|  |  |  |  |
| --- | --- | --- | --- |
| 实验地点 | 专业班级 | 学号 | 姓名 |
| 宿舍 | 软件工程三班 | 2220202929 | 蔡博宇 |

2-1十进制转换为二进制

一、实验环境

编程语言： Java/Python

IDE环境： IDEA/Pycharm

二、课设题目

请将十进制的数n转化成2进制的数m

三、分析问题、问题的模型

如果直接循环来解，可能会造成最后所得数倒序，所以需要翻转前后顺序。

四、解题思路、方案

4.1方案实现

4.1.1 选用的算法策略、使用该策略的理由

栈+循环，将每次得到的数字加入栈中，最后再依次弹出，即可获得倒叙数字。在python中直接用reserve 函数实现。

4.1.2 算法的时间复杂度、空间复杂度分析

时间复杂度：O（n）

空间复杂度：O（n）:栈空间

4.2 Java 实现程序源代码

|  |
| --- |
| import java.util.Scanner;  import java.util.Stack;  public class 进制转换 {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  print(n);  }  static void print(int n){  Stack<Integer> a = new Stack<Integer>();  int s;  while (n != 0) {  s = n % 2;  n = n / 2;  a.push(s);  }  StringBuilder ans = new StringBuilder();  while (!a.isEmpty()){  ans.append(a.pop());  }  System.out.println(ans);  }  } |

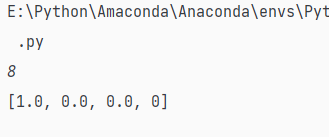
4.2.1程序运行结果：



4.3 python 实现

|  |
| --- |
| def conversion(n):  a = []  while n >= 1:  s = n % 2  n = n / 2  a.append(s)  print(n)  a.reverse() ## python 中直接调用函数实现  print(a)  if \_\_name\_\_ == '\_\_main\_\_':  n = int(input())  conversion(n) |

4.3.1 程序运行结果



五、结论

本题熟悉并掌握了循环的妙用，并通过将变量声明在循环外，重复使用空间，显著减少了空间复杂度。也感受到了数据结构stack对于算法的重要性。

2-2 零钱问题

一、实验环境

编程语言： Java/Python

IDE环境： IDEA/Pycharm

二、课设题目

一个顾客买了价值x元的商品，并将y元的钱交给售货员。售货员希望用张数最少的钱币找给顾客。

三、分析问题、问题的模型

1）贪心解法，从较大钱币遍历即可

2）需要使用数组来记录使用钱币的张数

3）声明一个数组来固定钱币的位置，可方便记录

四、解题思路、方案

4.1方案实现

4.1.1 选用的算法策略、使用该策略的理由

从大面额钱币开始循环，这样可保证钱币数最少

4.1.2 算法的时间复杂度、空间复杂度分析

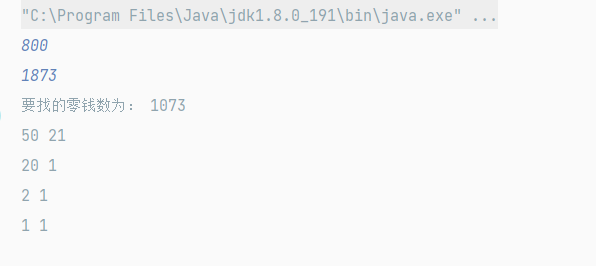
时间复杂度：O（n）

空间复杂度：O（0） 开辟的钱币面额与记录张数的数组忽略不计

4.2 Java 实现程序源代码

|  |
| --- |
| import java.util.Scanner;  public class change {  static int[] array = {50,20,10,5,2,1};  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  int x = scanner.nextInt();  int y = scanner.nextInt();  print(x,y);  }  static void print(int x,int y){  int z = y - x;  int a;  int[] s = new int[array.length];  System.out.println("要找的零钱数为： " + z );  for (int i = 0; i < array.length; i++) {  a = z / array[i];  s[i] = s[i] + a;  z -= a \* array[i];  }  for (int i = 0; i < array.length ; i++) {  if (s[i]!=0) {  System.out.println(array[i] + " " + s[i]);  }  }  }  } |

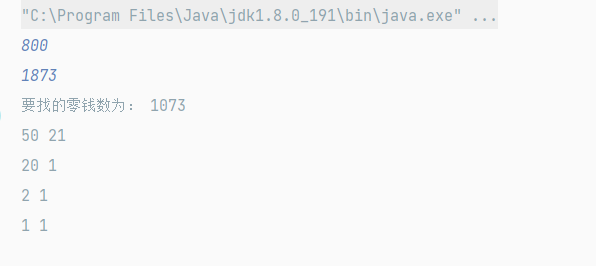
4.2.1程序运行结果：



4.3 python 实现

|  |
| --- |
| def change(x,y):  for i in range(1000):  s = i  for j in range(1,i):  if i % j == 0:  s -= j  a = z / array[i]  s[i] = s[i] + a  z -= a \* array[i]  if s == 0 and i != 0:  print(i)  if \_\_name\_\_ == '\_\_main\_\_':  x = int(input())  y = int(input())  change(x,y) |

4.3.1 程序运行结果



五、结论

此题用贪心的思想，大大简化了判断逻辑，让程序更加简单易懂。同时也感受到了在实际编程中，程序的简易性能大大代替人在实际思考中的诸多细节，让代码更简单的解决复杂度问题。

2-3 x^2 各不同

一、实验环境

编程语言： Java/Python

IDE环境： IDEA/Pycharm

二、课设题目

求x，使x2为一个各位数字互不相同的九位数。

三、分析问题、问题的模型

由九位数可知，该x的范围大致在10000~32000范围内，然后只需要数字各不同即可，我们需要使用一个0~9的数组来判断各个数字是否已经使用，在此规定1为没有使用，而0则是使用，然后使用一个数来计数，每使用一个数就加1，如果最后结果为9，则找到了结果x。

四、解题思路、方案

4.1方案实现

4.1.1 选用的算法策略、使用该策略的理由

循环+标志位

4.1.2 算法的时间复杂度、空间复杂度分析

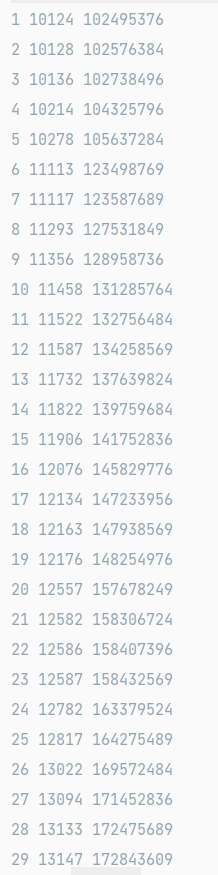
时间复杂度：O（n） 0~9的循环忽略

空间复杂度：O（0）

4.2 Java 实现程序源代码

|  |
| --- |
| public class square {  public static void main(String[] args) {  print();  }  static void print(){  int[] s = new int[10];  int k = 0; // 0~9 的个数  int y1,y2 = 0; // 记录平方数  int t; // 记录下标  int num = 0; // 符合要求的个数  for (int i = 10000; i < 32000; i++ ) {  y1 = i \* i;  y2 = y1;  k = 0;  for (int j = 0; j <=9 ;j++) {  s[j] = 1;  }  for (int j = 0; j <= 9;j++ ) {  t = y2 % 10;  y2 = y2 / 10;  if(s[t] == 1){  k++;  s[t] = 0;  }  }  if (k == 9) {  num++;  System.out.println(num+" " + i + " " + y1 );  }  }  }  } |

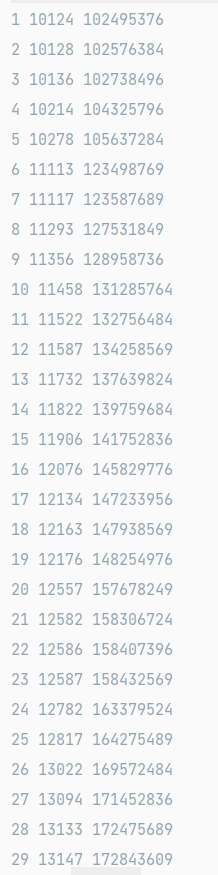
4.2.1程序运行结果：



4.3 python 实现

|  |
| --- |
| def squre(n, m):  s[]  k = 0; // 0~9 的个数  y1,y2 = 0; // 记录平方数  t; // 记录下标  num = 0; // 符合要求的个数  for i = 10000 in range (32000):  y1 = i \* i;  y2 = y1;  k = 0;  for j in range (9):  s[j] = 1;  for j in range(9):  t = y2 % 10;  y2 = y2 / 10;  if(s[t] == 1):  k++;  s[t] = 0;  if (k == 9):  num++;  print(num+" " + i + " " + y1 );  if \_\_name\_\_ == '\_\_main\_\_':  nn = input()  nn = int(nn)  square(divider(nn, nn)) |

4.3.1 程序运行结果



五、结论

本题标志位的使用，让原本复杂的逻辑变简单，其次由九位数的估算，可确定结果的大致范围，这样大大减少了循环和判断次数，从而让程序效率变高

3-4 园圈报数退出问题

一、实验环境

编程语言： Java/Python

IDE环境： IDEA/Pycharm

二、课设题目

n个小朋友手拉手站成一个圆圈，从第k个小朋友开始报数，报到m的那个小朋友退到圈外，然后他的下一位重新报 “1”。这样继续下去，直到最后只剩下一个小朋友，求解 这个小朋友原来站在什么位置上?

三、分析问题、问题的模型

此题的难点在于每个人出圈后，排列顺序会改变。首先用标志位1代表初始未出圈的位置，这样当出圈时设置为0，这样我们就可以进行累加，当累加到数m，就让该下标的数为0。最后依次循环知道只有一个数为1的下标为止。

四、解题思路、方案

4.1方案实现

4.1.1 选用的算法策略、使用该策略的理由

数组标志位

4.1.2 算法的时间复杂度、空间复杂度分析

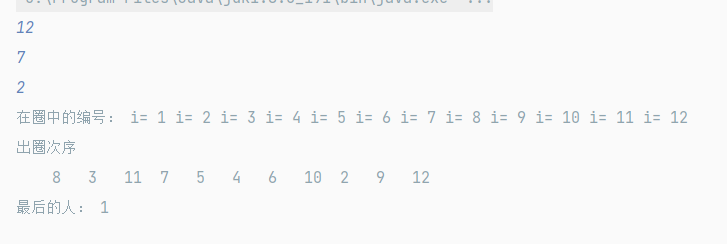
时间复杂度：O（n ）

空间复杂度：O（n） n个人数组空间

4.2 Java 实现程序源代码

|  |
| --- |
| public class circle {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  int n = scanner.nextInt();  int m = scanner.nextInt();  int k = scanner.nextInt();  print(n,m,k);  }  static void print(int n,int m,int k){  int[] ans = new int[100];  for (int i = 1; i <= n ; i++) {  ans[i] = 1;  }  int p =0;  k--;  int x = 0;  System.out.print("在圈中的编号： ");  for (int i = 1; i <= n; i++) {  if (ans[i] == 1) {  System.out.print("i= " + i + " ");  }  }  System.out.println();  System.out.println("出圈次序");  while (p < n-1){  x = 0;  while (x < m){  k++;  if(k > n){  k = 1;  }  x = x + ans[k];  }  System.out.print("\t" + k);  ans[k] = 0;  p++;  }  System.out.println();  for (int i = 1; i <= n; i++) {  if (ans[i] == 1){  System.out.println("最后的人： " + i );  }  }  }  } |

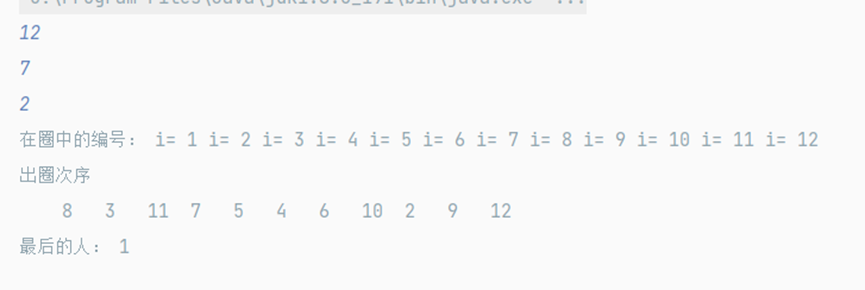
4.2.1程序运行结果：



4.3 python 实现

|  |
| --- |
| def constitute():  global a  a = []  global total  total = 0  for r in range(0, 100):  a.append(0)  n = int(input())  r = int(input())  a[0] = r  comb(n, r)  if n < r:  return  print(total)  def comb(n, r):  global total  for i in range(n, r-1, -1):  a[r] = i  if r > 1:  comb(i - 1, r - 1)  else:  for j in range(a[0], 0, -1):  print(a[j], end=" ")  total += 1  print(end="\n")  if \_\_name\_\_ == '\_\_main\_\_':  constitute() |

4.3.1 程序运行结果



五、结论

本题再次体会到标志位的灵活使用，也感受到了python和Java编码的区别，同时更加熟悉了这两语言的特性，在使用不同语言之余，也感受到算法思想对于每种语言也是一样的。同时感叹算法对于处理实际问题的优越性，感叹计算机对于人类生活的便利。