Table of Contents

Monsoon Umbrellas

给定一个 int 和一个 list[int], 让你用 list 里的数来组合出所求的 int, 最少需要多少个 (Leetcode 322)

```
public class Solution {
    public int coinChange(int[] coins, int amount) {
        Arrays.sort(coins);
        int[] dp = new int[amount + 1];
        Arrays.fill(dp, -1);
        dp[0] = 0;
        for(int i = 1; i <= amount; ++i) {
            int min = Integer.MAX_VALUE;
            for (int coin:coins) {
                if (coin>i) break;
                if (dp[i-coin]<0) continue;
                min = Math.min(min, dp[i - coin] + 1);
            }
            if (min==Integer.MAX_VALUE)
                dp[i] = -1;
            else
                dp[i] = min;
        }
        if (dp[amount]==-1 && amount==Integer.MAX_VALUE && coins[0]==1)
            return Integer.MAX_VALUE;
        return dp[amount];
    }
}
```

subarray sum

求 sum(sum(subarr) for subarr of arr) continuous subarray

Solution:
sum((i+1)*(len-i)*A_i for i in range(len))

leetcode 20 Valid Parentheses

```
public class Solution {
    public boolean isValid(String s) {
        Stack stack = new Stack();
        char[] cs = s.toCharArray();
        HashSet<Character> hashSet = new HashSet();
        hashSet.addAll(Arrays.asList('(','{','['));
        for (char c : cs) {
            if (hashSet.contains(c)) {
                stack.push(c);
            }
            else {
                if (stack.empty()) return false;
                char tmp = (char)stack.pop();
                if (tmp=='(' && c!=')')  return false;
                if (tmp=='{' && c!='}')  return false;
                if (tmp=='[' && c!=']')  return false;
            }
        }
        if (stack.empty())
            return true;
        else
            return false;
    }
}
```

Maximum difference in an array

Given an array arr[] of integers, find out the maximum difference between any two elements such that larger element appears after the smaller number.

```
Input : arr = {2, 3, 10, 6, 4, 8, 1}

Output : 8

Explanation : The maximum difference is between 10 and 2.


Input : arr = {7, 9, 5, 6, 3, 2}

Output : 2

Explanation : The maximum difference is between 9 and 7.
```

```
class MaximumDiffrence
{
    /* The function assumes that there are at least two
       elements in array.
       The function returns a negative value if the array is
       sorted in decreasing order.
       Returns 0 if elements are equal  */
    int maxDiff(int arr[], int arr_size)
    {
        int max_diff = arr[1] - arr[0];
        int min_element = arr[0];
        int i;
        for (i = 1; i < arr_size; i++)
        {
            if (arr[i] - min_element > max_diff)
                max_diff = arr[i] - min_element;
            if (arr[i] < min_element)
                min_element = arr[i];
        }
        return max_diff < 0 ? -1 : max_diff;
    }

    /* Driver program to test above functions */
    public static void main(String[] args)
    {
        MaximumDiffrence maxdif = new MaximumDiffrence();
        int arr[] = {1, 2, 90, 10, 110};
        int size = arr.length;
        System.out.println("MaximumDifference is " +
                                maxdif.maxDiff(arr, size));
    }
}
```

Can you sort

sort array by frequency
[4,5,6,5,4,3] -> [3,6,4,4,5,5]

```
public class Main {
    public static void main(String[] args) {
        int n = 10;
        int[] input = new int[]{8,5,5,5,5,1,1,3,3,4};

        HashMap<Integer, Integer> map = new HashMap<>();
        Integer[] nums = new Integer[n];
        for (int i = 0; i < n; i++) {
            nums[i] = new Integer(input[i]);
            map.put(input[i], map.getOrDefault(input[i], 0) + 1);
        }
        Arrays.sort(nums, new Comparator<Integer>(){
            @Override
            public int compare(Integer o1, Integer o2) {
                Integer c1 = map.get(o1);
                Integer c2 = map.get(o2);
                if (c1 != c2) {
                    return Integer.compare(c1, c2);
                }
                else {
                    return Integer.compare(o1, o2);
                }
            }
        });
        for (Integer num : nums) {
            System.out.println(num);
        }
    }
}
```

movie ratings - house robber 互补版本

array 里面取数字，不能连续跳过两个及以上，求最大 sum。

dp(i) = max(dp(i - 1), dp(i - 2)) + nums(i)，结果是 max(dp(n - 1), dp(n - 2))

cut sticks

给一堆火柴 各有长度　每次选取最短的长度 len　然后所有火柴都剪去 len　直到没有为止　记录每次剩余的火柴数量

先排序，后看有多少个值和前一个不一样

Zombies cluster

和 lc 的 friend circle 类似 dfs 直接解决
题目很长，大概意思就是僵尸会传染，且具有传递性，如果一个僵尸传染了另一个僵尸（第二个），第二个又传递了第三个，则为一个 cluster。求 cluster 数量。

```
class Solution {
    int find(int parent[], int i) {
        if (parent[i] == -1)
            return i;
        return find(parent, parent[i]);
    }

    void union(int parent[], int x, int y) {
        int xset = find(parent, x);
        int yset = find(parent, y);
        if (xset != yset)
            parent[xset] = yset;
    }
    public int findCircleNum(int[][] M) {
        int[] parent = new int[M.length];
        Arrays.fill(parent, -1);
        for (int i = 0; i < M.length; i++) {
            for (int j = 0; j < M.length; j++) {
                if (M[i][j] == 1 && i != j) {
                    union(parent, i, j);
                }
            }
        }
        int count = 0;
        for (int i = 0; i < parent.length; i++) {
            if (parent[i] == -1)
                count++;
        }
        return count;
    }
}
```

shift string

We define the following operations on a string:

- Left Shift: abcde -> bcdea
- Right Shift: abcde -> eabcd

Complete the function getShiftedString. The function must return the string s after performing several left shifts and right shifts.

different parity permutation

输入 n，返回 1-n 的所有 permutation，但是要求是相邻两个数奇偶性不同。 并且 permutation 按照字典序大小输出

n = 3

符合条件的有[1,2,3],[3,2,1]

n = 4

符合条件的有 8 个([1,3,2,4]是错的因为奇数和奇数相邻偶数偶数相邻)

```
public class Main {
    private static void dfs(int n, List<List<Integer>> lists, List<Integer> cur) {
        if (cur.size() == n) {
            List<Integer> list = new ArrayList<>(cur);
            lists.add(list);
        }

        for (int i = 1; i <= n; i++) {
            if (cur.contains(i)) {
                continue;
            }
            if (cur.size() > 0) {
                if ((cur.get(cur.size() - 1) % 2) + (i % 2) != 1) {
                    continue;
                }
            }
            cur.add(i);
            dfs(n, lists, cur);
            cur.remove(cur.size() - 1);
        }
    }

    public static void main(String[] args) {
        int n = 4;

        List<List<Integer>> lists = new ArrayList<>();
        if (n == 0) {
            System.out.println("null");
        }
        dfs(n, lists, new ArrayList<Integer>());
        for (List<Integer> innerList : lists) {
            System.out.println(innerList);
        }
    }
}
```

Distinct pairs

2sum 变形
Determine the distinct pairs of elements in an array that sum to the target value. Two pairs (a,b) and (c,d) are considered distinct if and only if the values in sorted order do not match.

Solution: HashSet 、对 target / 2 特别处理

simple queries

两个 array，求第一个 array 里多少个元素小于或等于 第二个 array 的每一个元素。
arr1[1,4,2,2,6] arr2 [3,5] 结果就是 [3,4], 暴力解法会超时，可以用 binary search

Minimum unique array sum

Given an array, you must increment any duplicated elements until all its elements are unique. Return the minimum possible sum of new array

Solution:
先排序，然后 iterate array，如果当前值小于等于前一个值，则改变当前值为前一个值加一，同时记录 sum

Sort by bitCount

给一个数组，按二进制中 1 的个数进行排序， 如果个数相同再比大小

```
public class Main {
    public static void main(String[] args) {
        Integer[] arr = new Integer[]{1,2,3,4,5,6,7};
        Arrays.sort(arr, new Comparator<Integer>() {
            @Override
            public int compare(Integer o1, Integer o2) {
                int c1 = Integer.bitCount(o1);
                int c2 = Integer.bitCount(o2);
                if (c1 != c2) {
                    return Integer.compare(c1, c2);
                }
                else {
                    return Integer.compare(o1, o2);
                }
            }
        });
        System.out.println(Arrays.toString(arr));
    }
}
```

Last substring

给一个 String, 返回它所有 substring 中字典序最大的 substring.

找到以最大的 character 开始的 substring，比较他们。

```
int result = str1.compareTo(str2)
```

给定一个数组 counts, 每个 index 代表一个 ID，每个 element 代表 ID 所在的组的大小，组里只能有 element 这么多个 ID，把 element 一样的 ID 归到一个或多个组里，按从小到大顺序输出所有组

举例说明：

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 3 | 3 | 3 | 3 | 3 | 1 | 3 |

上面一排是 index 也就是 ID
下面一排是 ID 所在的组的大小

输出结果就是

```
0 1 2
3 4 6
5
```

```
public class Main {
    public static void main(String[] args) {
        int[] counts = new int[]{3,3,3,3,3,1,3};

        HashMap<Integer, Integer> map = new HashMap<>();
        List<List<Integer>> result = new ArrayList<>();
        for (int i = 0; i < counts.length; i++) {
            int count = counts[i];
            if (!map.containsKey(count)) {
                List<Integer> tmp = new ArrayList();
                tmp.add(i);
                result.add(tmp);
                map.put(count, result.size() - 1);
            }
            else {
                List<Integer> tmp = result.get(map.get(count));
                tmp.add(i);
                if (tmp.size() == count) {
                    map.remove(count);
                }
            }
        }
        for (List<Integer> list : result) {
            System.out.println(list);
        }
    }
}
```

Usernames System

给一个 list of string, 输出一个 list of string, 要求是在重复的字符串后加上数字消歧义，比如 A 出现第二次的时候就要变成 A1,第三次 A2...