目录

[一、入门训练 3](#_Toc382391318)

[1、Fibonacci数列（入门 数列 取模） 3](#_Toc382391319)

[2、圆的面积（入门 实数 输出） 4](#_Toc382391320)

[3、序列求和（入门 求和 公式） 5](#_Toc382391321)

[4、A+B问题(入门) 6](#_Toc382391322)

[二、基础练习 10](#_Toc382391323)

[1、数列排序(数组 排序) 10](#_Toc382391324)

[2、十六进制转八进制(进制转换 字符 循环) 11](#_Toc382391325)

[3、十六进制转十进制(进制转换 字符处理 判断) 16](#_Toc382391326)

[4、十进制转十六进制(循环 整除 求余 判断) 17](#_Toc382391327)

[5、特殊回文数(回文数 循环 条件语句) 18](#_Toc382391328)

[6、回文数  (循环 判断 回文数) 20](#_Toc382391329)

[7、特殊的数字(循环 判断 数位) 21](#_Toc382391330)

[8、杨辉三角形(基础练习 二维数组) 21](#_Toc382391331)

[9、查找整数(循环 判断) 23](#_Toc382391332)

[10、数列特征(循环 最大值 最小值 累加) 24](#_Toc382391333)

[11、字母图形(循环 字符串) 26](#_Toc382391334)

[12、01字串(循环) 28](#_Toc382391335)

[13、闰年判断  (条件判断) 29](#_Toc382391336)

[三、算法训练 30](#_Toc382391337)

[1、送分啦 30](#_Toc382391338)

[2、操作格子(线段树) 31](#_Toc382391339)

[3、逆序对(平衡二叉树) 34](#_Toc382391340)

[4、安慰奶牛(最小生成树) 38](#_Toc382391341)

[5、最短路 42](#_Toc382391342)

[6、结点选择(树形动态规划) 44](#_Toc382391343)

[7、K好数  (动态规划) 47](#_Toc382391344)

[8、最大最小公倍数(贪心) 49](#_Toc382391345)

[9、区间k大数查询(排序、查找) 50](#_Toc382391346)

[四、算法提高 52](#_Toc382391347)

[1、最小方差生成树(最小生成树) 52](#_Toc382391348)

[2、道路和航路(最短路) 53](#_Toc382391349)

[3、金属采集(树形动态规划) 59](#_Toc382391350)

[4、矩阵翻转(枚举 贪心) 62](#_Toc382391351)

[5、两条直线(排序) 64](#_Toc382391352)

[五、历届试题 67](#_Toc382391353)

[1、核桃的数量(最小公倍数) 67](#_Toc382391354)

[2、打印十字图(文字图形) 68](#_Toc382391355)

[3、带分数(搜索) 72](#_Toc382391356)

[4、剪格子(搜索) 76](#_Toc382391357)

[5、错误票据(排序) 80](#_Toc382391358)

[6、翻硬币(贪心) 82](#_Toc382391359)

[7、连号区间数(并查集) 84](#_Toc382391360)

[8、买不到的数目(数论 动态规划) 86](#_Toc382391361)

[9、大臣的旅费(深度优先遍历) 87](#_Toc382391362)

[10、幸运数(堆) 91](#_Toc382391363)

[11、横向打印二叉树(排序二叉树) 93](#_Toc382391364)

[12、危险系数(割点) 97](#_Toc382391365)

[13、网络寻路(构图) 100](#_Toc382391366)

[14、高僧斗法(博弈论) 102](#_Toc382391367)

[15、格子刷油漆(动态规划) 104](#_Toc382391368)

[16、农场阳光（计算几何） 106](#_Toc382391369)

[17、约数倍数选卡片（博弈论） 107](#_Toc382391370)

[18、车轮轴迹（计算几何） 108](#_Toc382391371)

[19、九宫重排  （排序） 111](#_Toc382391372)

[20、公式求值（组合数字） 114](#_Toc382391373)

# 一、入门训练

## 1、Fibonacci数列（入门 数列 取模）

问题描述

Fibonacci数列的递推公式为：Fn=Fn-1+Fn-2，其中F1=F2=1。

当n比较大时，Fn也非常大，现在我们想知道，Fn除以10007的余数是多少。

输入格式

输入包含一个整数n。

输出格式

输出一行，包含一个整数，表示Fn除以10007的余数。

说明：在本题中，答案是要求Fn除以10007的余数，因此我们只要能算出这个余数即可，而不需要先计算出Fn的准确值，再将计算的结果除以10007取余数，直接计算余数往往比先算出原数再取余简单。

样例输入

10

样例输出

55

样例输入

22

样例输出

7704

数据规模与约定

1 <= n <= 1,000,000。

本题的C参考代码如下：

1. #include <stdlib.h>
2. #include <stdio.h>
3. #define MOD 10007
4. #define MAXN 1000001
5. int n, i, F[MAXN];
6. int main()
7. {
8. scanf("%d", &n);
9. F[1] = 1;
10. F[2] = 1;
11. for (i = 3; i <= n; ++i)
12. F[i] = (F[i-1] + F[i-2]) % MOD;
13. printf("%d\n", F[n]);
14. return 0;
15. }

## 2、圆的面积（入门 实数 输出）

问题描述

给定圆的半径r，求圆的面积。

输入格式

输入包含一个整数r，表示圆的半径。

输出格式

输出一行，包含一个实数，四舍五入保留小数点后7位，表示圆的面积。

说明：在本题中，输入是一个整数，但是输出是一个实数。

对于实数输出的问题，请一定看清楚实数输出的要求，比如本题中要求保留小数点后7位，则你的程序必须**严格的**输出7位小数，输出过多或者过少的小数位数都是不行的，都会被认为错误。

实数输出的问题如果没有特别说明，舍入都是按四舍五入进行。

样例输入

4

样例输出

50.2654825

数据规模与约定

1 <= r <= 10000。

提示

本题对精度要求较高，请注意π的值应该取较精确的值。你可以使用常量来表示π，比如PI=3.14159265358979323，也可以使用数学公式来求π，比如PI=atan(1.0)\*4。

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <math.h>
3. int main()
4. {
5. int r;
6. double s, PI;
7. scanf("%d", &r);
8. PI = atan(1.0) \* 4;
9. s = PI \* r \* r;
10. printf("%.7lf", s);
11. return 0;
12. }

## 3、序列求和（入门 求和 公式）

问题描述

求1+2+3+...+n的值。

输入格式

输入包括一个整数n。

输出格式

输出一行，包括一个整数，表示1+2+3+...+n的值。

样例输入

4

样例输出

10

样例输入

100

说明：有一些试题会给出多组样例输入输出以帮助你更好的做题。

一般在提交之前所有这些样例都需要测试通过才行，但这不代表这几组样例数据都正确了你的程序就是完全正确的，潜在的错误可能仍然导致你的得分较低。

样例输出

5050

数据规模与约定

1 <= n <= 1,000,000,000。

说明：请注意这里的数据规模。

本题直接的想法是直接使用一个循环来累加，然而，当数据规模很大时，这种“暴力”的方法往往会导致超时。此时你需要想想其他方法。你可以试一试，如果使用1000000000作为你的程序的输入，你的程序是不是能在规定的上面规定的时限内运行出来。

本题另一个要值得注意的地方是答案的大小不在你的语言默认的整型(int)范围内，如果使用整型来保存结果，会导致结果错误。

如果你使用C++或C语言而且准备使用printf输出结果，则你的格式字符串应该写成%I64d以输出long long类型的整数。

本题的C参考代码如下：

1. #include <stdio.h>
2. int main()
3. {
4. int n;
5. long long ret = 0, tmp;
6. scanf("%d", &n);
7. tmp = n;
8. printf("%I64d", (1+tmp) \* tmp / 2);
9. return 0;
10. }

## 4、A+B问题(入门)

问题描述

输入A、B，输出A+B。

说明：在“问题描述”这部分，会给出试题的意思，以及所要求的目标。

输入格式

输入的第一行包括两个整数，由空格分隔，分别表示A、B。

说明：“输入格式”是描述在测试你的程序时，所给的输入一定满足的格式。

做题时你应该假设所给的输入是一定满足输入格式的要求的，所以你不需要对输入的格式进行检查。多余的格式检查可能会适得其反，使用你的程序错误。

在测试的时候，系统会自动将输入数据输入到你的程序中，你不能给任何提示。比如，你在输入的时候提示“请输入A、B”之类的话是不需要的，这些多余的输出会使得你的程序被判定为错误。

输出格式

输出一行，包括一个整数，表示A+B的值。

说明：“输出格式”是要求你的程序在输出结果的时候必须满足的格式。

在输出时，你的程序必须满足这个格式的要求，不能少任何内容，也不能多任何内容。如果你的内容和输出格式要求的不一样，你的程序会被判断为错误，包括你输出了提示信息、中间调试信息、计时或者统计的信息等。

样例输入

12 45

说明：“样例输入”给出了一组满足“输入格式”要求的输入的例子。

这里给出的输入只是可能用来测试你的程序的一个输入，在测试的时候，还会有更多的输入用来测试你的程序。

样例输出

57

说明：“样例输出”给出了一组满足“输出格式”要求的输出的例子。

样例输出中的结果是和样例输入中的是对应的，因此，你可以使用样例的输入输出简单的检查你的程序。

要特别指出的是，能够通过样例输入输出的程序并不一定是正确的程序，在测试的时候，会用很多组数据进行测试，而不局限于样例数据。有可能一个程序通过了样例数据，但测试的时候仍只能得0分，可能因为这个程序只在一些类似样例的特例中正确，而不具有通用性，再测试更多数据时会出现错误。

比如，对于本题，如果你写一个程序不管输入是什么都输入57，则样例数据是对的，但是测试其他数据，哪怕输入是1和2，这个程序也输出57，则对于其他数据这个程序都不正确。

数据规模与约定

-10000 <= A, B <= 10000。

说明：“数据规模与约定”中给出了试题中主要参数的范围。

这个范围对于解题非常重要，不同的数据范围会导致试题需要使用不同的解法来解决。比如本题中给的A、B范围不大，可以使用整型(int)来保存，如果范围更大，超过int的范围，则要考虑其他方法来保存大数。

有一些范围在方便的时候是在“问题描述”中直接给的，所以在做题时不仅要看这个范围，还要注意问题描述。

提示

本题的C++源代码如下：

1. #include <iostream>
3. using namespace std;
5. int main()
6. {
7. int a, b;
8. cin >> a >> b;
9. cout << a + b;
10. return 0;
11. }

本题的C源代码如下：

1. #include <stdio.h>
3. int main()
4. {
5. int a, b;
6. scanf("%d%d", &a, &b);
7. printf("%d", a+b);
8. return 0;
9. }

本题的Java源代码如下：

1. import java.util.\*;
3. public class Main
4. {
5. public static void main(String args[])
6. {
7. Scanner sc = new Scanner(System.in);
8. Integer a = sc.nextInt();
9. Integer b = sc.nextInt();
10. System.out.println(a + b);
11. }
12. }

说明：要答题，请点击页面上方的“提交此题”按钮，页面将跳转到提交代码的页面，选择好你的编译语言，将你的编写好的代码粘贴到代码框中，再点击“提交答案”即可。

你的答案提交给系统后系统会自动对你的代码进行判分，并跳转到结果的列表里面，你可以直接从列表中看到你提交的代码的状态，一般几秒钟后就可以看到判分的结果。

本题作为第一题，在提示中已经分别给了C++和Java的代码，你可以直接把这个代码拷贝过去作为自己的代码提交。

请特别注意，Java的主类名必须是Main。

本题的C参考代码如下：

1. #include <stdio.h>
2. int main()
3. {
4. int a, b;
5. scanf("%d%d", &a, &b);
6. printf("%d", a+b);
7. return 0;
8. }

# 二、基础练习

## 1、数列排序(数组 排序)

问题描述

　　给定一个长度为n的数列，将这个数列按从小到大的顺序排列。1<=n<=200

输入格式

　　第一行为一个整数n。  
　　第二行包含n个整数，为待排序的数，每个整数的绝对值小于10000。

输出格式

　　输出一行，按从小到大的顺序输出排序后的数列。

样例输入

5  
8 3 6 4 9

样例输出

3 4 6 8 9

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. void Print(int \*data,int n)
4. {
5. int i;
6. for(i=0;i<n;i++)
7. {
8. printf("%d ",data[i]);
9. }
10. printf("\n");
11. }
12. int Split(int \*data,int pre,int rear)
13. {
14. int value=data[pre];
15. while(pre<rear)
16. {
17. while(data[rear]>=value && pre<rear) rear--;
18. data[pre]=data[rear];
19. while(data[pre]<value && pre<rear) pre++;
20. data[rear]=data[pre];
21. }
22. data[pre]=value;
23. return pre;
24. }
25. //快速排序
26. void QuickSort(int \*data,int pre,int rear)
27. {
28. if(pre<rear)
29. {
30. int mid=Split(data,pre,rear);
31. QuickSort(data,pre,mid-1);
32. QuickSort(data,mid+1,rear);
33. }
34. }
35. int main()
36. {
37. int i;
38. int n;
39. int \*data;
40. scanf("%d",&n);
41. data=(int \*)malloc(sizeof(int)\*n);
42. for(i=0;i<n;i++)
43. {
44. scanf("%d",&data[i]);
45. }
46. QuickSort(data,0,n-1);
47. Print(data,n);
48. return 0;
49. }

## 2、十六进制转八进制(进制转换 字符 循环)

问题描述

　　给定n个十六进制正整数，输出它们对应的八进制数。

输入格式

　　输入的第一行为一个正整数n （1<=n<=10）。  
　　接下来n行，每行一个由0~9、大写字母A~F组成的字符串，表示要转换的十六进制正整数，每个十六进制数长度不超过100000。

输出格式

　　输出n行，每行为输入对应的八进制正整数。

注意

　　输入的十六进制数不会有前导0，比如012A。  
　　输出的八进制数也不能有前导0。

样例输入

2  
39  
123ABC

样例输出

71  
4435274

提示

　　先将十六进制数转换成某进制数，再由某进制数转换成八进制。

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4. #define MaxSize 100000
5. void saveB(char \*b,char c3,char c2,char c1,char c0,int start)
6. {
7. b[start]=c3;
8. b[start+1]=c2;
9. b[start+2]=c1;
10. b[start+3]=c0;
11. }
12. int htob(char \*h,char \*b)
13. {
14. int i,j;
15. int hl=strlen(h);
16. for(i=0;i<hl;i++)
17. switch(h[i])
18. {
19. case '0':
20. {
21. saveB(b,'0','0','0','0',4\*i);
22. break;
23. }
24. case '1':
25. {
26. saveB(b,'0','0','0','1',4\*i);
27. break;
28. }
29. case '2':
30. {
31. saveB(b,'0','0','1','0',4\*i);
32. break;
33. }
34. case '3':
35. {
36. saveB(b,'0','0','1','1',4\*i);
37. break;
38. }
39. case '4':
40. {
41. saveB(b,'0','1','0','0',4\*i);
42. break;
43. }
44. case '5':
45. {
46. saveB(b,'0','1','0','1',4\*i);
47. break;
48. }
49. case '6':
50. {
51. saveB(b,'0','1','1','0',4\*i);
52. break;
53. }
54. case '7':
55. {
56. saveB(b,'0','1','1','1',4\*i);
57. break;
58. }
59. case '8':
60. {
61. saveB(b,'1','0','0','0',4\*i);
62. break;
63. }
64. case '9':
65. {
66. saveB(b,'1','0','0','1',4\*i);
67. break;
68. }
69. case 'A':
70. {
71. saveB(b,'1','0','1','0',4\*i);
72. break;
73. }
74. case 'B':
75. {
76. saveB(b,'1','0','1','1',4\*i);
77. break;
78. }
79. case 'C':
80. {
81. saveB(b,'1','1','0','0',4\*i);
82. break;
83. }
84. case 'D':
85. {
86. saveB(b,'1','1','0','1',4\*i);
87. break;
88. }
89. case 'E':
90. {
91. saveB(b,'1','1','1','0',4\*i);
92. break;
93. }
94. case 'F':
95. {
96. saveB(b,'1','1','1','1',4\*i);
97. break;
98. }
99. }
100. return 4\*hl;
101. }
102. int btoo(char \*b,char \*o,int bl)
103. {
104. int i,j;
105. int ol;
106. int value;
107. if(bl%3==0)
108. ol=bl/3;
109. else
110. ol=bl/3+1;
111. j=bl-1;
112. for(i=ol-1;i>=0;i--)
113. {
114. if(i>0)
115. o[i]=b[j]-48+(b[j-1]-48)\*2+(b[j-2]-48)\*4+48;
116. else
117. {
118. switch(j)
119. {
120. case 2:
121. o[i]=b[j]-48+(b[j-1]-48)\*2+(b[j-2]-48)\*4+48;
122. break;
123. case 1:
124. o[i]=b[j]-48+(b[j-1]-48)\*2+48;
125. break;
126. case 0:
127. o[i]=b[j];
128. break;
129. }
130. }
131. j=j-3;
132. }
133. return ol;
134. }
135. void printO(char \*o,int ol)
136. {
137. int i=0;
138. if(o[0]=='0')
139. i=1;
140. for(;i<ol;i++)
141. {
142. printf("%c",o[i]);
143. }
144. printf("\n");
145. }
146. main()
147. {
148. char h[MaxSize];
149. char b[4\*MaxSize];
150. char o[4\*MaxSize/3+1];
151. int n,i,bl,ol;
152. scanf("%d",&n);
153. getchar();
154. for(i=0;i<n;i++)
155. {
156. gets(h);
157. bl=htob(h,b);
158. ol=btoo(b,o,bl);
160. printO(o,ol);
161. }
162. }

## 3、十六进制转十进制(进制转换 字符处理 判断)

问题描述

　　从键盘输入一个不超过8位的正的十六进制数字符串，将它转换为正的十进制数后输出。  
　　注：十六进制数中的10~15分别用大写的英文字母A、B、C、D、E、F表示。

样例输入

FFFF

样例输出

65535

本题的C参考代码如下：

1. #include<stdio.h>
2. #include<string.h>
3. #include<math.h>
4. int main()
5. {
6. double sum=0,x;
7. char a[8];
8. int len,i=0;
9. gets(a);
10. len=strlen(a);
11. while(len)
12. {
13. if(a[len-1]>='A'&&a[len-1]<='F')
14. x=(a[len-1]-'7')\*pow(16,i++);
15. else
16. x=(a[len-1]-'0')\*pow(16,i++);
17. sum+=x;
18. len--;
19. }
20. printf("%.0lf",sum);
21. return 0;
23. }

## 4、十进制转十六进制(循环 整除 求余 判断)

问题描述

　　十六进制数是在程序设计时经常要使用到的一种整数的表示方式。它有0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F共16个符号，分别表示十进制数的0至15。十六进制的计数方法是满16进1，所以十进制数16在十六进制中是10，而十进制的17在十六进制中是11，以此类推，十进制的30在十六进制中是1E。  
　　给出一个非负整数，将它表示成十六进制的形式。

输入格式

　　输入包含一个非负整数a，表示要转换的数。0<=a<=2147483647

输出格式

　　输出这个整数的16进制表示

样例输入

30

样例输出

1E

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. char data[]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
4. //输出
5. void Print(char \*num,int n)
6. {
7. int i;
8. for(i=n-1;i>=0;i--) printf("%c",num[i]);
9. printf("\0");
10. printf("\n");
11. }
12. //将十六进制数转换为十进制数
13. int Transform(char \*num,long long value)
14. {
15. int n=0;
16. while(value>=16)
17. {
18. num[n++]=data[value%16];
19. value/=16;
20. }
21. num[n++]=data[value%16];
22. return n;
23. }
24. int main()
25. {
26. long long value;
27. char num[10];
28. int n;
29. scanf("%I64d",&value);
30. n=Transform(num,value);
31. Print(num,n);
32. return 0;
33. }

## 5、特殊回文数(回文数 循环 条件语句)

问题描述

　　123321是一个非常特殊的数，它从左边读和从右边读是一样的。  
　　输入一个正整数n， 编程求所有这样的五位和六位十进制数，满足各位数字之和等于n 。

输入格式

　　输入一行，包含一个正整数n。

输出格式

　　按从小到大的顺序输出满足条件的整数，每个整数占一行。

样例输入

52

样例输出

899998  
989989  
998899

数据规模和约定

　　1<=n<=54。

本题的C参考代码如下：

1. #include<stdio.h>
2. int main()
3. {
4. int a,b,c,d,e,f,t,all;
5. scanf("%d",&t);
6. for(a=1;a<10;a++)
7. for(b=0;b<10;b++)
8. for(c=0;c<10;c++)
9. for(d=0;d<10;d++)
10. for(e=0;e<10;e++)
11. {
12. if(a==e)
13. if(b==d)
14. {
15. all=a+b+c+d+e;
16. if(all==t)
17. printf("%d\n",a\*10000+b\*1000+c\*100+d\*10+e);
18. }
19. }
20. for(a=1;a<10;a++)
21. for(b=0;b<10;b++)
22. for(c=0;c<10;c++)
23. for(d=0;d<10;d++)
24. for(e=0;e<10;e++)
25. for(f=0;f<10;f++)
26. {
27. if(a==f)
28. if(b==e)
29. if(c==d)
30. {
31. all=a+b+c+d+e+f;
32. if(all==t)
33. printf("%d\n",a\*100000+b\*10000+c\*1000+d\*100+e\*10+f);
34. }
35. }
37. return 0;
38. }

## 6、回文数  (循环 判断 回文数)

问题描述

　　1221是一个非常特殊的数，它从左边读和从右边读是一样的，编程求所有这样的四位十进制数。

输出格式

　　按从小到大的顺序输出满足条件的四位十进制数。

本题的C参考代码如下：

1. #include<stdio.h>
2. int main()
3. {
4. int a,b,c,d,i;
5. for(i=1000;i<=9999;i++)
6. {
7. a=i/1000;
8. b=i/100%10;
9. c=i/10%10;
10. d=i%10;
11. if(a==d&&b==c)
12. printf("%d\n",i);
13. }
14. return 0;
15. }

## 7、特殊的数字(循环 判断 数位)

问题描述

　　153是一个非常特殊的数，它等于它的每位数字的立方和，即153=1\*1\*1+5\*5\*5+3\*3\*3。编程求所有满足这种条件的三位十进制数。

输出格式

　　按从小到大的顺序输出满足条件的三位十进制数，每个数占一行。

本题的C参考代码如下：

1. #include<stdio.h>
2. int main()
3. {
4. int i,j,k,n;
5. for(n=100;n<1000;n++)
6. {
7. i=n/100;
8. j=n/10%10;
9. k=n%10;
10. if(i\*100+j\*10+k==i\*i\*i+j\*j\*j+k\*k\*k)
11. printf("%d\n",n);
12. }
13. return 0;
14. }

## 8、杨辉三角形(基础练习 二维数组)

问题描述

杨辉三角形又称Pascal三角形，它的第i+1行是(a+b)i的展开式的系数。

它的一个重要性质是：三角形中的每个数字等于它两肩上的数字相加。

下面给出了杨辉三角形的前4行：

   1

  1 1

 1 2 1

1 3 3 1

给出n，输出它的前n行。

输入格式

输入包含一个数n。

输出格式

输出杨辉三角形的前n行。每一行从这一行的第一个数开始依次输出，中间使用一个空格分隔。请不要在前面输出多余的空格。

样例输入

4

样例输出

1  
1 1  
1 2 1  
1 3 3 1

数据规模与约定

1 <= n <= 34。

本题的C参考代码如下：

1. #include <stdio.h>
2. #define MAXN 40
3. int n;
4. int a[MAXN][MAXN];
5. int main()
6. {
7. int i, j;
8. scanf("%d", &n);
9. a[0][0] = 1;
10. for (i = 0; i < n; ++i)
11. {
12. a[i][0] = a[i][i] = 1;
13. for (j = 1; j < i; ++j)
14. a[i][j] = a[i-1][j-1] + a[i-1][j];
15. }
16. for (i = 0; i < n; ++i)
17. {
18. for (j = 0; j <= i; ++j)
19. printf("%d ", a[i][j]);
20. printf("\n");
21. }
22. return 0;
23. }

## 9、查找整数(循环 判断)

问题描述

给出一个包含n个整数的数列，问整数a在数列中的第一次出现是第几个。

输入格式

第一行包含一个整数n。

第二行包含n个非负整数，为给定的数列，数列中的每个数都不大于10000。

第三行包含一个整数a，为待查找的数。

输出格式

如果a在数列中出现了，输出它第一次出现的位置(位置从1开始编号)，否则输出-1。

样例输入

6  
1 9 4 8 3 9  
9

样例输出

2

数据规模与约定

1 <= n <= 1000。

本题的C参考代码如下：

1. #include <stdio.h>
2. #define MAXN 10001
3. int n, a, ans;
4. int s[MAXN];
5. int main()
6. {
7. int i;
8. scanf("%d", &n);
9. for (i = 0; i < n; ++i)
10. scanf("%d", &s[i]);
11. scanf("%d", &a);
12. ans = -1;
13. for (i = 0; i < n; ++i)
14. {
15. if (s[i] == a)
16. {
17. ans = i + 1;
18. break;
19. }
20. }
21. printf("%d", ans);
22. return 0;
23. }

## 10、数列特征(循环 最大值 最小值 累加)

问题描述

给出n个数，找出这n个数的最大值，最小值，和。

输入格式

第一行为整数n，表示数的个数。

第二行有n个数，为给定的n个数，每个数的绝对值都小于10000。

输出格式

输出三行，每行一个整数。第一行表示这些数中的最大值，第二行表示这些数中的最小值，第三行表示这些数的和。

样例输入

5  
1 3 -2 4 5

样例输出

5  
-2  
11

数据规模与约定

1 <= n <= 10000。

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. #define MAXSIZE 10000
4. int main()
5. {
6. int i;
7. int n;
8. int value;
9. int max=-MAXSIZE;
10. int min=MAXSIZE;
11. int sum=0;
12. scanf("%d",&n);
13. for(i=0;i<n;i++)
14. {
15. scanf("%d",&value);
16. if(value>max) max=value;
17. if(value<min) min=value;
18. sum+=value;
19. }
20. printf("%d\n",max);
21. printf("%d\n",min);
22. printf("%d\n",sum);
23. return 0;
24. }

## 11、字母图形(循环 字符串)

问题描述

利用字母可以组成一些美丽的图形，下面给出了一个例子：

ABCDEFG

BABCDEF

CBABCDE

DCBABCD

EDCBABC

这是一个5行7列的图形，请找出这个图形的规律，并输出一个n行m列的图形。

输入格式

输入一行，包含两个整数n和m，分别表示你要输出的图形的行数的列数。

输出格式

输出n行，每个m个字符，为你的图形。

样例输入

5 7

样例输出

ABCDEFG  
BABCDEF  
CBABCDE  
DCBABCD  
EDCBABC

数据规模与约定

1 <= n, m <= 26。

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. #define MAXSIZE 26
4. //输出
5. void Print(char matrix[MAXSIZE][MAXSIZE],int n,int m)
6. {
7. int i,j;
8. for(i=0;i<n;i++)
9. {
10. for(j=0;j<m;j++)
11. {
12. printf("%c",matrix[i][j]);
13. }
14. printf("\n");
15. }
16. }
17. //实现字母图形
18. void Drawing(int n,int m)
19. {
20. int i,j;
21. int point=0;
22. char str;
23. char matrix[MAXSIZE][MAXSIZE];
24. for(i=0;i<n;i++)
25. {
26. str='A';
27. for(j=i;j<m;j++)
28. {
29. matrix[i][j]=str++;
30. }
31. str='A';
32. for(j=i-1;j>=0;j--)
33. {
34. matrix[i][j]=++str;
35. }
36. }
37. Print(matrix,n,m);
38. }
39. int main()
40. {
41. int n,m;
42. scanf("%d%d",&n,&m);
43. Drawing(n,m);
44. return 0;
45. }

## 12、01字串(循环)

问题描述

对于长度为5位的一个01串，每一位都可能是0或1，一共有32种可能。它们的前几个是：

00000

00001

00010

00011

00100

请按从小到大的顺序输出这32种01串。

输入格式

本试题没有输入。

输出格式

输出32行，按从小到大的顺序每行一个长度为5的01串。

样例输出

00000  
00001  
00010  
00011  
< 以下部分省略>

本题的C参考代码如下：

1. #include <stdio.h>
2. int main()
3. {
4. int i, j, k, l, m;
5. for (i = 0; i <= 1; ++i)
6. for (j = 0; j <= 1; ++j)
7. for (k = 0; k <= 1; ++k)
8. for (l = 0; l <= 1; ++l)
9. for (m = 0; m <= 1; ++m)
10. printf("%d%d%d%d%d\n", i, j, k, l, m);
11. return 0;
12. }

## 13、闰年判断  (条件判断)

问题描述

给定一个年份，判断这一年是不是闰年。

当以下情况之一满足时，这一年是闰年：

1. 年份是4的倍数而不是100的倍数；

2. 年份是400的倍数。

其他的年份都不是闰年。

输入格式

输入包含一个整数y，表示当前的年份。

输出格式

输出一行，如果给定的年份是闰年，则输出yes，否则输出no。

说明：当试题指定你输出一个字符串作为结果（比如本题的yes或者no，你需要严格按照试题中给定的大小写，写错大小写将不得分。

样例输入

2013

样例输出

no

样例输入

2016

样例输出

yes

数据规模与约定

1990 <= y <= 2050。

本题的C参考代码如下：

1. #include <stdio.h>
2. int main()
3. {
4. int y;
5. scanf("%d", &y);
6. if (y%4==0 && y%100!=0 || y%400==0)
7. printf("yes");
8. else
9. printf("no");
10. return 0;
11. }

# 三、算法训练

## 1、送分啦

问题描述

　　这题想得分吗？想，请输出“yes”；不想，请输出“no”。

输出格式

　　输出包括一行，为“yes”或“no”。

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdio.h>
3. int main()
4. {
5. printf("yes\n");
6. return 0;
7. }

## 2、操作格子(线段树)

问题描述

有n个格子，从左到右放成一排，编号为1-n。

共有m次操作，有3种操作类型：

1.修改一个格子的权值，

2.求连续一段格子权值和，

3.求连续一段格子的最大值。

对于每个2、3操作输出你所求出的结果。

输入格式

第一行2个整数n，m。

接下来一行n个整数表示n个格子的初始权值。

接下来m行，每行3个整数p,x,y，p表示操作类型，p=1时表示修改格子x的权值为y，p=2时表示求区间[x,y]内格子权值和，p=3时表示求区间[x,y]内格子最大的权值。

输出格式

有若干行，行数等于p=2或3的操作总数。

每行1个整数，对应了每个p=2或3操作的结果。

样例输入

4 3  
1 2 3 4  
2 1 3  
1 4 3  
3 1 4

样例输出

6  
3

数据规模与约定

对于20%的数据n <= 100，m <= 200。

对于50%的数据n <= 5000，m <= 5000。

对于100%的数据1 <= n <= 100000，m <= 100000，0 <= 格子权值 <= 10000。

本题的C参考代码如下：

1. #include <stdio.h>
2. #define N 100000
3. #define A 1000
4. #define B 100
5. int sum(int\* a, int m, int n)
6. {
7. int i, s = 0;
8. for (i = m; i <= n; i++)
9. s += a[i];
10. return s;
11. }
12. int max(int\* a, int m, int n)
13. {
14. int i, s = a[m];
15. for (i = m + 1; i <= n; i++)
16. if (s < a[i])
17. s = a[i];
18. return s;
19. }
20. int main()
21. {
22. int i, j, k, m, n;
23. int a[100000], b[100000][3], c[A][2] = {0};
24. scanf("%d%d", &n, &m);
25. for (i = 0; i < n; i++)
26. scanf("%d", &a[i]);
27. for (i = 0; i < m; i++)
28. for (j = 0; j < 3; j++)
29. scanf("%d", &b[i][j]);
30. for (i = 0; i < (n + B - 1) / B; i++)
31. {
32. c[i][0] = c[i][1] = a[i \* B];
33. for (j = i \* B + 1; j < i \* B + B && j < n; j++)
34. {
35. c[i][0] += a[j];
36. if (c[i][1] < a[j])
37. c[i][1] = a[j];
38. }
39. }
40. for (i = 0; i < m; i++)
41. {
42. if (b[i][0] == 1)
43. {
44. c[(b[i][1] - 1) / B][0] += b[i][2] - a[b[i][1] - 1];
45. k = (b[i][1] - 1) / B;
46. if (c[k][1] <= b[i][2])
47. {
48. c[k][1] = b[i][2];
49. }
50. else if (a[b[i][1] - 1] == c[k][1])
51. {
52. a[b[i][1] - 1] = b[i][2];
53. c[k][1] = max(a, k \* B, k \* B + B > n ? n - 1 : k \* B + B - 1);
54. }
55. a[b[i][1] - 1] = b[i][2];
56. }
57. else if (b[i][0] == 2)
58. {
59. int s = 0;
60. b[i][1]--, b[i][2]--;
61. int o = b[i][2] / B - b[i][1] / B;
62. if (o < 2)
63. {
64. s = sum(a, b[i][1], b[i][2]);
65. }
66. else
67. {
68. s = sum(a, b[i][1], (b[i][1] + B) / B \* B - 1);
69. s += sum(a, b[i][2] / B \* B, b[i][2]);
70. for (j = b[i][1] / B + 1; j < b[i][2] / B; j++)
71. s += c[j][0];
72. }
73. printf("%d\n", s);
74. }
75. else if (b[i][0] == 3)
76. {
77. int s = 0, t;
78. b[i][1]--, b[i][2]--;
79. int o = b[i][2] / B - b[i][1] / B;
80. if (o < 2)
81. {
82. s = max(a, b[i][1], b[i][2]);
83. }
84. else
85. {
86. s = max(a, b[i][1], (b[i][1] + B) / B \* B - 1);
87. t = max(a, b[i][2] / B \* B, b[i][2]);
88. if (s < t) s = t;
89. for (j = b[i][1] / B + 1; j < b[i][2] / B; j++)
90. if (s < c[j][1])
91. s = c[j][1];
92. }
93. printf("%d\n", s);
94. }
95. }
96. return 0;
97. }

## 3、逆序对(平衡二叉树)

问题描述

Alice是一个让人非常愉跃的人!他总是去学习一些他不懂的问题，然后再想出许多稀奇古怪的题目。这几天，Alice又沉浸在逆序对的快乐当中，他已近学会了如何求逆序对对数，动态维护逆序对对数等等题目，他认为把这些题让你做简直是太没追求了，于是，经过一天的思考和完善，Alice终于拿出了一道他认为差不多的题目：

有一颗2n-1个节点的二叉树，它有恰好n个叶子节点，每个节点上写了一个整数。如果将这棵树的所有叶子节点上的数从左到右写下来，便得到一个序列a[1]…a[n]。现在想让这个序列中的逆序对数量最少，但唯一的操作就是选树上一个非叶子节点，将它的左右两颗子树交换。他可以做任意多次这个操作。求在最优方案下，该序列的逆序对数最少有多少。

Alice自己已近想出了题目的正解，他打算拿来和你分享，他要求你在最短的时间内完成。

输入格式

第一行一个整数n。

下面每行，一个数x。

如果x=0，表示这个节点非叶子节点，递归地向下读入其左孩子和右孩子的信息，如果x≠0，表示这个节点是叶子节点，权值为x。

输出格式

输出一个整数，表示最少有多少逆序对。

样例输入

3  
0  
0  
3  
1  
2

样例输出

1

数据规模与约定

对于20%的数据，n <= 5000。

对于100%的数据，1 <= n <= 200000，0 <= a[i]<2^31。

本题的C参考代码如下：

1. 该题暂时没有人完全正确，暂时没有该语言的参考程序。

本题的C++参考代码如下：

1. #include<stdio.h>
2. #include<iostream>
3. using namespace std;
4. #define ForD(i,n) for(int i=n;i;i--)
5. #define F (100000007)
6. #define MAXN (2\*200000+10)
7. long long mul(long long a,long long b){return (a\*b)%F;}
8. long long add(long long a,long long b){return (a+b)%F;}
9. long long sub(long long a,long long b){return (a-b+(a-b)/F\*F+F)%F;}
10. int n,root=0;
11. struct node
12. {
13. int fa,ch[2],size,c;
14. node():size(0),c(0){ch[0]=ch[1]=fa=0;}
15. }a[MAXN];
16. void update(int x){a[x].size=a[a[x].ch[0]].size+a[a[x].ch[1]].size+(a[x].c>0);}
17. int tail=0;
18. void pushdown(int x){a[a[x].ch[0]].fa=a[a[x].ch[1]].fa=x;}
19. void build(int &x)
20. {
21. if (!x) x=++tail;
22. scanf("%d",&a[x].c);
23. if (a[x].c==0)
24. {
25. build(a[x].ch[0]);
26. build(a[x].ch[1]);
27. update(x);pushdown(x);
28. }else a[x].size=1;
29. }
30. void rotate(int x)
31. {
32. int y=a[x].fa,z=a[y].fa;
33. bool p=a[y].ch[0]==x;
34. if (z)
35. {
36. if (a[z].ch[0]==y) a[z].ch[0]=x;
37. else a[z].ch[1]=x;
38. }
39. a[x].fa=z,a[y].fa=x;
40. if (a[x].ch[p]) a[a[x].ch[p]].fa=y;
41. a[y].ch[p^1]=a[x].ch[p];
42. a[x].ch[p]=y;
43. update(y);
44. }
45. void splay(int x)
46. {
47. while (a[x].fa)
48. {
49. int y=a[x].fa,z=a[y].fa;
50. if (z)
51. if ((a[y].ch[0]==x)^(a[z].ch[0]==y)) rotate(x);
52. else rotate(y);
53. rotate(x);
54. }
55. update(x);
56. }
57. void ins(long long &tot,int x,int y)
58. {
59. a[x].size++;
60. if (a[y].c<=a[x].c)
61. {
62. if (a[x].ch[0]) ins(tot,a[x].ch[0],y);
63. else a[y].fa=x,splay(a[x].ch[0]=y);
64. }
65. else
66. {
67. tot+=a[a[x].ch[0]].size+(a[x].c>0);
68. if (a[x].ch[1]) ins(tot,a[x].ch[1],y);
69. else a[y].fa=x,splay(a[x].ch[1]=y);
70. }
71. }
72. int q[MAXN],size;
73. void clac(int x,int y)
74. {
75. if (a[y].ch[0]) clac(x,a[y].ch[0]);
76. if (a[y].c) q[++size]=y;
77. if (a[y].ch[1]) clac(x,a[y].ch[1]);
78. }
79. long long merge(bool &lor,int z)
80. {
81. int x=a[z].ch[0],y=a[z].ch[1];
82. if (a[x].size<a[y].size) swap(x,y);
83. a[x].fa=0;a[y].fa=0;q[1]=y;
84. size=0;clac(x,y);
85. long long tot=0;
86. ForD(i,size)
87. {
88. int now=q[i];
89. a[now].ch[0]=a[now].ch[1]=a[now].fa=0;a[now].size=1;
90. ins(tot,x,now);
91. x=now;
92. }
93. a[x].fa=z;
94. a[z].ch[0]=0,a[z].ch[1]=x;
95. return tot;
96. }
97. long long qur(int &x)
98. {
99. if (a[x].c) return 0;
100. else
101. {
102. long long lson=a[a[x].ch[0]].size,rson=a[a[x].ch[1]].size,ls=qur(a[x].ch[0]),rs=qur(a[x].ch[1]);
103. bool lor=0;
104. long long ms=merge(lor,x);
105. return ls+rs+min(lson\*rson-ms,ms);
106. }
107. }
108. int main()
109. {
110. scanf("%d",&n);
111. build(root);
112. cout<<qur(root)<<endl;
113. return 0;
114. }

## 4、安慰奶牛(最小生成树)

问题描述

Farmer John变得非常懒，他不想再继续维护供奶牛之间供通行的道路。道路被用来连接N个牧场，牧场被连续地编号为1到N。每一个牧场都是一个奶牛的家。FJ计划除去P条道路中尽可能多的道路，但是还要保持牧场之间 的连通性。你首先要决定那些道路是需要保留的N-1条道路。第j条双向道路连接了牧场Sj和Ej(1 <= Sj <= N; 1 <= Ej <= N; Sj != Ej)，而且走完它需要Lj的时间。没有两个牧场是被一条以上的道路所连接。奶牛们非常伤心，因为她们的交通系统被削减了。你需要到每一个奶牛的住处去安慰她们。每次你到达第i个牧场的时候(即使你已经到过)，你必须花去Ci的时间和奶牛交谈。你每个晚上都会在同一个牧场(这是供你选择的)过夜，直到奶牛们都从悲伤中缓过神来。在早上 起来和晚上回去睡觉的时候，你都需要和在你睡觉的牧场的奶牛交谈一次。这样你才能完成你的 交谈任务。假设Farmer John采纳了你的建议，请计算出使所有奶牛都被安慰的最少时间。

输入格式

第1行包含两个整数N和P。

接下来N行，每行包含一个整数Ci。

接下来P行，每行包含三个整数Sj, Ej和Lj。

输出格式

输出一个整数, 所需要的总时间(包含和在你所在的牧场的奶牛的两次谈话时间)。

样例输入

5 7  
10  
10  
20  
6  
30  
1 2 5  
2 3 5  
2 4 12  
3 4 17  
2 5 15  
3 5 6

样例输出

176

数据规模与约定

5 <= N <= 10000，N-1 <= P <= 100000，0 <= Lj <= 1000，1 <= Ci <= 1,000。

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. #define maxN 10000
4. #define maxP 100000
5. typedef struct
6. {
7. int u;
8. int v;
9. int w;
10. }Edge;
11. int N,P;
12. int C[maxN];
13. Edge edge[maxP];
14. int parent[maxN];
15. int com(const void \*a, const void \*b)
16. {
17. return ((Edge\*)a)->w-((Edge\*)b)->w;
18. }
19. int find(int x)
20. {
21. int root,temp;
22. for(root=x;parent[root]>=0;root=parent[root])
23. ;
24. while(root!=x)
25. {
26. temp=parent[x];
27. parent[x]=root;
28. x=temp;
29. }
30. return root;
31. }
32. void Union(int x,int y)
33. {
34. int r1,r2;
35. int num;
36. r1=find(x);
37. r2=find(y);
38. num=parent[r1]+parent[r2];
39. if(parent[r1]<parent[r2])
40. {
41. parent[r2]=r1;
42. parent[r1]=num;
43. }
44. else
45. {
46. parent[r1]=r2;
47. parent[r2]=num;
48. }
49. }
50. int Kruskal()
51. {
52. int i;
53. int u,v;
54. int sumweight=0,count=0;
55. for(i=0;i<N;i++)
56. parent[i]=-1;
57. qsort(edge,P,sizeof(edge[0]),com);
58. for(i=0;i<P;i++)
59. {
60. u=edge[i].u;
61. v=edge[i].v;
62. if(find(u)!=find(v))
63. {
64. sumweight+=edge[i].w;
65. Union(u,v);
66. count++;
67. if(count>=N-1)
68. break;
69. }
70. }
71. return sumweight;
72. }
73. int main()
74. {
75. int i;
76. int u,v,w;
77. int min=999999;
78. scanf("%d%d",&N,&P);
79. for(i=0;i<N;i++)
80. {
81. scanf("%d",&C[i]);
82. min=min<C[i]?min:C[i];
83. }
84. for(i=0;i<P;i++)
85. {
86. scanf("%d%d%d",&u,&v,&w);
87. edge[i].u=u-1;
88. edge[i].v=v-1;
89. edge[i].w=w\*2+C[u-1]+C[v-1];
90. }
91. printf("%d",Kruskal()+min);
92. }

## 5、最短路

问题描述

给定一个n个顶点，m条边的有向图（其中某些边权可能为负，但保证没有负环）。请你计算从1号点到其他点的最短路（顶点从1到n编号）。

输入格式

第一行两个整数n, m。

接下来的m行，每行有三个整数u, v, l，表示u到v有一条长度为l的边。

输出格式

共n-1行，第i行表示1号点到i+1号点的最短路。

样例输入

3 3  
1 2 -1  
2 3 -1  
3 1 2

样例输出

-1  
-2

数据规模与约定

对于10%的数据，n = 2，m = 2。

对于30%的数据，n <= 5，m <= 10。

对于100%的数据，1 <= n <= 20000，1 <= m <= 200000，-10000 <= l <= 10000，保证从任意顶点都能到达其他所有顶点。

本题的C参考代码如下：

1. #include"stdio.h"
2. #include"stdlib.h"
3. #define MAXN 1000003
4. #define MAXM 1000003
5. #define oo 1000000000
6. struct edge{
7. int node,len,next;
8. } e[MAXM]={0};
9. int dist[MAXN]={0},n,m,head[MAXN]={0},q[MAXM],tot=0,f[MAXN]={0};
10. void add(int x,int y,int z){
11. e[++tot].node=y;
12. e[tot].len=z;
13. e[tot].next=head[x];
14. head[x]=tot;
15. }
16. void init(){
17. int i,x,y,z;
18. scanf("%d%d",&n,&m);
19. for(i=2;i<=n;i++)
20. dist[i]=oo;
21. for(i=1;i<=m;i++){
22. scanf("%d%d%d",&x,&y,&z);
23. add(x,y,z);
24. }
25. }
26. void spfa(){
27. int op=1,cl=1,i,j,k;
28. q[1]=1;           f[1]=1;
29. while(op>=cl){
30. j=head[q[cl]];
31. while(j){
32. k=e[j].node;
33. if(dist[k]>dist[q[cl]]+e[j].len){
34. dist[k]=dist[q[cl]]+e[j].len;
35. if(!f[k]) f[q[++op]=k]=1;
36. }
37. j=e[j].next;
38. }
39. f[q[cl++]]=0;
40. }
41. for(i=2;i<n;i++)
42. printf("%d\n",dist[i]);
43. printf("%d",dist[i]);
44. }
45. int main(){
46. init();
47. spfa();
48. return 0;
49. }

## 6、结点选择(树形动态规划)

问题描述

有一棵 n 个节点的树，树上每个节点都有一个正整数权值。如果一个点被选择了，那么在树上和它相邻的点都不能被选择。求选出的点的权值和最大是多少？

输入格式

第一行包含一个整数 n 。

接下来的一行包含 n 个正整数，第 i 个正整数代表点 i 的权值。

接下来一共 n-1 行，每行描述树上的一条边。

输出格式

输出一个整数，代表选出的点的权值和的最大值。

样例输入

5  
1 2 3 4 5  
1 2  
1 3  
2 4  
2 5

样例输出

12

样例说明

选择3、4、5号点，权值和为 3+4+5 = 12 。

数据规模与约定

对于20%的数据， n <= 20。

对于50%的数据， n <= 1000。

对于100%的数据， n <= 100000。

权值均为不超过1000的正整数。

本题的C参考代码如下：

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. typedef struct Node
5. {
6. int to;
7. int next;
8. }Node;
9. #define N 100020
10. int max(int a, int b)
11. {
12. return a > b ? a : b;
13. }
14. int on[N], off[N];
15. int rel[N];
16. Node relBus[2 \* N];
17. int relBusTop = 1;
18. int queue[N] = {1};
19. int qStart = 0, qEnd = 1;
20. int checked[N] = {0, 1};
21. int ser[N];
22. int sp = 0;
23. int main(void)
24. {
25. int n, i, j;
27. scanf("%d", &n);
28. for(i = 1; i <= n; i++)
29. {
30. scanf("%d", &on[i]);
31. off[i] = 0;
32. }
34. for(i = 0; i < n - 1; i++)
35. {
36. int a, b;
37. scanf("%d%d", &a, &b);
39. relBus[relBusTop].to = b;
40. relBus[relBusTop].next = rel[a];
41. rel[a] = relBusTop++;
43. relBus[relBusTop].to = a;
44. relBus[relBusTop].next = rel[b];
45. rel[b] = relBusTop++;
46. }
47. while(qStart < qEnd)
48. {
49. int now = queue[qStart++];
50. ser[sp++] = now;
51. int p = rel[now];
52. while(p > 0)
53. {
54. int son = relBus[p].to;
55. if(checked[son] == 0)
56. {
57. queue[qEnd++] = son;
58. checked[son] = 1;
59. }
60. p = relBus[p].next;
61. }
62. }
64. for(i = n - 1; i >= 0; i--)
65. {
66. int son = ser[i];
67. int p = rel[son];
69. while(p > 0)
70. {
71. int father = relBus[p].to;
73. on[father] += off[son];
74. off[father] += max(on[son], off[son]);
75. p = relBus[p].next;
76. }
77. }
79. printf("%d", max(on[1], off[1]));
80. return 0;
81. }

## 7、K好数  (动态规划)

问题描述

如果一个自然数N的K进制表示中任意的相邻的两位都不是相邻的数字，那么我们就说这个数是K好数。求L位K进制数中K好数的数目。例如K = 4，L = 2的时候，所有K好数为11、13、20、22、30、31、33 共7个。由于这个数目很大，请你输出它对1000000007取模后的值。

输入格式

输入包含两个正整数，K和L。

输出格式

输出一个整数，表示答案对1000000007取模后的值。

样例输入

4 2

样例输出

7

数据规模与约定

对于30%的数据，KL <= 106；

对于50%的数据，K <= 16， L <= 10；

对于100%的数据，1 <= K,L <= 100。

本题的C参考代码如下：

1. #include<stdio.h>
2. int main()
3. {
4. int i;
5. int k;      //进制数
6. int l;      //位数
7. long long ka[100];      //前
8. long long kb[100];      //当前
9. long long cont=0;       //计数
10. scanf("%d%d",&k,&l);
11. kb[0]=ka[0]=0;
12. for(i=1;i<k;i++)
13. {
14. kb[i]=ka[i]=1;
15. }
16. for(i=2;i<=l;i++)
17. {
18. int j;
19. for(j=0;j<k;j++)
20. {
21. int m=0;
22. for(m=0;m<k;m++)
23. {
24. if(m<j-1 || m>j+1)
25. kb[j]+=ka[m];
26. }
28. }
29. for(j=0;j<k;j++)
30. {
31. ka[j]=kb[j];
32. ka[j]=kb[j]%1000000007;
33. }
34. }
35. while(k--)
36. {
37. cont+=ka[k];
38. cont=cont%1000000007;
39. }
40. printf("%I64d\n",cont);
41. return 0;
42. }

## 8、最大最小公倍数(贪心)

问题描述

已知一个正整数N，问从1~N中任选出三个数，他们的最小公倍数最大可以为多少。

输入格式

输入一个正整数N。

输出格式

输出一个整数，表示你找到的最小公倍数。

样例输入

9

样例输出

504

数据规模与约定

1 <= N <= 106。

本题的C参考代码如下：

1. 该题暂时没有人完全正确，暂时没有该语言的参考程序。

## 9、区间k大数查询(排序、查找)

问题描述

给定一个序列，每次询问序列中第l个数到第r个数中第K大的数是哪个。

输入格式

第一行包含一个数n，表示序列长度。

第二行包含n个正整数，表示给定的序列。

第三个包含一个正整数m，表示询问个数。

接下来m行，每行三个数l,r,K，表示询问序列从左往右第l个数到第r个数中，从大往小第K大的数是哪个。序列元素从1开始标号。

输出格式

总共输出m行，每行一个数，表示询问的答案。

样例输入

5  
1 2 3 4 5  
2  
1 5 2  
2 3 2

样例输出

4  
2

数据规模与约定

对于30%的数据，n,m<=100；

对于100%的数据，n,m<=1000；

保证k<=(r-l+1)，序列中的数<=106。

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. int Split(int \*data,int pre,int rear)
4. {
5. int value=data[pre];
6. while(pre<rear)
7. {
8. while(data[rear]>=value && pre<rear) rear--;
9. data[pre]=data[rear];
10. while(data[pre]<value && pre<rear) pre++;
11. data[rear]=data[pre];
12. }
13. data[pre]=value;
14. return pre;
15. }
16. //快速排序
17. void QuickSort(int \*data,int pre,int rear,int k)
18. {
19. if(pre<=rear)
20. {
21. int mid=Split(data,pre,rear);
22. if(mid==k)
23. {
24. printf("%d\n",data[mid]);
25. }
26. else if(mid>k)
27. {
28. QuickSort(data,pre,mid-1,k);
29. }
30. else
31. {
32. QuickSort(data,mid+1,rear,k);
33. }
34. }
35. }
36. void Copy(int \*data,int n,int \*temp)
37. {
38. int i;
39. for(i=0;i<n;i++)
40. {
41. temp[i]=data[i];
42. }
43. }
44. int main()
45. {
46. int i;
47. int n;
48. int m;
49. int \*data;
50. scanf("%d",&n);
51. data=(int \*)malloc(sizeof(int)\*n);
52. for(i=0;i<n;i++)
53. {
54. scanf("%d",&data[i]);
55. }
56. scanf("%d",&m);
57. while(m)
58. {
59. int pre;
60. int rear;
61. int k;
62. int \*temp=(int \*)malloc(sizeof(int)\*n);
63. scanf("%d%d%d",&pre,&rear,&k);
64. Copy(data,n,temp);
65. QuickSort(temp,pre-1,rear-1,rear-k);
66. m--;
67. }
68. return 0;
69. }

# 四、算法提高

## 1、最小方差生成树(最小生成树)

问题描述

给定带权无向图，求出一颗方差最小的生成树。

输入格式

输入多组测试数据。第一行为N,M，依次是点数和边数。接下来M行，每行三个整数U,V,W，代表连接U,V的边，和权值W。保证图连通。n=m=0标志着测试文件的结束。

输出格式

对于每组数据，输出最小方差，四舍五入到0.01。输出格式按照样例。

样例输入

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

样例输出

Case 1: 0.22  
Case 2: 0.00

数据规模与约定

1<=U,V<=N<=50,N-1<=M<=1000,0<=W<=50。数据不超过5组。

本题的C参考代码如下：

1. 该题暂时没有人完全正确，暂时没有该语言的参考程序。

## 2、道路和航路(最短路)

问题描述

农夫约翰正在针对一个新区域的牛奶配送合同进行研究。他打算分发牛奶到T个城镇（标号为1..T），这些城镇通过R条标号为（1..R）的道路和P条标号为（1..P）的航路相连。

每一条公路i或者航路i表示成连接城镇Ai（1<=A\_i<=T）和Bi（1<=Bi<=T）代价为Ci。每一条公路，Ci的范围为0<=Ci<=10,000；由于奇怪的运营策略，每一条航路的Ci可能为负的，也就是-10,000<=Ci<=10,000。

每一条公路都是双向的，正向和反向的花费是一样的，都是非负的。

每一条航路都根据输入的Ai和Bi进行从Ai->Bi的单向通行。实际上，如果现在有一条航路是从Ai到Bi的话，那么意味着肯定没有通行方案从Bi回到Ai。

农夫约翰想把他那优良的牛奶从配送中心送到各个城镇，当然希望代价越小越好，你可以帮助他嘛？配送中心位于城镇S中（1<=S<=T）。

输入格式

输入的第一行包含四个用空格隔开的整数T，R，P，S。

接下来R行，描述公路信息，每行包含三个整数，分别表示Ai，Bi和Ci。

接下来P行，描述航路信息，每行包含三个整数，分别表示Ai，Bi和Ci。

输出格式

输出T行，分别表示从城镇S到每个城市的最小花费，如果到不了的话输出NO PATH。

样例输入

6 3 3 4  
1 2 5  
3 4 5  
5 6 10  
3 5 -100  
4 6 -100  
1 3 -10

样例输出

NO PATH  
NO PATH  
5  
0  
-95  
-100

数据规模与约定

对于20%的数据，T<=100，R<=500，P<=500；

对于30%的数据，R<=1000，R<=10000，P<=3000；

对于100%的数据，1<=T<=25000，1<=R<=50000，1<=P<=50000。

本题的C++参考代码如下：

1. #include <iostream>
2. #include <cstdio>
3. #include <cstring>
4. #include <algorithm>
5. #include <queue>
6. #include <stack>
7. #include <vector>
8. #define clr(a,b) memset(a, b, sizeof(a))
9. using namespace std;
10. const int N = 25050;
11. const int E = 150500;
12. //邻接表
13. int h[N], v[E], w[E], nxt[E], el;
14. void initEdge() {
15. clr(h, -1); el = 0;
16. }
17. void addEdge(int x, int y, int z) {
18. v[el] = y; w[el] = z; nxt[el] = h[x]; h[x] = el++;
19. }
20. //belong[i] 表示节点 i 所在的强连通分量；
21. //cnt 表示强连通分量的个数；
22. int dfn[N], sta[N], low[N], belong[N];
23. int top, cnt, ind, n;
24. bool vis[N];
25. void TarjanSolve(int u) {
26. dfn[u] = low[u] = ++ind;
27. vis[u] = true;
28. sta[++top] = u;
29. for(int p=h[u]; ~p; p=nxt[p]) {
30. int i = v[p];
31. if(!dfn[i]) {
32. TarjanSolve(i);
33. if(low[i] < low[u]) low[u] = low[i];
34. }
35. else
36. if(vis[i] && dfn[i] < low[u])
37. low[u] = dfn[i];
38. }
39. if(dfn[u] == low[u]) {
40. ++cnt;
41. while(1) {
42. int i = sta[top--];
43. vis[i] = false;
44. belong[i] = cnt;
45. if(i == u) break;
46. }
47. }
48. }
49. void Tarjan() {//注意节点是从几开始存的
50. clr(dfn, 0);
51. clr(vis, 0);
52. top = cnt = ind = 0;
53. for(int i=1; i<=n; i++)//这里节点从1开始存，若从0开始存要改这里
54. if(!dfn[i]) TarjanSolve(i);
55. }
56. struct EDGE {
57. int u, v, w;
58. bool flag;
59. EDGE(){}
60. EDGE(int x, int y, int z, bool f):u(x), v(y), w(z), flag(f){}
61. }   edge[E];
62. int edgel;
63. bool visitable[N];
64. void dfs(int x) {
65. visitable[x] = true;
66. for(int i=h[x]; ~i; i=nxt[i]) {
67. if(!visitable[v[i]]) {
68. dfs(v[i]);
69. }
70. }
71. }
72. int indegree[N];
73. //链表
74. int lh[N], lel, lv[E], lnxt[E];
75. void initLink() {
76. clr(lh, -1); lel = 0;
77. }
78. void addLink(int x, int y) {
79. lv[lel] = y; lnxt[lel] = lh[x]; lh[x] = lel++;
80. }
81. int dis[N];
82. bool tag[N];
83. int main() {
84. int r, p, s;
85. while(~scanf("%d%d%d%d", &n, &r, &p, &s)) {
86. clr(visitable, 0);
87. initEdge();
88. edgel = 0;
89. int x, y, z;
90. for(int i=0; i<r; i++) {
91. scanf("%d%d%d", &x, &y, &z);
92. addEdge(x, y, z);
93. addEdge(y, x, z);
94. edge[edgel++] = EDGE(x, y, z, false);
95. }
96. for(int i=0; i<p; i++) {
97. scanf("%d%d%d", &x, &y, &z);
98. addEdge(x, y, z);
99. edge[edgel++] = EDGE(x, y, z, true);
100. }
101. Tarjan();
102. dfs(s);
103. initEdge();
104. initLink();
105. clr(indegree, 0);
106. for(int i=0; i<edgel; i++) {
107. if(visitable[edge[i].u] && visitable[edge[i].v]) {
108. addEdge(edge[i].u, edge[i].v, edge[i].w);
109. if(edge[i].flag) {
110. ++ indegree[belong[edge[i].v]];
111. addLink(belong[edge[i].v], edge[i].v);
112. } else {
113. addEdge(edge[i].v, edge[i].u, edge[i].w);
114. }
115. }
116. }
117. stack<int> zeroDegree;
118. priority\_queue<pair<int,int> > que;
119. clr(vis, false);
120. clr(tag, false);
121. clr(dis, 0x3f);
122. dis[s] = 0;
123. que.push(make\_pair(0, s));
124. while(!que.empty() || !zeroDegree.empty()) {
125. if(que.empty()) {
126. int x = zeroDegree.top(); zeroDegree.pop();
127. for(int i=lh[x]; ~i; i=lnxt[i]) {
128. int y = lv[i];
129. if(!vis[y]) {
130. vis[y] = true;
131. que.push(make\_pair(-dis[y], y));
132. }
133. }
134. } else {
135. int x = que.top().second; que.pop();
136. if(tag[x]) continue;
137. tag[x]  = true;
138. for(int i=h[x]; ~i; i=nxt[i]) {
139. int y = v[i];
140. if(!tag[y] && dis[y] > dis[x] + w[i]) {
141. dis[y] = dis[x] + w[i];
142. if(belong[x] == belong[y]) {
143. que.push(make\_pair(-dis[y], y));
144. }
145. }
146. if(belong[x] != belong[y]) {
147. -- indegree[belong[y]];
148. if(indegree[belong[y]] == 0) {
149. zeroDegree.push(belong[y]);
150. }
151. }
152. }
153. }
154. }
155. for(int i=1; i<=n; i++) {
156. if(visitable[i]) {
157. printf("%d\n", dis[i]);
158. } else {
159. puts("NO PATH");
160. }
161. }
162. }
163. return 0;
164. }

## 3、金属采集(树形动态规划)

问题描述

人类在火星上发现了一种新的金属！这些金属分布在一些奇怪的地方，不妨叫它节点好了。一些节点之间有道路相连，所有的节点和道路形成了一棵树。一共有 n 个节点，这些节点被编号为 1~n 。人类将 k 个机器人送上了火星，目的是采集这些金属。这些机器人都被送到了一个指定的着落点， S 号节点。每个机器人在着落之后，必须沿着道路行走。当机器人到达一个节点时，它会采集这个节点蕴藏的所有金属矿。当机器人完成自己的任务之后，可以从任意一个节点返回地球。当然，回到地球的机器人就无法再到火星去了。我们已经提前测量出了每条道路的信息，包括它的两个端点 x 和 y，以及通过这条道路需要花费的能量 w 。我们想花费尽量少的能量采集所有节点的金属，这个任务就交给你了。

输入格式

第一行包含三个整数 n, S 和 k ，分别代表节点个数、着落点编号，和机器人个数。

接下来一共 n-1 行，每行描述一条道路。一行含有三个整数 x, y 和 w ，代表在 x 号节点和 y 号节点之间有一条道路，通过需要花费 w 个单位的能量。所有道路都可以双向通行。

输出格式

输出一个整数，代表采集所有节点的金属所需要的最少能量。

样例输入

6 1 3  
1 2 1  
2 3 1  
2 4 1000  
2 5 1000  
1 6 1000

样例输出

3004

样例说明

所有机器人在 1 号节点着陆。

第一个机器人的行走路径为 1->6 ，在 6 号节点返回地球，花费能量为1000。

第二个机器人的行走路径为 1->2->3->2->4 ，在 4 号节点返回地球，花费能量为1003。

第一个机器人的行走路径为 1->2->5 ，在 5 号节点返回地球，花费能量为1001。

数据规模与约定

本题有10个测试点。

对于测试点 1~2 ， n <= 10 ， k <= 5 。

对于测试点 3 ， n <= 100000 ， k = 1 。

对于测试点 4 ， n <= 1000 ， k = 2 。

对于测试点 5~6 ， n <= 1000 ， k <= 10 。

对于测试点 7~10 ， n <= 100000 ， k <= 10 。

道路的能量 w 均为不超过 1000 的正整数。

本题的C++参考代码如下：

1. #include <iostream>
2. #include <cstdio>
3. using namespace std;
4. const int MAXN=100000+10,oo=100000000,MAXK=10+1;
5. typedef long long LL;
6. int N,S,K,fa[MAXN];
7. int g[MAXN],num[MAXN\*2],next[MAXN\*2],cost[MAXN\*2],tot=1;
8. LL f[MAXN][MAXK],sum;
9. inline void read(int &x)
10. {
11. char ch;
12. while (ch=getchar(),ch>'9' || ch<'0') ; x=ch-48;
13. while (ch=getchar(),ch<='9' && ch>='0') x=x\*10+ch-48;
14. }
15. inline void addedge(int a,int b,int c) { ++tot; num[tot]=b; next[tot]=g[a]; g[a]=tot; cost[tot]=c; }
16. void dfs(int x)
17. {
18. for (int i=g[x];i;i=next[i])
19. if (num[i]!=fa[x])
20. {
21. fa[num[i]]=x;
22. dfs(num[i]);
24. for (int a=K;a;--a)
25. for (int b=1;b<=a;++b)
26. f[x][a]=max(f[x][a],f[x][a-b]+f[num[i]][b]+(LL)(-b+2)\*cost[i]);
27. }
28. }
29. int main()
30. {
31. read(N); read(S); read(K);
32. for (int i=1;i<N;++i)
33. {
34. int x,y,z;
35. read(x); read(y); read(z); sum+=z;
36. addedge(x,y,z); addedge(y,x,z);
37. }
39. sum=sum+sum;
40. dfs(S);
42. LL ans=oo; ans=ans\*ans;
43. for (int i=0;i<=K;++i) ans=min(ans,sum-f[S][i]);
45. cout << ans << endl;
47. return 0;
48. }

## 4、矩阵翻转(枚举 贪心)

问题描述

Ciel有一个N\*N的矩阵，每个格子里都有一个整数。

N是一个奇数，设X = (N+1)/2。Ciel每次都可以做这样的一次操作：他从矩阵选出一个X\*X的子矩阵，并将这个子矩阵中的所有整数都乘以-1。

现在问你经过一些操作之后，矩阵中所有数的和最大可以为多少。

输入格式

第一行为一个正整数N。

接下来N行每行有N个整数，表示初始矩阵中的数字。每个数的绝对值不超过1000。

输出格式

输出一个整数，表示操作后矩阵中所有数之和的最大值。

样例输入

3  
-1 -1 1  
-1 1 -1  
1 -1 -1

样例输出

9

数据规模与约定

1 <= N <= 33，且N为奇数。

本题的C++参考代码如下：

1. #include<cstdio>
2. #include<algorithm>
3. #include<cstring>
4. using namespace std;
5. const int N=50;
6. int n,m,a[N][N],f[N][N];
7. int getval(int j)
8. {
9. int ans=0;
10. for(int i=0;i<m;i++)
11. ans+=abs(a[i][j]+f[i][m]\*a[i][j+m+1]+f[m][j]\*a[i+m+1][j]+f[i][m]\*f[m][j+m+1]\*a[i+m+1][j+m+1]);
12. return ans;
13. }
14. int solve(int mask)
15. {
16. //printf("mask=%d ",mask);
17. int ans=0;
18. for(int i=0;i<=m;i++)
19. {
20. f[i][m]=(mask&(1<<i))?-1:1;
21. ans+=a[i][m]\*f[i][m];
22. }
23. for(int i=m+1;i<n;i++)
24. {
25. f[i][m]=f[i-m-1][m]\*f[m][m];
26. ans+=a[i][m]\*f[i][m];
27. }
28. for(int j=0;j<m;j++)
29. {
30. int ta;
31. f[m][j]=-1;
32. f[m][j+m+1]=f[m][j]\*f[m][m];
33. ta=getval(j)+f[m][j]\*a[m][j]+f[m][j+m+1]\*a[m][j+m+1];
34. f[m][j]=1;
35. f[m][j+m+1]=f[m][j]\*f[m][m];
36. ta=max(ta,getval(j)+f[m][j]\*a[m][j]+f[m][j+m+1]\*a[m][j+m+1]);
37. ans+=ta;
38. }
39. //printf("ans=%d\n",ans);
40. return ans;
41. }
42. int main()
43. {
44. //freopen("data.in","r",stdin);
45. while(scanf("%d",&n)!=EOF)
46. {
47. int ans=-10000000;
48. m=n>>1;
49. for(int i=0;i<n;i++)
50. for(int j=0;j<n;j++)
51. scanf("%d",&a[i][j]);
52. for(int mask=0,msize=1<<(m+1);mask<msize;mask++)
53. ans=max(ans,solve(mask));
54. printf("%d\n",ans);
55. }
56. return 0;
57. }

## 5、两条直线(排序)

问题描述

给定平面上n个点。

求两条直线，这两条直线互相垂直，而且它们与x轴的夹角为45度，并且n个点中离这两条直线的曼哈顿距离的最大值最小。

两点之间的曼哈顿距离定义为横坐标的差的绝对值与纵坐标的差的绝对值之和，一个点到两条直线的曼哈顿距离是指该点到两条直线上的所有点的曼哈顿距离中的最小值。

输入格式

第一行包含一个数n。

接下来n行，每行包含两个整数，表示n个点的坐标（横纵坐标的绝对值小于109）。

输出格式

输出一个值，表示最小的最大曼哈顿距离的值，保留一位小数。

样例输入

4  
1 0  
0 1  
2 1  
1 2

样例输出

1.0

数据规模与约定

对于30%的数据，n<=100。

对于另外30%的数据，坐标范的绝对值小于100。

对于100%的数据，n<=105。

本题的C++参考代码如下：

1. #include<iostream>
2. #include<algorithm>
3. #include<cstdio>
4. #include<cmath>
5. using namespace std;
6. const int N=100000;
7. struct P{int x,y;};
8. bool cmp(P a,P b){
9. if(a.x==b.x)return a.y<b.y;
10. return a.x<b.x;
11. }
12. P d[N+5];
13. struct F{int max,min;};
14. F fl[N+5],fr[N+5];
15. inline double Max(double a,double b){return a>b?a:b;}
16. inline double Min(double a,double b){return a>b?b:a;}
17. bool check(double m,int n){
18. m\*=2;
19. int i,j=0;
20. for(i=0;i<n;i++){
21. while(j<n&&d[j].x-d[i].x<=m)j++;
22. double MAX=-1e10;
23. double MIN=1e10;
24. if(j!=n){
25. MAX=Max(MAX,fr[j].max);
26. MIN=Min(MIN,fr[j].min);
27. }
28. if(i-1>=0){
29. MAX=Max(MAX,fl[i-1].max);
30. MIN=Min(MIN,fl[i-1].min);
31. }
32. //   cout<<i<<" "<<j<<" "<<MAX<<" "<<MIN<<endl;
33. if(MAX-MIN<=m)return true;
34. }
35. return false;
36. }
37. void init(int n){
38. int i;
39. fl[0].min=fl[0].max=d[0].y;
40. for(i=1;i<n;i++){
41. fl[i].max=Max(fl[i-1].max,d[i].y);
42. fl[i].min=Min(fl[i-1].min,d[i].y);
43. }
44. fr[n-1].min=fr[n-1].max=d[n-1].y;
45. for(i=n-2;i>=0;i--){
46. fr[i].max=Max(fr[i+1].max,d[i].y);
47. fr[i].min=Min(fr[i+1].min,d[i].y);
48. }
49. }
50. int main(){
51. int i,n;
52. cin>>n;
53. for(i=0;i<n;i++){
54. int x,y;
55. scanf("%d%d",&x,&y);
56. d[i].x=x+y;
57. d[i].y=x-y;
58. }
59. sort(d,d+n,cmp);
60. init(n);
61. double l=0.0;
62. double r=1000000000;
63. while(r-l>=0.01){
64. double m=(l+r)/2;
65. //  cout<<m<<endl;
66. if(check(m,n))r=m;
67. else l=m;
68. }
69. printf("%.1f\n",r);
70. return 0;
71. }

# 五、历届试题

## 1、核桃的数量(最小公倍数)

问题描述

小张是软件项目经理，他带领3个开发组。工期紧，今天都在加班呢。为鼓舞士气，小张打算给每个组发一袋核桃（据传言能补脑）。他的要求是：

1. 各组的核桃数量必须相同

2. 各组内必须能平分核桃（当然是不能打碎的）

3. 尽量提供满足1,2条件的最小数量（节约闹革命嘛）

输入格式

输入包含三个正整数a, b, c，表示每个组正在加班的人数，用空格分开（a,b,c<30）

输出格式

输出一个正整数，表示每袋核桃的数量。

样例输入1

2 4 5

样例输出1

20

样例输入2

3 1 1

样例输出2

3

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. //最小公倍数
4. int LCM(int num1,int num2,int num3)
5. {
6. int value=num1;
7. while(value%num1!=0||value%num2!=0||value%num3!=0)
8. {
9. value+=num1;
10. }
11. return value;
12. }
13. int main()
14. {
15. int num1,num2,num3;
16. scanf("%d%d%d",&num1,&num2,&num3);
17. printf("%d\n",LCM(num1,num2,num3));
18. return 0;
19. }

## 2、打印十字图(文字图形)

问题描述

小明为某机构设计了一个十字型的徽标（并非红十字会啊），如下所示：

..$$$$$$$$$$$$$..  
..$...........$..  
$$$.$$$$$$$$$.$$$  
$...$.......$...$  
$.$$$.$$$$$.$$$.$  
$.$...$...$...$.$  
$.$.$$$.$.$$$.$.$  
$.$.$...$...$.$.$  
$.$.$.$$$$$.$.$.$  
$.$.$...$...$.$.$  
$.$.$$$.$.$$$.$.$  
$.$...$...$...$.$  
$.$$$.$$$$$.$$$.$  
$...$.......$...$  
$$$.$$$$$$$$$.$$$  
..$...........$..  
..$$$$$$$$$$$$$..

对方同时也需要在电脑dos窗口中以字符的形式输出该标志，并能任意控制层数。

输入格式

一个正整数 n (n<30) 表示要求打印图形的层数。

输出格式

对应包围层数的该标志。

样例输入1

1

样例输出1

..$$$$$..  
..$...$..  
$$$.$.$$$  
$...$...$  
$.$$$$$.$  
$...$...$  
$$$.$.$$$  
..$...$..  
..$$$$$..

样例输入2

3

样例输出2

..$$$$$$$$$$$$$..  
..$...........$..  
$$$.$$$$$$$$$.$$$  
$...$.......$...$  
$.$$$.$$$$$.$$$.$  
$.$...$...$...$.$  
$.$.$$$.$.$$$.$.$  
$.$.$...$...$.$.$  
$.$.$.$$$$$.$.$.$  
$.$.$...$...$.$.$  
$.$.$$$.$.$$$.$.$  
$.$...$...$...$.$  
$.$$$.$$$$$.$$$.$  
$...$.......$...$  
$$$.$$$$$$$$$.$$$  
..$...........$..  
..$$$$$$$$$$$$$..

提示

请仔细观察样例，尤其要注意句点的数量和输出位置。

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <malloc.h>
3. #include <string.h>
4. int main()
5. {
6. int n,w,h,l,i,j,x,y,m;
7. char \*arry;
8. scanf("%d",&n);
9. w=h=5+n\*4;
10. arry=(char \*)malloc(w\*h);
11. memset(arry,'.',w\*h);
12. /\*
13. for(i=n\*2;i<w-n\*2;i++)//画基础图形
14. {
15. arry[i+((n+1)\*2)\*w]='$';
16. arry[i\*w+(n+1)\*2]='$';
17. }\*/
18. for(m=0;m<=n;m++)//画没一层
19. {
20. for(i=(m+1)\*2;i<w-(m+1)\*2;i++)//四边
21. {
22. x=m\*2;
23. y=i;
24. arry[x+y\*w]='$';
25. x=w-m\*2-1;
26. y=i;
27. arry[x+y\*w]='$';
28. x=i;
29. y=m\*2;
30. arry[x+y\*w]='$';
31. x=i;
32. y=w-m\*2-1;
33. arry[x+y\*w]='$';
34. }
35. for(i=m\*2;i<=(m+1)\*2;i++)//角
36. {
37. x=i;
38. y=(m+1)\*2;
39. arry[x+y\*w]='$';
40. x=(m+1)\*2;
41. y=i;
42. arry[x+y\*w]='$';
44. x=w-i-1;
45. y=(m+1)\*2;
46. arry[x+y\*w]='$';
47. x=w-(m+1)\*2-1;
48. y=i;
49. arry[x+y\*w]='$';
51. x=i;
52. y=h-(m+1)\*2-1;
53. arry[x+y\*w]='$';
54. x=(m+1)\*2;
55. y=h-i-1;
56. arry[x+y\*w]='$';
58. x=w-i-1;
59. y=h-(m+1)\*2-1;
60. arry[x+y\*w]='$';
61. x=w-(m+1)\*2-1;
62. y=h-i-1;
63. arry[x+y\*w]='$';
64. }
65. }
66. for(i=0;i<h;i++)
67. {
68. for(j=0;j<w;j++)
69. {
70. printf("%c",arry[i\*w+j]);
71. }
72. printf("\n");
73. }
74. return 0;
75. }

## 3、带分数(搜索)

问题描述

100 可以表示为带分数的形式：100 = 3 + 69258 / 714。

还可以表示为：100 = 82 + 3546 / 197。

注意特征：带分数中，数字1~9分别出现且只出现一次（不包含0）。

类似这样的带分数，100 有 11 种表示法。

输入格式

从标准输入读入一个正整数N (N<1000\*1000)

输出格式

程序输出该数字用数码1~9不重复不遗漏地组成带分数表示的全部种数。

注意：不要求输出每个表示，只统计有多少表示法！

样例输入1

100

样例输出1

11

样例输入2

105

样例输出2

6

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. typedef struct Interval
4. {
5. int pre;
6. int rear;
7. int satisfy;
8. }Interval;
9. Interval interval[7][5];
10. int count=0;
11. //初始化
12. void Init()
13. {
14. int i,j;
15. int value;
16. for(i=1;i<7;i++)
17. {
18. value=i;
19. for(j=1;j<5;j++)
20. {
21. interval[i][j].pre=value++;
22. interval[i][j].rear=value;
23. }
24. }
25. }
26. //数组初始化为0
27. void InitZero(int \*sign)
28. {
29. int i;
30. sign[0]=1;
31. for(i=1;i<10;i++)
32. {
33. sign[i]=0;
34. }
35. }
36. //将一个数的各个位上拆分，并在相应的位上赋值1
37. int Split(int \*sign,int value)
38. {
39. int index;
40. while(value)
41. {
42. index=value%10;
43. if(sign[index]==0) sign[index]=1;
44. else return 1;
45. value/=10;
46. }
47. return 0;
48. }
49. //计算一个数的位数
50. int CountBit(int value)
51. {
52. int n=0;
53. while(value)
54. {
55. n++;
56. value/=10;
57. }
58. return n;
59. }
60. //将一个整型数组转换成一个整数
61. int CreateInteger(int \*data,int n)
62. {
63. int i;
64. int value=0;
65. for(i=0;i<n;i++)
66. {
67. value=value\*10+data[i];
68. }
69. return value;
70. }
71. //检查是否每个数都用到
72. int Check(int \*sign)
73. {
74. int i;
75. for(i=1;i<10;i++)
76. {
77. if(sign[i]==0) return 0;
78. }
79. return 1;
80. }
81. //复制
82. void Copy(int \*sign,int \*temp\_sign)
83. {
84. int i;
85. for(i=0;i<10;i++)
86. {
87. temp\_sign[i]=sign[i];
88. }
89. }
90. //创建一个n位数的整数
91. void CreateNBitNumber(int \*sign,int \*data,int n,int m,int value,int value3)
92. {
93. if(n==m)
94. {
95. int value1=CreateInteger(data,n);
96. int value2=value1\*value;
97. int temp\_sign[10];
98. Copy(sign,temp\_sign);
99. if(!Split(temp\_sign,value2) && Check(temp\_sign))
100. {
101. count++;
102. }
103. }
104. else
105. {
106. int i;
107. for(i=1;i<10;i++)
108. {
109. if(sign[i]==0)
110. {
111. sign[i]=1;
112. data[m]=i;
113. CreateNBitNumber(sign,data,n,m+1,value,value3);
114. sign[i]=0;
115. }
116. }
117. }
118. }
119. //求出解
120. void Create(int value)
121. {
122. int i,j;
123. int sign[10];
124. int result;
125. int result\_n;
126. int n;
127. for(i=3;i<value;i++)
128. {
129. InitZero(sign);
130. if(Split(sign,i)) continue;
131. result=value-i;
132. result\_n=CountBit(result);
133. n=CountBit(i);
134. for(j=1;j<5;j++)
135. {
136. if( ((interval[result\_n][j].pre+j)==(9-n)) || ((interval[result\_n][j].rear+j)==(9-n)))
137. {
138. int data[5];
139. CreateNBitNumber(sign,data,j,0,result,i);
140. }
141. }
142. }
143. }
144. int main()
145. {
146. int value;
147. scanf("%d",&value);
148. Init();
149. Create(value);
150. printf("%d\n",count);
151. return 0;
152. }

## 4、剪格子(搜索)

问题描述

如下图所示，3 x 3 的格子中填写了一些整数。

+--\*--+--+  
|10\* 1|52|  
+--\*\*\*\*--+  
|20|30\* 1|  
\*\*\*\*\*\*\*--+  
| 1| 2| 3|  
+--+--+--+

我们沿着图中的星号线剪开，得到两个部分，每个部分的数字和都是60。

本题的要求就是请你编程判定：对给定的m x n 的格子中的整数，是否可以分割为两个部分，使得这两个区域的数字和相等。

如果存在多种解答，请输出包含左上角格子的那个区域包含的格子的最小数目。

如果无法分割，则输出 0。

输入格式

程序先读入两个整数 m n 用空格分割 (m,n<10)。

表示表格的宽度和高度。

接下来是n行，每行m个正整数，用空格分开。每个整数不大于10000。

输出格式

输出一个整数，表示在所有解中，包含左上角的分割区可能包含的最小的格子数目。

样例输入1

3 3  
10 1 52  
20 30 1  
1 2 3

样例输出1

3

样例输入2

4 3  
1 1 1 1  
1 30 80 2  
1 1 1 100

样例输出2

10

本题的C参考代码如下：

1. #include <stdio.h>
2. #define N 10
3. int num[N][N];
4. int tag[N][N] = {0};
5. int m, n;
6. int r = 100;
7. int find(int i, int j, int t, int ntag[][N])
8. {
9. int count = 0;
10. if (i < 0 || i >= n || j < 0 || j >= m || ntag[i][j] == 1)
11. return 0;
12. ntag[i][j] = 1;
13. if (tag[i][j] != t)
14. return 0;
15. count++;
16. count += find(i - 1, j, t, ntag);
17. count += find(i + 1, j, t, ntag);
18. count += find(i, j - 1, t, ntag);
19. count += find(i, j + 1, t, ntag);
20. return count;
21. }
23. int isbad()
24. {
25. int i, j, k = 0,ge2;
26. int t = tag[0][0];
27. int ntag1[N][N] = {0};
28. int ntag2[N][N] = {0};
29. int ge1 = find(0, 0, t, ntag1);
30. for (i = 0; i < n; i++)
31. {
32. for (j = 0; j < m; j++)
33. {
34. if (tag[i][j] != t)
35. {
36. k = 1;
37. break;
38. }
39. }
40. if (k == 1)
41. break;
42. }
44. if (i == n && j == m)
45. return 0;
46. ge2 = find(i, j, tag[i][j], ntag2);
47. return ge1 + ge2 != m \* n;
48. }
49. int bad(int i, int j)
50. {
51. int b;
52. if (i < 0 || i >= n || j < 0 || j >= m || tag[i][j] == 1)
53. return 1;
55. tag[i][j] = 1;
56. b = isbad();
57. tag[i][j] = 0;
58. return b;
59. }
60. void go(int i, int j, int k, int count)
61. {
63. if (bad(i, j) || count < num[i][j])
64. return;
65. k++;
67. if (count == num[i][j])
68. {
69. if (r > k)
70. r = k;
71. return;
72. }

75. tag[i][j] = 1;
76. count -= num[i][j];
77. go(i - 1, j, k, count);
78. go(i + 1, j, k, count);
79. go(i, j - 1, k, count);
80. go(i, j + 1, k, count);
81. tag[i][j] = 0;
82. }
83. int main()
84. {
86. int i, j;
87. int half = 0;
88. scanf("%d %d", &m, &n);
89. for (i = 0; i < n; i++)
90. for (j = 0; j < m; j++)
91. {
92. scanf("%d", &num[i][j]);
94. half += num[i][j];
95. }

98. if (half % 2 == 0 && half >= num[0][0] \* 2)
99. {
101. half /= 2;
102. go(0, 0, 0, half);
103. }
105. if (r == 100)
106. r = 0;
108. printf("%d", r);
109. return 0;
110. }

## 5、错误票据(排序)

问题描述

某涉密单位下发了某种票据，并要在年终全部收回。

每张票据有唯一的ID号。全年所有票据的ID号是连续的，但ID的开始数码是随机选定的。

因为工作人员疏忽，在录入ID号的时候发生了一处错误，造成了某个ID断号，另外一个ID重号。

你的任务是通过编程，找出断号的ID和重号的ID。

假设断号不可能发生在最大和最小号。

输入格式

要求程序首先输入一个整数N(N<100)表示后面数据行数。

接着读入N行数据。

每行数据长度不等，是用空格分开的若干个（不大于100个）正整数（不大于100000），请注意行内和行末可能有多余的空格，你的程序需要能处理这些空格。

每个整数代表一个ID号。

输出格式

要求程序输出1行，含两个整数m n，用空格分隔。

其中，m表示断号ID，n表示重号ID

样例输入1

2  
5 6 8 11 9   
10 12 9

样例输出1

7 9

样例输入2

6  
164 178 108 109 180 155 141 159 104 182 179 118 137 184 115 124 125 129 168 196  
172 189 127 107 112 192 103 131 133 169 158   
128 102 110 148 139 157 140 195 197  
185 152 135 106 123 173 122 136 174 191 145 116 151 143 175 120 161 134 162 190  
149 138 142 146 199 126 165 156 153 193 144 166 170 121 171 132 101 194 187 188  
113 130 176 154 177 120 117 150 114 183 186 181 100 163 160 167 147 198 111 119

样例输出2

105 120

本题的C参考代码如下：

1. #include <stdio.h>
2. int main()
3. {
4. int a[10001]={0};
5. long m,min=100000,max=0,i,n;
6. char c;
7. scanf("%d",&n);
8. for(i=0;i<n;i++)
9. while(1)
10. {
11. scanf("%ld",&m);
12. if(m>max) max=m;
13. if(m<min) min=m;
14. a[m]++;
15. c=getchar();
16. if(c!=' ') break;
17. }
18. for(i=min;i<=max;i++)
19. {
20. if(a[i]==0) printf("%ld ",i);
21. if(a[i]==2) m=i;
22. }
23. printf("%ld",m);
25. return 0;
26. }

## 6、翻硬币(贪心)

问题描述

小明正在玩一个“翻硬币”的游戏。

桌上放着排成一排的若干硬币。我们用 \* 表示正面，用 o 表示反面（是小写字母，不是零）。

比如，可能情形是：\*\*oo\*\*\*oooo

如果同时翻转左边的两个硬币，则变为：oooo\*\*\*oooo

现在小明的问题是：如果已知了初始状态和要达到的目标状态，每次只能同时翻转相邻的两个硬币,那么对特定的局面，最少要翻动多少次呢？

我们约定：把翻动相邻的两个硬币叫做一步操作，那么要求：

输入格式

两行等长的字符串，分别表示初始状态和要达到的目标状态。每行的长度<1000

输出格式

一个整数，表示最小操作步数。

样例输入1

\*\*\*\*\*\*\*\*\*\*  
o\*\*\*\*o\*\*\*\*

样例输出1

5

样例输入2

\*o\*\*o\*\*\*o\*\*\*  
\*o\*\*\*o\*\*o\*\*\*

样例输出2

1

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. int turn(char a[], char b[])
4. {
5. int i;
6. int n = 0;
7. for(i = 0;a[i]!='\0';i++)
8. {
9. if(a[i] == b[i])
10. {
11. continue;
12. }
13. else
14. {
15. b[i+1]=(b[i+1]=='\*'?'o':'\*');
16. n++;
17. }
18. }
19. return n;
20. }
21. int main(void)
22. {
23. char a[1000];
24. char b[1000];
25. gets(a);
26. gets(b);
27. printf("%d\n", turn(a, b));
28. return 0;
29. }

## 7、连号区间数(并查集)

问题描述

小明这些天一直在思考这样一个奇怪而有趣的问题：

在1~N的某个全排列中有多少个连号区间呢？这里所说的连号区间的定义是：

如果区间[L, R] 里的所有元素（即此排列的第L个到第R个元素）递增排序后能得到一个长度为R-L+1的“连续”数列，则称这个区间连号区间。

当N很小的时候，小明可以很快地算出答案，但是当N变大的时候，问题就不是那么简单了，现在小明需要你的帮助。

输入格式

第一行是一个正整数N (1 <= N <= 50000), 表示全排列的规模。

第二行是N个不同的数字Pi(1 <= Pi <= N)， 表示这N个数字的某一全排列。

输出格式

输出一个整数，表示不同连号区间的数目。

样例输入1

4  
3 2 4 1

样例输出1

7

样例输入2

5  
3 4 2 5 1

样例输出2

9

本题的C参考代码如下：

1. #include<stdio.h>
2. int main()
3. {
4. int s[50005],a,i,min,max,count=0,j;
5. scanf("%d",&a);
6. for( i = 0; i < a; i++) {
7. scanf("%d",&s[i]);
8. }
9. for( i = 0; i <a;i++ ){
10. min=s[i];
11. max=s[i];
12. for( j = i; j <a; j++) {
13. if(min>s[j]){min =s[j];}
14. if(max<s[j]){max =s[j];}
15. if((max-min)==(j-i)){
16. count++;
17. }
18. }
19. }
20. printf("%d",count);
21. return 0;
22. }

## 8、买不到的数目(数论 动态规划)

问题描述

小明开了一家糖果店。他别出心裁：把水果糖包成4颗一包和7颗一包的两种。糖果不能拆包卖。

小朋友来买糖的时候，他就用这两种包装来组合。当然有些糖果数目是无法组合出来的，比如要买 10 颗糖。

你可以用计算机测试一下，在这种包装情况下，最大不能买到的数量是17。大于17的任何数字都可以用4和7组合出来。

本题的要求就是在已知两个包装的数量时，求最大不能组合出的数字。

输入格式

两个正整数，表示每种包装中糖的颗数(都不多于1000)

输出格式

一个正整数，表示最大不能买到的糖数

样例输入1

4 7

样例输出1

17

样例输入2

3 5

样例输出2

7

本题的C参考代码如下：

1. #include<stdio.h>
2. #define MAXSIZE 1000
3. int main() {
4. int flag[MAXSIZE] = {0};
5. int maxunuse[MAXSIZE] = {0};
6. int num1, num2, minNum, maxNum, temp, i, count = 0;
7. scanf("%d%d", &num1, &num2);
8. minNum = ((num1 < num2) ? num1 : num2);
9. maxNum = num1 + num2 - minNum;
10. for(i = 1; ;i++){
11. temp = i \* maxNum % minNum;
12. if((temp) && flag[temp] == 0){
13. flag[temp] = 1;
14. maxunuse[temp] = i \* maxNum -minNum;
15. count++;
16. if(count == minNum - 1){
17. break;
18. }
19. }
20. }
21. printf("%d\n", maxunuse[temp]);
22. return 0;
23. }

## 9、大臣的旅费(深度优先遍历)

问题描述

很久以前，T王国空前繁荣。为了更好地管理国家，王国修建了大量的快速路，用于连接首都和王国内的各大城市。

为节省经费，T国的大臣们经过思考，制定了一套优秀的修建方案，使得任何一个大城市都能从首都直接或者通过其他大城市间接到达。同时，如果不重复经过大城市，从首都到达每个大城市的方案都是唯一的。

J是T国重要大臣，他巡查于各大城市之间，体察民情。所以，从一个城市马不停蹄地到另一个城市成了J最常做的事情。他有一个钱袋，用于存放往来城市间的路费。

聪明的J发现，如果不在某个城市停下来修整，在连续行进过程中，他所花的路费与他已走过的距离有关，在走第x千米到第x+1千米这一千米中（x是整数），他花费的路费是x+10这么多。也就是说走1千米花费11，走2千米要花费23。

J大臣想知道：他从某一个城市出发，中间不休息，到达另一个城市，所有可能花费的路费中最多是多少呢？

输入格式

输入的第一行包含一个整数n，表示包括首都在内的T王国的城市数

城市从1开始依次编号，1号城市为首都。

接下来n-1行，描述T国的高速路（T国的高速路一定是n-1条）

每行三个整数Pi, Qi, Di，表示城市Pi和城市Qi之间有一条高速路，长度为Di千米。

输出格式

输出一个整数，表示大臣J最多花费的路费是多少。

样例输入1

5  
1 2 2  
1 3 1  
2 4 5  
2 5 4

样例输出1

135

输出格式

大臣J从城市4到城市5要花费135的路费。

本题的C参考代码如下：

1. #include <stdio.h>
2. #include <string.h>
3. #include <math.h>
4. #include <stdlib.h>
5. struct node;
6. typedef struct node Node;
7. typedef Node \*PtrToNode;
8. typedef PtrToNode List;
9. typedef PtrToNode Position;
10. struct node
11. {
12. int n;
13. int val;
14. Position next;
15. };
16. int count=0;
17. int max=0;
18. int x;
19. int \*visit;     //是否已遍历
20. Position Last(List l);                       //找出最后项
21. void Insert(int x,int q,List l,Position p);  //在p后插入含x的项
22. void Dfs(int a,List l[]);                    //深度优先搜索
23. int Num(List l);
24. int main(void)
25. {
26. int n,u,v,q,a,b;
27. int i,j,k;
28. Node \*head;
29. List \*l,tmp;
30. Position p;
31. fscanf(stdin,"%d",&n);
32. head=(Node \*)malloc(sizeof(Node)\*(n+1));
33. l=(List \*)malloc(sizeof(List)\*(n+1));
34. visit=(int \*)malloc(sizeof(int)\*(n+1));
35. for(i=0;i<=n;i++)  //初始化表头及链表
36. {
37. head[i].next=NULL;
38. l[i]=&head[i];
39. }
40. for(i=1;i<=n-1;i++)  //建立无向图
41. {
42. fscanf(stdin,"%d%d%d",&u,&v,&q);
43. Insert(v,q,l[u],Last(l[u]));
44. Insert(u,q,l[v],Last(l[v]));
45. }
46. for(j=1;j<=n;j++)
47. visit[j]=0;
48. Dfs(1,l);  //第一次遍历，找到点a，用全局变量x保存
49. for(j=1;j<=n;j++)
50. visit[j]=0;
51. count=0;
52. max=0;
53. Dfs(x,l);  //第二次遍历，找到点b，用全局变量x保存，此时max为最大距离
54. printf("%d",max\*10+(max+1)\*max/2);
55. return 0;
56. }
57. Position Last(List l)
58. {
59. Position p;
60. for(p=l;p->next!=NULL;p=p->next);
61. return p;
62. }
63. void Insert(int x,int q,List l,Position p)
64. {
65. Position tmp;
66. tmp=(Position) malloc(sizeof(Node));
67. tmp->n=x;
68. tmp->val=q;
69. tmp->next=p->next;
70. p->next=tmp;
71. }
72. void Dfs(int a,List l[])
73. {
74. Position p;
75. visit[a]=1;
76. for(p=l[a]->next;p!=NULL;p=p->next)
77. if(!(visit[p->n]))
78. {
79. count+=p->val;
80. if(count>max)
81. {
82. max=count;
83. x=p->n;
84. }
85. Dfs(p->n,l);
86. count-=p->val;
87. }
88. }
89. int Num(List l)
90. {
91. int n=0;
92. Position p;
93. for(p=l->next;p!=NULL;p=p->next)
94. n++;
95. return n;
96. }

## 10、幸运数(堆)

问题描述

幸运数是波兰数学家乌拉姆命名的。它采用与生成素数类似的“筛法”生成

。

首先从1开始写出自然数1,2,3,4,5,6,....

1 就是第一个幸运数。

我们从2这个数开始。把所有序号能被2整除的项删除，变为：

1 \_ 3 \_ 5 \_ 7 \_ 9 ....

把它们缩紧，重新记序，为：

1 3 5 7 9 .... 。这时，3为第2个幸运数，然后把所有能被3整除的序号位置的数删去。注意，是序号位置，不是那个数本身能否被3整除!! 删除的应该是5，11, 17, ...

此时7为第3个幸运数，然后再删去序号位置能被7整除的(19,39,...)

最后剩下的序列类似：

1, 3, 7, 9, 13, 15, 21, 25, 31, 33, 37, 43, 49, 51, 63, 67, 69, 73, 75, 79, ...

输入格式

输入两个正整数m n, 用空格分开 (m < n < 1000\*1000)

输出格式

程序输出 位于m和n之间的幸运数的个数（不包含m和n）。

样例输入1

1 20

样例输出1

5

样例输入2

30 69

样例输出2

8

本题的C参考代码如下：

1. #include<stdio.h>
2. #define MAXN 1000010
3. int flag[MAXN];
4. int m,n,a[MAXN],s[MAXN],size=0;
5. int fa(int k)
6. {
7. if(flag[k])
8. return a[k];
9. return fa(k-1);
10. }
11. int main()
12. {
13. int i, p, k, j;
14. scanf("%d%d",&m,&n);
15. for(i=1;i<=n;i+=2)
16. {
17. s[++size]=i;
18. flag[i]=1;
19. a[i]=size;
20. }
21. for(i=2;i<=size;i++)
22. {
23. int Mod=s[i],d=s[i]-1;
24. if(Mod>size)
25. break;
26. for(p=1,j=Mod;j<=size;j+=Mod,p++)
27. {
28. flag[s[j]]=0;
29. for(k=1;k<Mod&&k+j<=size;k++)
30. {
31. s[++d]=s[j+k];
32. a[s[j+k]]-=p;
33. }
34. }
35. size=d;
36. }
37. printf("%d\n",fa(n-1)-fa(m));
38. return 0;
39. }

## 11、横向打印二叉树(排序二叉树)

问题描述

二叉树可以用于排序。其原理很简单：对于一个排序二叉树添加新节点时，先与根节点比较，若小则交给左子树继续处理，否则交给右子树。

当遇到空子树时，则把该节点放入那个位置。

比如，10 8 5 7 12 4 的输入顺序，应该建成二叉树如下图所示，其中.表示空白。

...|-12  
10-|  
...|-8-|  
.......|...|-7  
.......|-5-|  
...........|-4

本题目要求：根据已知的数字，建立排序二叉树，并在标准输出中横向打印该二叉树。

输入格式

输入数据为一行空格分开的N个整数。 N<100，每个数字不超过10000。

输入数据中没有重复的数字。

输出格式

输出该排序二叉树的横向表示。为了便于评卷程序比对空格的数目，请把空格用句点代替：

样例输入1

10 5 20

样例输出1

...|-20  
10-|  
...|-5

样例输入2

5 10 20 8 4 7

样例输出2

.......|-20  
..|-10-|  
..|....|-8-|  
..|........|-7  
5-|  
..|-4

本题的C参考代码如下：

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. typedef struct TNode
5. {
6. int key;
7. struct TNode \*left;
8. struct TNode \*right;
9. }TNode, \*Tree;
10. Tree insert(Tree root, Tree src)
11. {
12. if(root == NULL)
13. {
14. root = src;
15. }
16. else if(src->key > root->key)
17. {
18. root->left = insert(root->left, src);
19. }
20. else
21. {
22. root->right = insert(root->right, src);
23. }
24. return root;
25. }
26. char l[1000];
27. #define U 1
28. #define D 2
29. #define S ('.')
30. void print(Tree root, int s, int dir)
31. {
32. if(root != NULL)
33. {
34. int i;
35. char buf[10];
36. sprintf(buf, "|-%d-", root->key);
37. int len = strlen(buf);
38. for(i = 0; i < len; i++)
39. {
40. l[s + i] = S;
41. }
42. if(dir == D)
43. {
44. l[s] = '|';
45. }
46. print(root->left, s + len, U);
48. l[s] = '\0';
49. if(root->left == NULL && root->right == NULL)
50. {
51. buf[len - 1] = '\0';
52. printf("%s%s\n", l, buf);
53. }
54. else
55. {
56. printf("%s%s|\n", l, buf);
57. }
58. l[s] = S;
60. if(dir == U)
61. {
62. l[s] = '|';
63. }
64. print(root->right, s + len, D);
65. l[s] = S;
66. }
67. }
68. void printPre(Tree root, int s)
69. {
70. if(root != NULL)
71. {
72. int i;
73. char buf[10];
74. sprintf(buf, "%d-", root->key);
75. int len = strlen(buf);
76. for(i = 0; i < len; i++)
77. {
78. l[s + i] = S;
79. }
80. print(root->left, s + len, U);
82. printf("%s|\n", buf);
84. print(root->right, s + len, D);
85. }
86. }
87. int main(void)
88. {
89. int n;
90. Tree tree = NULL;
91. while(scanf("%d", &n) > 0)
92. {
93. Tree neo = malloc(sizeof(TNode));
94. neo->key = n;
95. neo->left = neo->right = NULL;
96. tree = insert(tree, neo);
97. }
98. printPre(tree, 0);
99. return 0;
100. }

## 12、危险系数(割点)

问题描述

抗日战争时期，冀中平原的地道战曾发挥重要作用。

地道的多个站点间有通道连接，形成了庞大的网络。但也有隐患，当敌人发现了某个站点后，其它站点间可能因此会失去联系。

我们来定义一个危险系数DF(x,y)：

对于两个站点x和y (x != y), 如果能找到一个站点z，当z被敌人破坏后，x和y不连通，那么我们称z为关于x,y的关键点。相应的，对于任意一对站点x和y，危险系数DF(x,y)就表示为这两点之间的关键点个数。

本题的任务是：已知网络结构，求两站点之间的危险系数。

输入格式

输入数据第一行包含2个整数n(2 <= n <= 1000), m(0 <= m <= 2000),分别代表站点数，通道数；

接下来m行，每行两个整数 u,v (1 <= u, v <= n; u != v)代表一条通道；

最后1行，两个数u,v，代表询问两点之间的危险系数DF(u, v)。

输出格式

一个整数，如果询问的两点不连通则输出-1.

样例输入

7 6  
1 3  
2 3  
3 4  
3 5  
4 5  
5 6  
1 6

样例输出

2

本题的C参考代码如下：

1. #include<stdio.h>
2. #include<stdlib.h>
3. struct Node
4. {
5. int data;
6. struct Node \*pNext;
7. };
8. struct Node tab[1001];
9. int visit[1001]={0};
10. int way[1001]={0};
11. int count[1001]={0};
12. int cnt=0;
13. void Insert(int n,int x);
14. void Init(int n);
15. void dfs(int x,int y,int n);
16. int fun(int n);
17. int main()
18. {
19. int x,y,n,m,u,v;
20. scanf("%d%d",&n,&m);
21. Init(n);
22. while(m--)
23. {
24. scanf("%d%d",&u,&v);
25. Insert(u,v);
26. Insert(v,u);
27. }
28. scanf("%d%d",&x,&y);
29. dfs(x,y,0);
30. int ret=fun(n);
31. printf("%d\n",ret);
32. return 0;
33. }
34. int fun(int n)
35. {
36. int i;
37. int ret=0;
38. for(i=1;i<=n;i++)
39. {
40. if(count[i]==cnt)
41. {
42. ret++;
43. }
44. }
45. return (ret-2);
46. }
47. void dfs(int x,int y,int n)
48. {
49. visit[x]=1;
50. way[n]=x;
51. struct Node \*p=&tab[x];
52. if(x==y)
53. {
54. int i;
55. cnt++;
56. for(i=0;i<=n;i++)
57. {
58. count[way[i]]++;
59. }
60. return ;
61. }
62. while((p=p->pNext)!=NULL)
63. {
64. if(visit[p->data]!=1)
65. {
66. dfs(p->data,y,n+1);
67. visit[p->data]=0;
68. }
69. }
70. }
71. void Init(int n)
72. {
73. int i;
74. for(i=1;i<=n;i++)
75. {
76. tab[i].data=i;
77. tab[i].pNext=NULL;
78. }
79. }
80. void Insert(int n,int x)
81. {
82. struct Node \*p=&tab[n];
83. while(p->pNext!=NULL)
84. {
85. p=p->pNext;
86. }
87. struct Node \*new=(struct Node \*)malloc(sizeof(struct Node));
88. p->pNext=new;
89. new->data=x;
90. new->pNext=NULL;
91. }

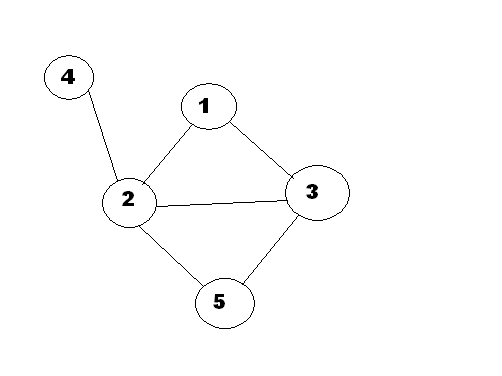
## 13、网络寻路(构图)

问题描述

X 国的一个网络使用若干条线路连接若干个节点。节点间的通信是双向的。某重要数据包，为了安全起见，必须恰好被转发两次到达目的地。该包可能在任意一个节点产生，我们需要知道该网络中一共有多少种不同的转发路径。

源地址和目标地址可以相同，但中间节点必须不同。

如下图所示的网络。



1 -> 2 -> 3 -> 1 是允许的

1 -> 2 -> 1 -> 2 或者 1 -> 2 -> 3 -> 2 都是非法的。

输入格式

输入数据的第一行为两个整数N M，分别表示节点个数和连接线路的条数(1<=N<=10000; 0<=M<=100000)。

接下去有M行，每行为两个整数 u 和 v，表示节点u 和 v 联通(1<=u,v<=N , u!=v)。

输入数据保证任意两点最多只有一条边连接，并且没有自己连自己的边，即不存在重边和自环。

输出格式

输出一个整数，表示满足要求的路径条数。

样例输入1

3 3  
1 2  
2 3  
1 3

样例输出1

6

样例输入2

4 4  
1 2  
2 3  
3 1  
1 4

样例输出2

10

本题的C参考代码如下：

1. #include<stdio.h>
2. #include<string.h>
3. #define MAXN 10010
4. #define MAXM 100010
5. int Du[MAXN],U[MAXM],V[MAXM];
6. int main()
7. {
8. int n,i,m;
9. long long ans=0;
10. scanf("%d%d",&n,&m);
11. memset(Du,0,sizeof(Du));
12. for(i=0;i<m;i++){
13. scanf("%d%d",&U[i],&V[i]);
14. Du[U[i]]++;
15. Du[V[i]]++;
16. }
17. for(i=0;i<m;i++)if(Du[U[i]]>1&&Du[V[i]]>1)ans+=(Du[U[i]]-1)\*(Du[V[i]]-1)\*2;
18. printf("%I64d\n",ans);
19. return 0;
20. }

## 14、高僧斗法(博弈论)

问题描述

　　古时丧葬活动中经常请高僧做法事。仪式结束后，有时会有“高僧斗法”的趣味节目，以舒缓压抑的气氛。  
　　节目大略步骤为：先用粮食（一般是稻米）在地上“画”出若干级台阶（表示N级浮屠）。又有若干小和尚随机地“站”在某个台阶上。最高一级台阶必须站人，其它任意。(如图1所示)  
　　两位参加游戏的法师分别指挥某个小和尚向上走任意多级的台阶，但会被站在高级台阶上的小和尚阻挡，不能越过。两个小和尚也不能站在同一台阶，也不能向低级台阶移动。  
　　两法师轮流发出指令，最后所有小和尚必然会都挤在高段台阶，再也不能向上移动。轮到哪个法师指挥时无法继续移动，则游戏结束，该法师认输。  
　　对于已知的台阶数和小和尚的分布位置，请你计算先发指令的法师该如何决策才能保证胜出。

输入格式

　　输入数据为一行用空格分开的N个整数，表示小和尚的位置。台阶序号从1算起，所以最后一个小和尚的位置即是台阶的总数。（N<100, 台阶总数<1000）

输出格式

　　输出为一行用空格分开的两个整数: A B, 表示把A位置的小和尚移动到B位置。若有多个解，输出A值较小的解，若无解则输出-1。

样例输入

1 5 9

样例输出

1 4

样例输入

1 5 8 10

样例输出

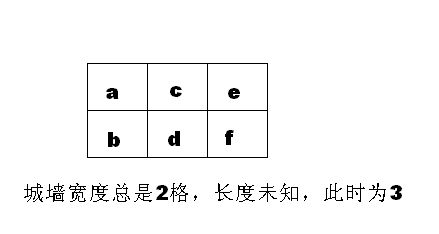
1 3

本题的C参考代码如下：

1. #include <stdio.h>   //参考蓝桥杯贴吧 dezhonger
2. int main()
3. {
4. int a[105],b[105],i=0,j,k,count,sum;
5. char c;
6. while(1)
7. {
8. scanf("%d%c",&a[i++],&c);
9. if(c=='\n')
10. break;
11. }
12. count=i;
13. for(i = 0;i < count-1;i++)
14. b[i]=a[i+1]-a[i]-1;
15. b[count-1]=0;
16. sum=b[0];
17. for(i = 2;i < count;i = i+2)
18. sum^=b[i];
19. if(sum == 0)
20. printf("-1\n");
21. else
22. {
23. for(i = 0;i < count;i++)
24. for(j = 1;j <= b[i];j++)
25. {
26. b[i] -= j;
27. if(i!=0)
28. b[i-1]+=j;
29. sum = b[0];
30. for(k = 2;k < count;k = k+2)
31. sum ^= b[k];
32. if(sum == 0)
33. {
34. printf("%d %d\n",a[i],a[i]+j);
35. break;
36. }
37. b[i] += j;
38. if(i != 0)
39. b[i-1] -= j;
40. }
41. }
42. return 0;
43. }
44. /\*
45. 1 3 5 7 12 14 17 26 38 45 66 100
46. 66 84
47. \*/

## 15、格子刷油漆(动态规划)

问题描述

　　X国的一段古城墙的顶端可以看成 2\*N个格子组成的矩形（如下图所示），现需要把这些格子刷上保护漆。  
  
  
　　你可以从任意一个格子刷起，刷完一格，可以移动到和它相邻的格子（对角相邻也算数），但不能移动到较远的格子（因为油漆未干不能踩！）  
　　比如：a d b c e f 就是合格的刷漆顺序。  
　　c e f d a b 是另一种合适的方案。  
　　当已知 N 时，求总的方案数。当N较大时，结果会迅速增大，请把结果对 1000000007 (十亿零七) 取模。

输入格式

　　输入数据为一个正整数（不大于1000）

输出格式

　　输出数据为一个正整数。

样例输入

2

样例输出

24

样例输入

3

样例输出

96

样例输入

22

样例输出

359635897

本题的C参考代码如下：

1. #include <stdio.h>
2. long long a[1001],b[1001],sum;
3. #define NUM 1000000007
4. int main()
5. {
6. int i,n;
7. scanf("%d",&n);
8. b[1]=1;
9. for (i=2;i<=n;i++)
10. b[i]=(b[i-1]\*2%NUM);
11. a[1]=1;a[2]=6;
12. for (i=3;i<=n;i++)
13. a[i]=(2\*a[i-1]+b[i]+4\*a[i-2])%NUM;
14. sum=4\*a[n];
15. for (i=2;i<n;i++)
16. sum=((sum+8\*b[n-i]\*a[i-1]%NUM)%NUM+(8\*a[n-i]\*b[i-1])%NUM)%NUM;
17. printf("%I64d\n",sum);
18. return 0;
19. }

## 16、农场阳光（计算几何）

问题描述

　　X星球十分特殊，它的自转速度与公转速度相同，所以阳光总是以固定的角度照射。  
　　最近，X星球为发展星际旅游业，把空间位置出租给Y国游客来晒太阳。每个租位是漂浮在空中的圆盘形彩云（圆盘与地面平行）。当然，这会遮挡住部分阳光，被遮挡的土地植物无法生长。  
　　本题的任务是计算某个农场宜于作物生长的土地面积有多大。

输入格式

　　输入数据的第一行包含两个整数a, b，表示某农场的长和宽分别是a和b，此时，该农场的范围是由坐标(0, 0, 0), (a, 0, 0), (a, b, 0), (0, b, 0)围成的矩形区域。  
　　第二行包含一个实数g，表示阳光照射的角度。简单起见，我们假设阳光光线是垂直于农场的宽的，此时正好和农场的长的夹角是g度，此时，空间中的一点(x, y, z)在地面的投影点应该是(x + z \* ctg(g度), y, 0)，其中ctg(g度)表示g度对应的余切值。  
　　第三行包含一个非负整数n，表示空中租位个数。  
　　接下来 n 行，描述每个租位。其中第i行包含4个整数xi, yi, zi, ri，表示第i个租位彩云的圆心在(xi, yi, zi)位置，圆半径为ri。

输出格式

　　要求输出一个实数，四舍五入保留两位有效数字，表示农场里能长庄稼的土地的面积。

样例输入

10 10  
90.0  
1  
5 5 10 5

样例输出

21.46

样例输入

8 8  
90.0  
1  
4 4 10 5

样例输出

1.81

样例输入

20 10  
45.0  
2  
5 0 5 5  
8 6 14 6

样例输出

130.15

本题的C参考代码如下：

1. 该题暂时没有人完全正确，暂时没有该语言的参考程序。

## 17、约数倍数选卡片（博弈论）

问题描述

　　闲暇时，福尔摩斯和华生玩一个游戏：  
　　在N张卡片上写有N个整数。两人轮流拿走一张卡片。要求下一个人拿的数字一定是前一个人拿的数字的约数或倍数。例如，某次福尔摩斯拿走的卡片上写着数字“6”，则接下来华生可以拿的数字包括：  
　　1，2，3, 6，12，18，24 ....  
　　当轮到某一方拿卡片时，没有满足要求的卡片可选，则该方为输方。  
　　请你利用计算机的优势计算一下，在已知所有卡片上的数字和可选哪些数字的条件下，怎样选择才能保证必胜！  
　　当选多个数字都可以必胜时，输出其中最小的数字。如果无论如何都会输，则输出-1。

输入格式

　　输入数据为2行。第一行是若干空格分开的整数（每个整数介于1~100间），表示当前剩余的所有卡片。  
　　第二行也是若干空格分开的整数，表示可以选的数字。当然，第二行的数字必须完全包含在第一行的数字中。

输出格式

　　程序则输出必胜的招法！！

样例输入

2 3 6  
3 6

样例输出

3

样例输入

1 2 2 3 3 4 5  
3 4 5

样例输出

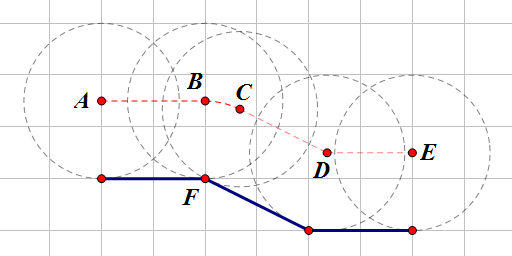
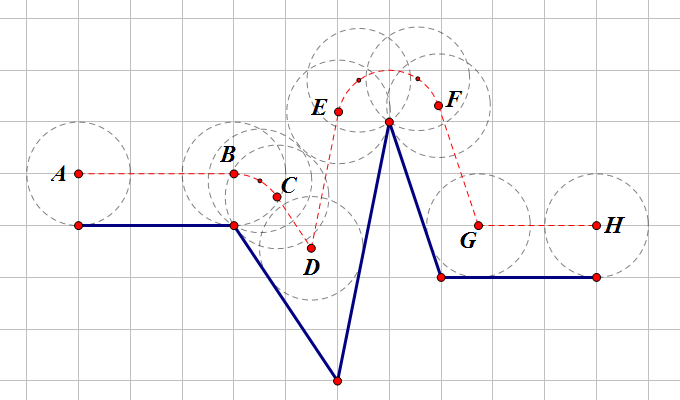
4

本题的C参考代码如下：

1. 该题暂时没有人完全正确，暂时没有该语言的参考程序。

## 18、车轮轴迹（计算几何）

问题描述

　　栋栋每天骑自行车回家需要经过一条狭长的林荫道。道路由于年久失修，变得非常不平整。虽然栋栋每次都很颠簸，但他仍把骑车经过林荫道当成一种乐趣。  
　　由于颠簸，栋栋骑车回家的路径是一条上下起伏的曲线，栋栋想知道，他回家的这条曲线的长度究竟是多长呢？更准确的，栋栋想知道从林荫道的起点到林荫道的终点，他的车前轮的轴（圆心）经过的路径的长度。  
　　栋栋对路面进行了测量。他把道路简化成一条条长短不等的直线段，这些直线段首尾相连，且位于同一平面内。并在该平面内建立了一个直角坐标系，把所有线段的端点坐标都计算好。  
　　假设栋栋的自行车在行进的过程中前轮一直是贴着路面前进的。  
  
  
　　上图给出了一个简单的路面的例子，其中蓝色实线为路面，红色虚线为车轮轴经过的路径。在这个例子中，栋栋的前轮轴从A点出发，水平走到B点，然后绕着地面的F点到C点（绕出一个圆弧），再沿直线下坡到D点，最后水平走到E点，在这个图中地面的坐标依次为：(0, 0), (2, 0), (4, -1), (6, -1)，前轮半径为1.50，前轮轴前进的距离依次为：  
　　AB=2.0000；弧长BC=0.6955；CD=1.8820；DE=1.6459。  
　　总长度为6.2233。  
  
　　下图给出了一个较为复杂的路面的例子，在这个例子中，车轮在第一个下坡还没下完时（D点）就开始上坡了，之后在坡的顶点要从E绕一个较大的圆弧到F点。这个图中前轮的半径为1，每一段的长度依次为：  
　　AB=3.0000；弧长BC=0.9828；CD=1.1913；DE=2.6848；弧长EF=2.6224； FG=2.4415；GH=2.2792。  
　　总长度为15.2021。  
  
　　现在给出了车轮的半径和路面的描述，请求出车轮轴轨迹的总长度。

输入格式

　　输入的第一行包含一个整数n和一个实数r，用一个空格分隔，表示描述路面的坐标点数和车轮的半径。  
　　接下来n行，每个包含两个实数，其中第i行的两个实数x[i], y[i]表示描述路面的第i个点的坐标。  
　　路面定义为所有路面坐标点顺次连接起来的折线。给定的路面的一定满足以下性质：  
  
　　\*第一个坐标点一定是(0, 0)；  
　　\*第一个点和第二个点的纵坐标相同；  
　　\*倒数第一个点和倒数第二个点的纵坐标相同；  
　　\*第一个点和第二个点的距离不少于车轮半径；  
　　\*倒数第一个点和倒数第二个点的的距离不少于车轮半径；  
　　\*后一个坐标点的横坐标大于前一个坐标点的横坐标，即对于所有的i，x[i+1]>x[i]。

输出格式

　　输出一个实数，四舍五入保留两个小数，表示车轮轴经过的总长度。  
　　你的结果必须和参考答案一模一样才能得分。数据保证答案精确值的小数点后第三位不是4或5。

样例输入

4 1.50  
0.00 0.00  
2.00 0.00  
4.00 -1.00  
6.00 -1.00

样例输出

6.22

样例说明

　　这个样例对应第一个图。

样例输入

6 1.00  
0.00 0.00  
3.00 0.00  
5.00 -3.00  
6.00 2.00  
7.00 -1.00  
10.00 -1.00

样例输出

15.20

样例说明

　　这个样例对应第二个图

数据规模和约定

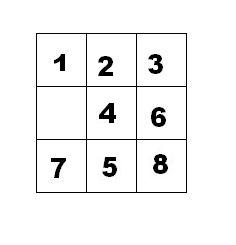
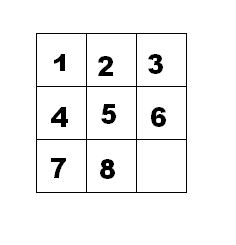
　　对于20%的数据，n=4；  
　　对于40%的数据，n≤10；  
　　对于100%的数据，4≤n≤100，0.5≤r≤20.0，x[i] ≤2000.0，-2000.0≤y[i] ≤2000.0。

本题的C参考代码如下：

1. 该题暂时没有人完全正确，暂时没有该语言的参考程序。

## 19、九宫重排  （排序）

问题描述

　　如下面第一个图的九宫格中，放着 1~8 的数字卡片，还有一个格子空着。与空格子相邻的格子中的卡片可以移动到空格中。经过若干次移动，可以形成第二个图所示的局面。  
  
　　我们把第一个图的局面记为：12345678.  
　　把第二个图的局面记为：123.46758  
　　显然是按从上到下，从左到右的顺序记录数字，空格记为句点。  
　　本题目的任务是已知九宫的初态和终态，求最少经过多少步的移动可以到达。如果无论多少步都无法到达，则输出-1。

输入格式

　　输入第一行包含九宫的初态，第二行包含九宫的终态。

输出格式

　　输出最少的步数，如果不存在方案，则输出-1。

样例输入

12345678.  
123.46758

样例输出

3

样例输入

13524678.  
46758123.

样例输出

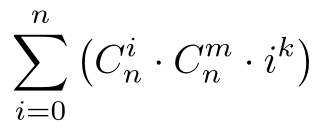
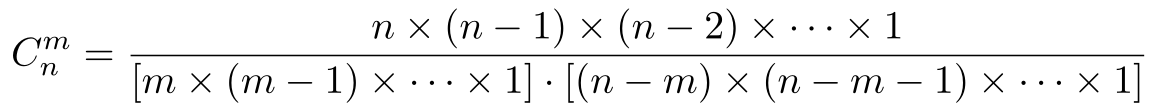
22

本题的C++参考代码如下：

1. #include<stdio.h>
2. #include<string.h>
3. #include<queue>
4. int d\_xy[][2]={0,-1,0,1,1,0,-1,0};
5. /\*
6. \* 双向广搜 康托展开   by 邱良雄
7. \*/
8. int  hash[362880][2];
9. int factory[]={1,1,2,6,24,120,720,5040,40320};
10. typedef  struct
11. {
12. int step;
13. int pos;
14. char state[9];
15. }Node;
16. int toHashValue(char \*map)
17. {
18. int value=0;
19. for(int i=0;i<9;i++)
20. {
21. int cnt=0;
22. for(int j=i+1;j<9;j++)
23. {
24. if(map[i]>map[j])
25. cnt++;
26. }
27. value+=cnt\*factory[8-i];
28. }
29. return value;
30. }
31. bool check(int x,int y)
32. {
33. if(x<0||x>=3||y<0||y>=3)
34. return true;
35. return false;
36. }
37. int initStr(char \*map)
38. {
39. for(int i=0;i<strlen(map);i++)
40. if(map[i]=='.')
41. {
42. map[i]='0';
43. return i;
44. }
45. return 0;
46. }
47. int bfs(char \*start,char \*end)
48. {
49. Node s,t;
50. s.pos=initStr(start);
51. strcpy(s.state,start);
52. std::queue<Node>q1,q2;
53. s.step=1;
54. q1.push(s);
55. s.pos=initStr(end);
56. strcpy(s.state,end);
57. q2.push(s);
58. hash[ toHashValue(start)][0]=1;
59. hash[ toHashValue(end)][1]=1;
60. while(!q1.empty()||!q2.empty())
61. {
62. if(!q1.empty())
63. {
64. s=q1.front();
65. q1.pop();
66. for(int k=0;k<4;k++)
67. {
68. int x=s.pos/3+d\_xy[k][0];
69. int y=s.pos%3+d\_xy[k][1];
70. if(check(x,y))
71. continue;
72. t.pos=x\*3+y;
73. strcpy(t.state,s.state);
74. t.step=s.step+1;
75. std::swap(t.state[t.pos],t.state[s.pos]);
76. if(hash[toHashValue(t.state) ][0])
77. continue;
78. hash[toHashValue(t.state)][0]=t.step;
79. if(hash[toHashValue(t.state)][1])
80. return  hash[toHashValue(t.state)][1]+s.step-1;
81. q1.push(t);
82. }
83. }
84. if(!q2.empty())
85. {
86. s=q2.front();
87. q2.pop();
88. for(int k=0;k<4;k++)
89. {
90. int x=s.pos/3+d\_xy[k][0];
91. int y=s.pos%3+d\_xy[k][1];
92. if(check(x,y))
93. continue;
94. t.pos=x\*3+y;
95. strcpy(t.state,s.state);
96. t.step=s.step+1;
97. std::swap(t.state[t.pos],t.state[s.pos]);
98. if(hash[toHashValue(t.state)][1])
99. continue;
100. hash[toHashValue(t.state)][1]=t.step;
101. if(hash[toHashValue(t.state)][0])
102. return  hash[toHashValue(t.state)][0]+s.step-1;
103. q2.push(t);
104. }
105. }
107. }
108. return -1;
110. }
111. int  main()
112. {
113. char start[9],end[9];
114. gets(start);
115. gets(end);
116. printf("%d\n",bfs(start,end));
117. return 0;
118. }

## 20、公式求值（组合数字）

问题描述

　　输入n, m, k，输出下面公式的值。  
  
　　其中C\_n^m是组合数，表示在n个人的集合中选出m个人组成一个集合的方案数。组合数的计算公式如下。  


输入格式

　　输入的第一行包含一个整数n；第二行包含一个整数m，第三行包含一个整数k。

输出格式

　　计算上面公式的值，由于答案非常大，请输出这个值除以999101的余数。

样例输入

3  
1  
3

样例输出

162

样例输入

20  
10  
10

样例输出

359316

数据规模和约定

　　对于10%的数据，n≤10，k≤3；  
　　对于20%的数据，n≤20，k≤3；  
　　对于30%的数据，n≤1000，k≤5；  
　　对于40%的数据，n≤10^7，k≤10；  
　　对于60%的数据，n≤10^15，k ≤100；  
　　对于70%的数据，n≤10^100，k≤200；  
　　对于80%的数据，n≤10^500，k ≤500；  
　　对于100%的数据，n在十进制下不超过1000位，即1≤n<10^1000，1≤k≤1000，同时0≤m≤n，k≤n。

提示

　　999101是一个质数；  
　　当n位数比较多时，绝大多数情况下答案都是0，但评测的时候会选取一些答案不是0的数据；

本题的C参考代码如下：

1. 该题暂时没有人完全正确，暂时没有该语言的参考程序。