Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Кафедра «Математической кибернетики и информационных технологий»

Курсовая работа по дисциплине

«Структуры и алгоритмы обработки данных»

Выполнил:

студент группы БВТ1902

Клюев А.П.

Введение

Цель данной работы – решить задачи использовав эффективные

алгоритмы.

Задачи:

• Задача 1. «Треугольник с максимальным периметром»

Массив A состоит из целых положительных чисел длин отрезков.

Составьте из трех отрезков такой треугольник, чтобы его периметр был

максимально возможным. Если невозможно составить треугольник с

положительной площадью функция возвращает 0.

• Задача 2. «Максимальное число»

Дан массив неотрицательных целых чисел nums. Расположите их в таком

порядке, чтобы вместе они образовали максимально возможное число.

Замечание: Результат может быть очень большим числом, поэтому

представьте его как string, а не integer.

• Задача 3. «Сортировка диагоналей в матрице»

Дана матрица mat размером m \* n, значения целочисленные.

Напишите функцию, сортирующую каждую диагональ матрицы по

возрастанию и возвращающую получившуюся матрицу.

• Задача 4. «Шарики и стрелы»

Некоторые сферические шарики распределены по двухмерному пространству. Для каждого шарика даны x координаты начала и конца его горизонтального диаметра. Так как пространство двумерно, то y координаты не имеют значения в данной задаче. Координата x start всегда меньше x end. Стрелу можно выстрелить строго вертикально (вдоль yоси) из разных точек x оси. Шарик с координатами x start и x end уничтожается стрелой, если она была выпущена из такой позиции x, что xstart ⩽ x ⩽ xend. Когда стрела выпущена, она летит в пространстве бесконечное время (уничтожая все шарики на пути). Дан массив points, где points[i] = [x start, x end]. Напишите функцию, возвращающую минимальное количество стрел, которые нужно выпустить, чтобы уничтожить все шарики.

• Задача 5. «Стопки монет»

На столе стоят 3n стопок монет. Вы и ваши друзья Алиса и Боб забираете

стопки монет по следующему алгоритму:

1. Вы выбираете 3 стопки монет из оставшихся на столе.

2. Алиса забирает себе стопку с максимальным количеством монет.

3. Вы забираете одну из двух оставшихся стопок.

4. Боб забирает последнюю стопку.

5. Если еще остались стопки, то действия повторяются с первого шага.

Дан массив целых положительных чисел piles. Напишите функцию,

возвращающую максимальное число монет, которое вы можете получить.

• Задача 6. «Перестановка строк»

Даны две строки: s1 и s2 с одинаковым размером, проверьте, может ли

некоторая перестановка строки s1 “победить” некоторую перестановку строки

s2 или наоборот. Строка x может “победить” строку y (обе имеют размер n),

если x[i]> = y[i] (в алфавитном порядке) для всех i от 0 до n-1.

• Задача 7. «Самая длинная полиндромная подстрока»

Дана строка s, вернуть самую длинную полиндромную подстроку в s.

• Задача 8. «Непустые подстроки текста»

Вернуть количество отдельных непустых подстрок текста, которые

могут быть записаны как конкатенация некоторой строки с самой собой (т.е.

она может быть записана, как a + a, где a - некоторая строка).

Листинг программы

public static int[][] createmat(int m, int n, int minl, int maxl){  
 final Random random = new Random();  
 int[][] a = new int[m][n];  
  
 for (int i = 0;i<a.length;i++){  
 for (int j = 0;j<a[0].length; j++){  
 a[i][j]= random.nextInt(maxl+Math.*abs*(minl))+minl;  
 }  
 }  
 return a;  
}  
  
public static int[] create(int n, int minl, int maxl){  
 final Random random = new Random();  
  
 int[] a = new int[n];  
 for (int i = 0;i<a.length;i++){  
 a[i]= random.nextInt(maxl+Math.*abs*(minl))+minl;  
 }  
 return a;  
}  
  
public static int[][] diagonal(int[][] a){  
 for (int i = 0; i < a.length; i++) {  
 for (int j = 0; j < a[1].length; j++) {  
 *sort*(a,i,j);  
 }  
  
 }  
 return a;  
}  
  
public static void sort(int [][] a, int i, int j){  
  
  
 if (i==0 || j==0){}else{  
 if(a[i][j]<a[i-1][j-1]){  
 int k = a[i][j];  
 a[i][j]=a[i-1][j-1];  
 a[i-1][j-1]=k;  
 }  
 *sort*(a,i-1,j-1);  
 }  
}  
  
public static String maxdigit(int[] mas){  
  
 String str = "";  
 int k=0;  
 for (int i= 0;i<mas.length;i++){  
 for (int j = 1; j < mas.length-i; j++) {  
  
 String XY = Integer.*toString*(mas[j-1])+Integer.*toString*(mas[j]);  
 String YX = Integer.*toString*(mas[j])+Integer.*toString*(mas[j-1]);  
 if (XY.compareTo(YX)<0) {  
 k=mas[j];mas[j]=mas[j-1];mas[j-1]=k;  
 }  
 }  
 }  
  
 for (int item:mas) {  
 str+=Integer.*toString*(item);  
 }  
 return str;  
}  
  
public static int maxP(int[] mas){  
 Arrays.*sort*(mas);  
 int P = 0;  
 for (int i = mas.length-1; i >=2 ; i--) {  
 if(mas[i]<mas[i-1]+mas[i-2]){  
 P=mas[i]+mas[i-1]+mas[i-2];  
 }  
 }  
 return P;  
}  
  
public static int balloons(int [][] points){  
 if (points.length == 0) return 0;  
 Arrays.*sort*(points, (a,b) -> Integer.*compare*(a[1],b[1]));  
 int result = 0;  
 long end = Long.*MIN\_VALUE*;  
 for( int[] p: points){  
 if( p[0]> end){  
 end = p[1];  
 result++;  
 }  
 }  
 return result;  
}  
  
public static boolean stringBattle(String s1, String s2){  
 s1 = s1.toLowerCase();  
 s2 = s2.toLowerCase();  
 char[] chars1 = s1.toCharArray();  
 char[] chars2 = s2.toCharArray();  
 for (int i = 0; i<chars1.length;i++){  
 for (int j = 0; j < chars2.length-1-i; j++){  
 if(chars1[j]>chars1[j+1]){  
 char tmp = chars1[j];  
 chars1[j]= chars1[j+1];  
 chars1[j+1] = tmp;  
 }  
  
 if(chars2[j]>chars2[j+1]){  
 char tmp = chars2[j];  
 chars2[j]= chars2[j+1];  
 chars2[j+1] = tmp;  
 }  
 }  
 }  
 int counter1 = 0;  
 int counter2 = 0;  
 for(int i = 0; i< chars1.length; i++){  
 if(chars1[i]>=chars2[i]) counter1++;  
 if(chars1[i]<=chars2[i]) counter2++;  
 }  
  
 return (Math.*abs*(counter1)==chars1.length)||(Math.*abs*(counter2)==chars2.length);  
  
}  
  
public static int coins(Integer[] piles){  
 Arrays.*sort*(piles);  
 List<Integer> list = new ArrayList<>(Arrays.*asList*(piles)) ;  
 int max = 0;  
 while (!list.isEmpty()){  
 max+=list.get(list.size()-2);  
 list.remove(list.size()-1);  
 list.remove(list.size()-2);  
 list.remove(0);  
 }  
 return max;  
}  
  
public static String longestPalindrome(String str){  
 String sub = "";  
 String result = "";  
 for(int i = 0;i<str.length();i++){  
 for (int j = i;j < str.length();j++){  
 sub+=str.charAt(j);  
 if(*isPali*(sub)&&(result.length()<sub.length())){  
 result = sub;  
 }  
 }  
 sub = "";  
 }  
 return result;  
}  
  
public static boolean isPali(String str){  
 String s2="";  
 for(int i = str.length()-1;i>=0;i--){  
 s2+=str.charAt(i);  
 }  
 return str.equals(s2);  
}  
  
public static int isConcat(String s){  
 int count = 0;  
 ArrayList<String> substrings = new ArrayList<>();  
 String substring = "";  
 for(int i = 0; i < s.length(); i++) {  
 substring = "";  
 for(int j = i; j < s.length(); j++) {  
 substring += s.charAt(j);  
 if(substring.length() % 2 == 0) {  
 if(!substrings.contains(substring) && substring.equals(substring.substring(0, substring.length()/2) + substring.substring(0, substring.length()/2))) {  
 substrings.add(substring);  
 count++;  
 }  
 }  
 }  
 }  
 return count;  
}

Вывод

Я реализовал разного вида задачи и подкрепил свои знания по теме структуры и алгоритмы обработки данных, рекурсии, работа со строками и тд.