Java方向编程题答案

Day37

> [编程题]木棒拼图
>
> 链接：https://www.nowcoder.com/questionTerminal/8bbc9415216d47459c425b5e19164365?orderByHotValue=1&mutiTagIds=665&page=1&onlyReference=false
>
> **【题目解析】：** 一个木棒集合，每根木棒知道长度，问能否用这些木棒构成一个面积大于0的简单多边形（不能自交）。数据有n次操作，每次操作要么增加一根长度为x的木棒，要么去掉一根长度为x的木棒，每次操作完后问剩下的木棒能否满足上述条件。
>
> **【解题思路】：** 把这个多边形看成一个三角形；把集合里长度最大的拿出来，再把剩下的所有边相加，只要这些边的长度之和大于最长的那边，就可以组成一个三角形；多边形是由三边或三边以上的边构成的。
>
> 要使这些木棒能构成一个面积大于0的简单多边形，只需满足最长的木棒短于剩下的所有木棒的总长。即多边形近乎为一条直线。 因此只需构造一个集合，从小到大存放木棒，每次操作更新总长度，判断 maxLen < sum - maxLen 即可。
>
> **【示例代码】：**

```
import java.util.ArrayList;
import java.util.Scanner;

public class Main {

public static void main(String[] args) {
  // TODO Auto-generated method stub
  Scanner scanner = new Scanner(System.in);
  while (scanner.hasNext()) {
      int n = scanner.nextInt();
      int[][] op = new int[n][2];
      for (int i = 0; i < n; i++) {
          op[i][0] = scanner.nextInt();
          op[i][1] = scanner.nextInt();
      }
      stickPuzzle(n, op);
  }
}

public static void stickPuzzle(int n, int[][] op) {
  ArrayList<Integer> list = new ArrayList<Integer>();
  for (int i = 0; i < n; i++) {
      if (op[i][0] == 1) {
          list.add(op[i][1]);
      } else {
          //不等于1移除
          list.remove(new Integer(op[i][1]));
      }
```

```
        if (canFormPoly(list)) {
            System.out.println("Yes");
        } else {
            System.out.println("No");
        }
    }
}
//判断能否构成多边形
//list: 各边长
public static boolean canFormPoly(ArrayList<Integer> list) {
    int len = list.size();
    for (int i = 0; i < len; i++) {
        int temp = list.remove(i);
        int sum = 0;
        for (int j = 0; j < len - 1; j++) {
            sum += list.get(j);
        }
        if (sum <= temp) { //判断是否任意len-1条边之和大于剩余一条边
            list.add(i, temp);
            return false;
        }
        list.add(i, temp);
    }
    return true;
}
}
```

[编程题]地下迷宫

链接：https://www.nowcoder.com/questionTerminal/571cfbe764824f03b5c0bfd2eb0a8ddf?toCommentId=388087

【题目解析】：无（见解题思路）

【解题思路】：小青蛙有一天不小心落入了一个地下迷宫,小青蛙希望用自己仅剩的体力值P跳出这个地下迷宫。为了让问题简单,假设这是一个n*m的格子迷宫,迷宫每个位置为0或者1,0代表这个位置有障碍物, 小青蛙达到不了这个位置;1代表小青蛙可以达到的位置。 小青蛙初始在(0,0)位置,地下迷宫的出口在(0,m-1)(保证这两个位置都是1,并且保证一定有起点到终点可达的路径),小青蛙在迷宫中水平移动一个单位距离需要消耗1点体力值,向上爬一个单位距离需要消耗3个单位的体力值,向下移动不消耗体力值,当小青蛙的体力值等于0的时候还没有到达出口,小青蛙将无法逃离迷宫。现在需要你帮助小青蛙计算出能否用仅剩的体力值跳出迷宫(即达到(0,m-1)位置)。

【示例代码】：

```
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Scanner;

public class Main  {

static int n, m, maxRemainEnergy = 0;
static int[][] map;
static boolean flag = false;
```

```
static String path = "";
static LinkedList<String> linkedlist = new LinkedList<>();

    public static void main(String[] args) {

      //输入
      Scanner sc = new Scanner(System.in);
      n = sc.nextInt();
      m = sc.nextInt();
      int P = sc.nextInt();
      map = new int[n][m];
      for (int i = 0; i < n; i++) {
          for (int j = 0; j < m; j++) {
              map[i][j] = sc.nextInt();
          }
      }

      //处理
      runMap(0, 0, P);

      //输出
      if (!flag)
          System.out.println("Can not escape!");
      else
          System.out.println(path);

    }

    public static void runMap(int x, int y, int energy) {
          if (energy < 0 || x<0 || y<0 || x>=n || y>= m || map[x][y] != 1) return;
          else {
              linkedlist.offer("[" + x + "," + y + "]");
              map[x][y] = 0;
              if (x == 0 && y == m - 1) {
                  if (energy >= maxRemainEnergy) {
                      maxRemainEnergy = energy;
                      updatePath();
                  }
                  map[x][y] = 1; linkedlist.removeLast();
                  flag = true; return;
              }
              runMap(x, y+1, energy-1);
              runMap(x+1, y, energy);
              runMap(x-1, y, energy-3);
              runMap(x, y-1, energy-1);
              map[x][y] = 1;linkedlist.removeLast();
          }
    }

    public static void updatePath() {
      StringBuilder sb = new StringBuilder();
      Iterator<String> iterator = linkedlist.iterator();

      while (iterator.hasNext())
```

```
            sb.append(iterator.next() + ",");
        if (sb.length() > 0)
            sb.deleteCharAt(sb.length() - 1);
        path = sb.toString();
    }

}
```