seq.size() - 1);

int b = seq.get(seq.size() - 2);

seq.add(a + b);

number = number - 1;

}

return seq;

}

}

public static void main(String[] args) {

List<Integer> fibNumbers = new ArrayList<>();

fibNumbers = fibonacci(13);

for (Integer n : fibNumbers) {

System.out.print(n + " ");

}

}

}

// 0 1 1 2 3 5 8 13 21 34 55 89 144

# Fibonacci As Recursive

public class Solution {

// Fibonacci recursive

private static int fibonacci(int number) {

if (number < 1) {

throw new IllegalArgumentException("Expected number to be 1 or greater. "

+ "Actual number is " + number + ".");

} else if (number == 1) {

return 0;

} else if (number == 2) {

return 1;

} else {

return fibonacci(number - 1) + fibonacci(number - 2);

}

}

public static void main(String[] arga) {

for (int i = 1; i <= 13; i++) {

System.out.print(fibonacci(i) + " " );

}

}

}

// 0 1 1 2 3 5 8 13 21 34 55 89 144

# FizzBuzz as String

public class Solution {

private static String fizzbuzz(int num) {

String answer = "";

if (num % 3 == 0) {

answer = "Fizz";

}

if (num % 5 == 0 ) {

answer += answer.isEmpty() ? "Buzz" : " Buzz";

}

return answer.isEmpty() ? num + "" : answer;

}

public static void main(String[] args) {

for (int i = 1; i < 101; i++)

System.out.println( i + " " + fizzbuzz(i));

}

}

# FizzBuzz as List

import java.util.ArrayList;

import java.util.List;

public class P04\_FizzBuzz\_As\_List {

public static List<String> fizzBuzz (int number) {

List<String> seq = new ArrayList<>(number);

for (int i = 1; i <= number; i++) {

if (i % 15 == 0 ) {

seq.add("FizzBuzz");

} else if (i % 3 == 0) {

seq.add("Fizz");

} else if (i % 5 == 0) {

seq.add("Buzz");

} else {

seq.add(Integer.toString(i));

}

//System.out.println(i + " " + seq.get(seq.size() - 1));

}

return seq;

}

public static void main(String[] args) {

List<String> seq = fizzBuzz(100);

for (String fb : seq) {

System.out.println(fb);

}

}

}

# BitSet Example #1

Pangrams

<https://www.hackerrank.com/challenges/pangrams>

<solution #4>

# BitSet: Example #2

Pangrams

<https://www.hackerrank.com/challenges/pangrams>

<solution #5>

# BitSet: Example #3

Gemstones

https://www.hackerrank.com/challenges/gem-stones

https://github.com/rshaghoulian/HackerRank\_solutions/blob/master/Algorithms/Strings/Gemstones/Solution.java

# BitSet: Example #4

Adler32

/Users/ben/documenets/ws2/java2/jpassion/course/chapter-10-01/javase\_iostream/samples\_netbeans/CustomDataInputOutput/src/Adler32.java