格式转换

**1. 将List<String> 转换为String[]:**

List<String> res = new LinkedList<>();

res.toArray(new String[res.size()])

用的是：public <T> T[] toArray(T[] a)

Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array. If the list fits in the specified array, it is returned therein. Otherwise, a new array is allocated with the runtime type of the specified array and the size of this list.

**2. Integer[] 与 int[] 与List<Integer> 之间的转换：**

**Naive Transformation:**

int[] to integer[]:

Integer[] newArray = new Integer[ids.length];

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

newArray[i] = Integer.valueOf(ids[i]);

}

return newArray;

Integer[] to int[]:

int[] newArray = new int[WrapperArray.length];

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

newArray[i] = WrapperArray[i].intValue();

}

return newArray;

List<Integer> to Integer[]:

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

Integer[] list2 = list.toArray(new Integer[list.size()]);

List<Integer> to int[]: 只能逐个取出放入

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

Integer[] list2 = list.toArray(new Integer[list.size()]);

**Using Java8:**

*/\*\**

*\* int[] vs Integer[]*

*\*/*

public Integer[] intToIntegerArray(int[] nums) {

*// IntStream.of( data ).boxed().toArray( Integer[]::new );*

return Arrays.stream(nums).boxed().toArray(Integer[]::new);

}

public int[] integerToIntArray(Integer[] nums) {

return Arrays.stream(nums).mapToInt(Integer::intValue).toArray();

}

*/\*\**

*\* int[] vs List<Integer>*

*\*/*

public List<Integer> intToIntegerList(int[] nums) {

*// IntStream.of( data ).boxed().collect( Collectors.toList() );*

return Arrays.stream(nums).boxed().collect(Collectors.toList());

}

public int[] integerListToInt(List<Integer> list) {

*// return list.stream().mapToInt(Integer::intValue).toArray();*

return list.stream().mapToInt(i->i).toArray();

}

map 有一个method，叫做putIfAbsent, 可以多用，比先查containsKey, 再操作要更concise

假设输入是一个collection, 比如是list，需要检查corner case，最好直接用

list == null || list.isEmpty(), 而不是用size。

计算某个int换算成二进制的1的位数：

Integer.bitCount(i)

public int findLongestChain(int[][] pairs) {

if (pairs == null || pairs.length == 0) return 0;

*// 1. pair is int[], sort based on pair[0]*

Arrays.sort(pairs, (a, b)->(a[0] - b[0]));

*// ...*

}

design类题目，在内部新建private class, 并且可能需要排序：

implement Comparable<Object> 并且重写compareTo函数：以下是递减：

private class Tweet implements Comparable<Tweet>{

int timestamp;

int tweetId;

Tweet prev;

public Tweet(int timestamp, int tweetId, Tweet prev){

this.timestamp = timestamp;

this.tweetId = tweetId;

this.prev = prev;

}

public int compareTo(Tweet a){ *// in decrease order*

return a.timestamp - this.timestamp;

}

}

**利用BufferedReader从文件中/STDIN读取数据：**

Read data/input from file, given filepath:

BufferedReader input = new BufferedReader(new FileReader(filepath));

while (input.ready()) {

String line = input.readLine();

*//....*

}

input.close();

**需要从System.in一行行读取数据时：**

在这里需要判断每行是否是数字，如果不是，则判为invalid，返回。

try (Scanner scanner = new Scanner(System.in)) {

String cur = null;

while (scanner.hasNextLine() && (cur = scanner.nextLine()) != null) {

if (cur.length() == 0) break; // 用于判断是不是已经读取完

int i = 0;

try {

i = Integer.parseInt(cur);

} catch (NumberFormatException nfe) {

return null;

}

res.add(i);

}

} catch (Exception e) {

}

Map<String, List<String>> map = new HashMap<>();

可以直接把values部分转换成List<List<String>>输出，

return new ArrayList<List<String>>(map.values()); 即可

一般情况下，存字符出现的次数可以直接用

int hash[] = new int[256];

hash[‘c’]++; // 这样直接操作

String s = “”;

s.isEmpty() is true only if s.length() == 0

善用s.startsWith(“abc”)

还可以设置offset，startsWith(“another String”, offset)

相当于 this.substring(offset).startsWith(“another String”)

**新建class，并且需要compare：**

class Tuple implements Comparable<Tuple> {

int x, y, val;

public Tuple (int x, int y, int val) {

this.x = x;

this.y = y;

this.val = val;

}

@Override

public int compareTo (Tuple that) {

return this.val - that.val;

}

}

**位运算相关：**

Integer.bitCount(n) 可以计算出n转换为二进制后一共有多少位为1.

Character有一个method，用来判断某个char是不是英文字符or数字：

Character.isLetterOrDigit