



中國人民大學
RENMIN UNIVERSITY OF CHINA

期末复习

余力

buaayuli@ruc.edu.cn

三个维度

- 章节内容
- 基础模块
- 题目类型

一、课程章节内容

- 输入输出（整型、字符、字符串）
- 运算符（单目、算术、赋值、比较、逻辑、条件）
- 控制语句（if语句、while\for循环语句, break/continue）
- 数组（一维、二维、字符串）
- 函数（基本概念、变量类型、参数传递、数组为形参、嵌套）
- 递归（递推与搜索）
- 结构体（定义、使用；统计分析）
- 指针（简单指标、指向<数组、指针、结构体>；程序补全）
- 文件



中國人民大學
RENMIN UNIVERSITY OF CHINA



1.1 控制语句

if (表达式)

 语句 1;

else

 语句 2;

if (表达式) {

 语句块 1;

}

else {

 语句块 2;

}

➤ 例如: $\max = (a > b) ? a : b;$

 if (a > b)

 max = a;

 else

 max = b;

if (<条件表达式1>) <语句1>

else if (<条件表达式2>) <语句2>

else if (<条件表达式3>) <语句3>

.....

else if (<条件表达式n>) <语句n>

else <语句n+1>

switch (表达式)

{

 case 常量1: 语句1;

 case 常量2: 语句2;

 !

 case 常量n: 语句n;

 default: 语句n+1;

}

```
while ( 表达式 )  
{  
    语句块; (循环体)  
}
```

```
for(表达式1; 表达式2; 表达式3) {  
    循环体 (语句组)  
}
```

```
for (i=1,total=0;i<=1000;i++)  
{ printf("please enter amount:");  
  scanf("%f",&amount);  
  total= total+amount;  
  if (total>=SUM) break;  
}
```

```
while((c=getchar())!='\n')  
    printf("%c", c+2);  
  
do {  
    循环体语句块;  
} while ( 表达式 )
```

```
while (1) {  
    printf (" Guess an odd number in [1,99]\n");  
    scanf ("%d", &guess);  
    if (guess % 2 == 0) {  
        printf ("Please guess an ODD number\n");  
        continue; }  
    if (guess > num) printf ( "Too big\n" );  
    else if (guess < num) printf (" Too small\n ");  
    else {  
        printf ( "Bingo! The number is %d" , num);  
        break;  
    }  
}
```



中國人民大學
RENMIN UNIVERSITY OF CHINA



1.2 数组

```
int a[5] = { 3, 5, 4, 1, 2 };
```

a	3	5	4	1	2
下标	0	1	2	3	4

二维数组的初始化

```
int a[2][3]={{1,2},{4,5,6}};
```

或写成

```
int a[2][3]={1,2,4,5,6};
```




中國人民大學
RENMIN UNIVERSITY OF CHINA



1.3 函数

类型名 函数名()

{

函数体

}

类型名 函数名(void)

{

函数体

}

包括声明部分和
语句部分

c=max(a,b);

(**main**函数)

int max(int x, int y) (**max**函数)

{

int z;

z=x>y?x:y;

return(z);

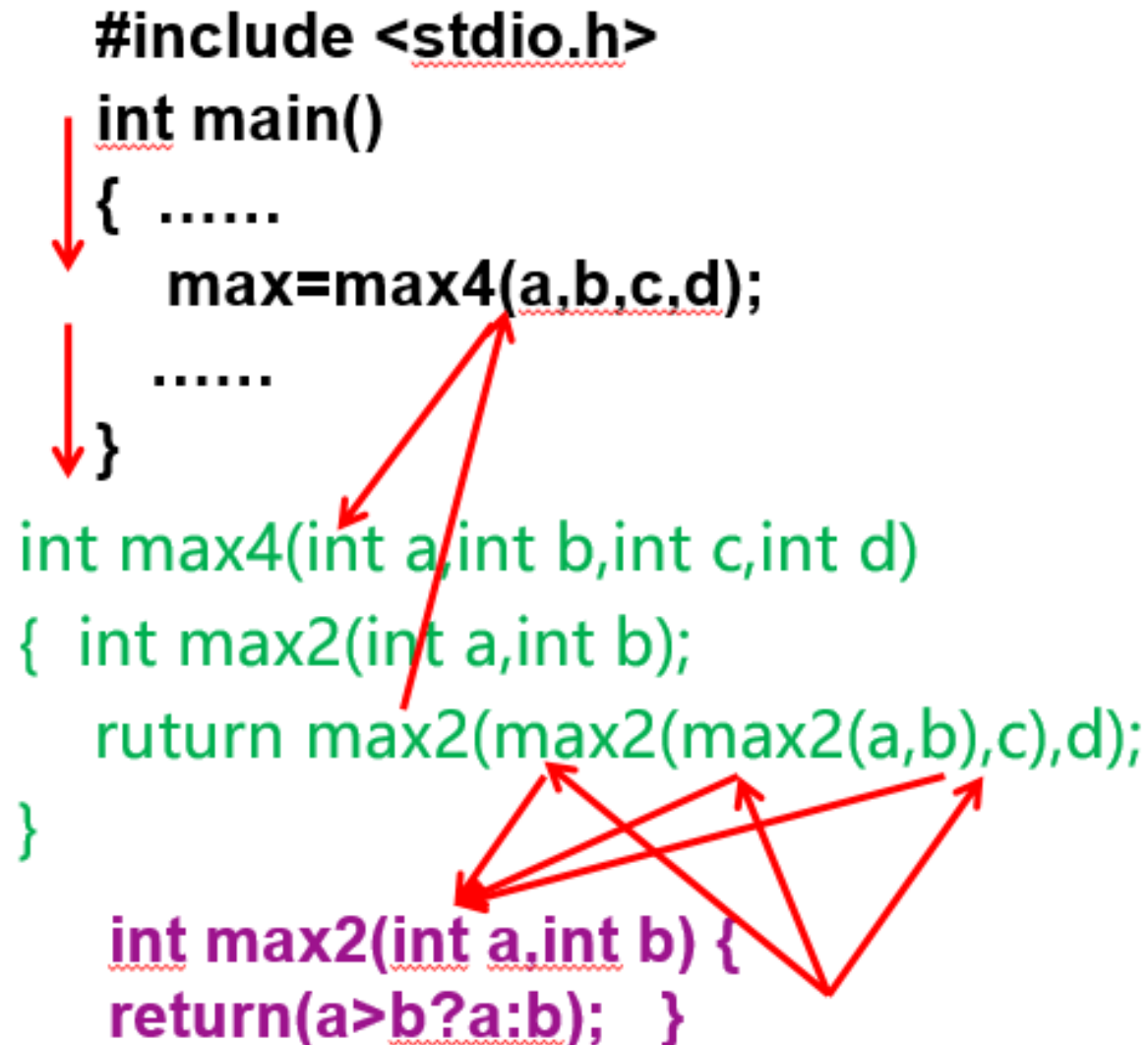
}

嵌套调用

```
#include <stdio.h>
int main()
{
    .....
    max=max4(a,b,c,d);
    .....
}

int max4(int a,int b,int c,int d)
{
    int max2(int a,int b);
    return max2(max2(max2(a,b),c),d);
}

int max2(int a,int b) {
    return(a>b?a:b); }


```

定义形参数组

```
float average(float array[])  
{ int i;  
  float aver,sum=array[0];  
  for(i=1;i<10;i++)  
    sum=sum+array[i];  
  aver=sum/10;  
  return(aver);  
}
```

相当于**score[0]**

相当于**score[i]**

```
float f1( int a)
{ int b,c;
  .....
}
```

a、b、c仅在此函数内有效

```
char f2(int x,int y)
{ int i,j;
  .....
}
```

x、y、i、j仅在此函数内有效

```
int main( )
{ int m,n;
  .....
  return 0;
}
```

m、n仅在此函数内有效

```
int p=1,q=5
float f1(int a)
{ int b,c; ..... }
```

```
char c1,c2;
char f2 (int x, int y)
{ int i,j; ..... }
int main ( )
{ int m,n;
  .....
  return 0;
}
```

#301 成绩统计

【问题描述】

期中考试后，老师需要对班里学生的成绩进行统计分析，了解各个分数段的人数。

给定所有学生的成绩和成绩分段方式，请你编程帮忙**以下任务**：

- 1) 统计各个分数段的人数，并按照分数段人数从大到小输出；
- 2) 将按分数段统计的结果用**模拟的直方图**显示出来。

【输入格式】

第 1 行包含 2 个整数 n 、 m 、 g ，分别表示学生人数、分数段个数、需解决的任务编号。

m 表示将成绩区间 $[0, 100]$ 平均分成 m 个分数段。例如 $m = 5$ 表示分为 5 个分数段，分别是 $[0, 20)$ 、 $[20, 40)$ 、 $[40, 60)$ 、 $[60, 80)$ 、 $[80, 100]$ 。

g 一共有 3 种取值，分别是 0、1、2，

- 0 表示任务 1、2 均需完成，且先完成任务 1，再完成任务 2；
- 1 表示只完成任务 1；
- 2 表示只完成任务 2。

第 2 行包含 n 个 $[0, 100]$ 之间的整数，为 n 个学生的期中考试成绩。

【输出格式】

任务 1 格式:

若干行，每行一个分数段以及该分数段的人数，分数段表示为 $[L,R)$ 或者 $[L,100]$ 的形式，其中 L 占 2 个字符的宽度， R 需要占用 3 个字符的宽度，分数段后面跟一个字符 $:$ ，之后输出一个空格，再输出该分数段的人数。

注意：如果该分数段的人数为 0，则不输出该分数段；按照分数段的人数从大到小的顺序输出，如果 2 个分数段人数相同，先输出分数段比较低的那一个。

任务 2 格式:

分数段的输出格式与任务 1 相同， $:$ 之后输出若干个字符 $*$ ，具体个数为该分数段的实际人数，或者该分数段人数归一化后的数量。人数最多的分数段能在显示在一行以内，归一化的方式为人数最多那个分数段最多输出 50 个字符 $*$ ，即如果人数多于 50，则输出 50 个字符 $*$ ，其他分数段按比例减少字符 $*$ 的个数（下取整）；否则直接输出实际数量的字符 $*$ 。

注意：任务 1 和任务 2 之间请输出一个空行。

【输入样例 1】

```
20 10 0.
88 90 75 68 77 85 86 99 100 95 60 55 32 93 63 60 78 96 70 80.
```

【输出样例 1】

```
[90,100]: 6.
[60, 70]: 4.
[70, 80]: 4.
[80, 90]: 4.
[30, 40]: 1.
[50, 60]: 1.

[ 0, 10]:.
[10, 20]:.
[20, 30]:.
[30, 40]:*.
[40, 50]:.
[50, 60]:*.
[60, 70]:****.
[70, 80]:****.
[80, 90]:****.
[90,100]:*****.
```

```

#include <stdio.h>
#include <math.h>
int main(){
    int n, m, g, i, j, sc[5001], a, b, c, d, t = 0, k[11] = {0}, ch[11][3];
    scanf("%d %d %d", &n, &m, &g);
    for (i = 0; i < n; i++)
        scanf("%d", &sc[i]);
    a = 100 / m;
    for (i = 0; i < n; i++) { // 每个分数
        for (j = 0; j < m; j++) // 每个分数段
            if (sc[i] >= j * a && sc[i] < (j + 1) * a)
                k[j] += 1;
        if (sc[i] == 100) k[m - 1] += 1;
    }
    for (i = 0; i < m; i++) {
        ch[i][0] = i;
        ch[i][1] = k[i];
    }
    for (i = 0; i < m; i++) // 统计结果排序
        for (j = 0; j < m; j++)
            if ((ch[i][1] > ch[j][1]) || (ch[i][1] == ch[j][1] && ch[i][0] < ch[j][0])) {
                c = ch[i][1]; ch[i][1] = ch[j][1]; ch[j][1] = c;
                d = ch[i][0]; ch[i][0] = ch[j][0]; ch[j][0] = d;
            }
}

```



```

if (g == 1) {
    for (i = 0; i < m; i++) {
        if (ch[i][1] != 0) {
            if (ch[i][0] == 0) {
                printf("[0, %d]", (ch[i][0] + 1) * a);
            }
            else {
                if ((ch[i][0] + 1) * a != 100) {
                    printf("[%d, %d]", ch[i][0] * a, (ch[i][0] + 1) * a);
                }
                else {
                    printf("[%d, 100]", ch[i][0] * a);
                }
            }
            printf(": %d", ch[i][1]);
            printf("\n");
        }
    }
}

```

```

if (g == 2) {
    if (ch[0][1] > 50) {
        for (j = 0; j < m; j++) {
            k[j] = k[j] * 50 / ch[0][1];
        }
    }
    for (i = 0; i < m; i++) {
        if (i == 0) {
            printf("[0, %d]", a);
        }
        else {
            if (i != m - 1) {
                printf("[%d, %d]", i * a, (i + 1) * a);
            }
            else {
                printf("[%d, 100]", i * a);
            }
        }
        printf(":");
        for (j = 0; j < k[i]; j++) {
            printf("**");
        }
        printf("\n");
    }
}

```

```

if (g == 0) {
    for (i = 0; i < m; i++) {
        if (ch[i][1] != 0) {
            if (ch[i][0] == 0) {
                printf("[0, %d]", (ch[i][0] + 1) * a);
            }
            else {
                if ((ch[i][0] + 1) * a != 100) {
                    printf("[%d, %d]", ch[i][0] * a, (ch[i][0] + 1) * a);
                }
                else {
                    printf("[%d, 100]", ch[i][0] * a);
                }
            }
            printf(": %d", ch[i][1]);
            printf("\n");
        }
    }
    printf("\n");
    if (ch[0][1] > 50) {
        for (j = 0; j < m; j++) {
            k[j] = k[j] * 50 / ch[0][1];
        }
    }
    for (i = 0; i < m; i++) {
        if (i == 0) {
            printf("[0, %d]", a);
        }
        else {
            if (i != m - 1) {
                printf("[%d, %d]", i * a, (i + 1) * a);
            }
            else {
                printf("[%d, 100]", i * a);
            }
        }
        printf(":");
        for (j = 0; j < k[i]; j++) {
            printf("**");
        }
        printf("\n");
    }
}

```

```

int n, m, g, i, j, sc[5001], a, b, c, d, t = 0, k[11] = {0}, ch[11][3];
int main() {
    void print_result();
    void print_figure();
    scanf("%d %d %d", &n, &m, &g);
    for (i = 0; i < n; i++)
        scanf("%d", &sc[i]);
    a = 100 / m;
    for (i = 0; i < n; i++) { // 每个分数
        for (j = 0; j < m; j++) // 每个分数段
            if (sc[i] >= j * a && sc[i] < (j + 1) * a)
                k[j] += 1;
        if (sc[i] == 100) k[m - 1] += 1;
    }
    for (i = 0; i < m; i++) {
        ch[i][0] = i;
        ch[i][1] = k[i];
    }
    for (i = 0; i < m; i++) // 统计结果排序
        for (j = 0; j < m; j++)
            if ((ch[i][1] > ch[j][1]) || (ch[i][1] == ch[j][1] && ch[i][0] < ch[j][0])) {
                c = ch[i][1]; ch[i][1] = ch[j][1]; ch[j][1] = c;
                d = ch[i][0]; ch[i][0] = ch[j][0]; ch[j][0] = d;
            }
    if (g == 1) print_result();
    if (g == 2) print_figure();
    if (g == 0) { print_result(); print_figure(); }
    return 0;
}

```

```

void print_result()
{
    for (i = 0; i < m; i++)
        if (ch[i][1] != 0) {
            if (ch[i][0] == 0)
                printf("[0, %d]", (ch[i][0] + 1) * a);
            else
                if ((ch[i][0] + 1) * a != 100)
                    printf("[%d, %d]", ch[i][0] * a, (ch[i][0] + 1) * a);
                else
                    printf("[%d, 100]", ch[i][0] * a);
            printf(": %d", ch[i][1]);
            printf("\n");
        }
}

```

```

void print_figure()
{
    if (ch[0][1] > 50)
        for (j = 0; j < m; j++)
            k[j] = k[j] * 50 / ch[0][1];
    for (i = 0; i < m; i++) {
        if (i == 0) printf("[0, %d]", a);
        else
            if (i != m - 1) printf("[%d, %d]", i * a, (i + 1) * a);
            else printf("[%d, 100]", i * a);
        printf(":");
        for (j = 0; j < k[i]; j++) printf("**");
        printf("\n");
    }
}

```



中國人民大學
RENMIN UNIVERSITY OF CHINA



1.4 结构体

结构体类型的定义

- 结构体：表示一组类型可能不同的数据

```
struct Student {  
    char name[20];           //姓名  
    char sex;                //性别  
    unsigned long birthday; //生日  
    float height;            //身高  
    float weight;            //体重  
};
```

```
struct farm {  
    int num;  
    int state;  
    int max_cap;  
    int basic_cost;  
    int pig_cost;  
};
```

```
struct Stu {  
    int id;  
    int kemu;  
    double aver;  
};
```

```
#include <stdio.h>

struct Student //名为student的结构类型
{
    char name[20];           //姓名
    char sex;                //性别
    unsigned long birthday;  //生日
    float height;            //身高
    float weight;            //体重
};

struct Student Room [ 4 ] = { //定义全局结构数组/初始化
    {"Li li", 'M', 19840318, 1.82, 65.0 },
    {"Mi mi", 'M', 19830918, 1.75, 58.0 },
    {"He lei", 'M', 19841209, 1.83, 67.1 },
    {"Ge li", 'M', 19840101, 1.70, 59.0 }
};
```

```

int main() {
    struct Student q;    //结构变量q, 用于交换时保存中间结果
    int i = 0 ; int j =0;
    //按照年龄排序
    for ( j = 0; j < 3; j = j+1 )
        for ( i = 0; i < 3 - j ; i = i +1 )
            if ( Room[ i ].birthday > Room[i+1].birthday )
                {   q= Room[ i ]; //交换, 结构变量赋值
                    Room[ i ]=Room[ i+1];
                    Room[ i+1]= q;
                }

    for( i = 0; i < 4; i = i + 1 ) { //输出排序后的结果
        printf( "%s\n", Room[i].name );
        printf( "%c\n", Room[i].sex );
        printf( "%d\n", Room[i].birthday );
        printf( "%f\n", Room[i].height );
        printf( "%f\n", Room[i].weight );
    }
    return 0;
}

```



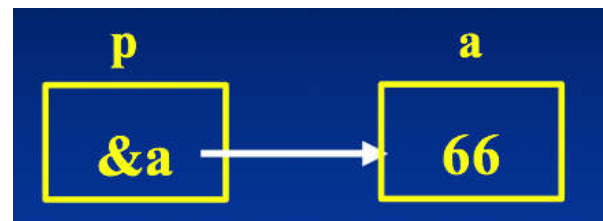
中國人民大學
RENMIN UNIVERSITY OF CHINA



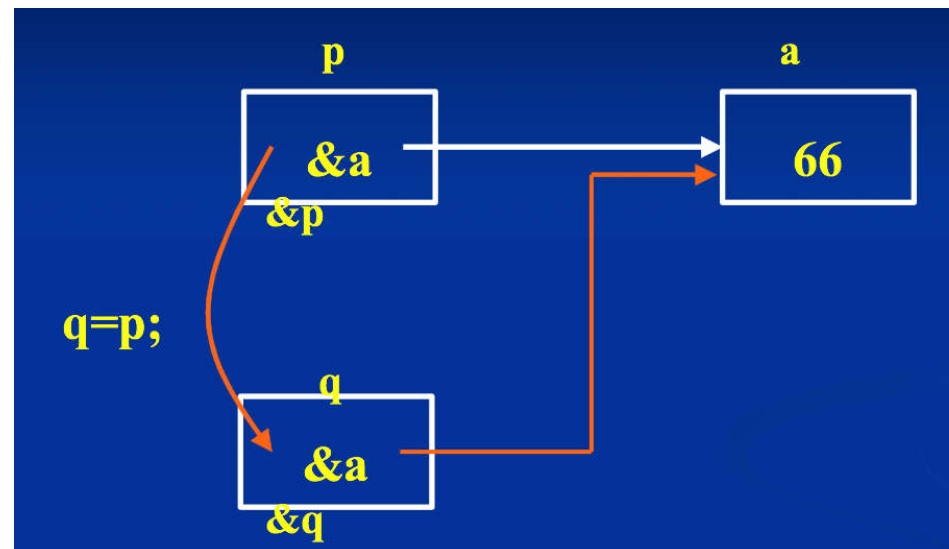
1.5 指针

理解指针

- `p = &a;` //将变量 a 的地址赋给 p



- `q = p;`



指针数组

■ 概念

- 指针数组是指针的数组
- 数组单元中存放的是地址，这些地址指向同一种数据类型的变量。

■ 指针数组的定义和初始化

- `int ar0[]={0,1,2};`
- `int ar1[]={1,2,3};`
- `int ar2[]={2,3,4};`
- `int* p[3] = {ar0, ar1, ar2}`
- `P[0]= ar0 P[0]= &ar0[0]`

动态数组

```
#include<iostream>
using namespace std;
#include <stdlib.h>
int main() {
    int *ptr;
    int len;
    cout << "Input array length: ";
    cin >> len;
    ptr = (int *) malloc( len * sizeof(int) );
    for (int i = 0; i < len; i++) {
        ptr[i] = i + 1;
        cout << ptr[i] << endl;
    }
    free(ptr);
    return 0;
}
```

引入stdlib.h头文件!

对字符指针和字符数组的赋值

1、对字符指针变量赋值的写法

(1) `char *p;`
 `p = "computer" ;`
(2) `char *p= "computer" ;`

以上两种都行。可以整体赋值。

2、对字符数组赋初值的写法

(1) `char as[12]= "department" ;// 可以。在定义时可以整体赋值`
 `char as[] = "department" ;// 可以。在定义时可以整体赋值`

(2) `char as[12];`
 `as = "department" ; // 不可以! 不可以整体赋值`
 `as[12]= "department" ; //不可以! 不可以整体赋值`

字符串处理函数

- C的库函数提供一组字符串处理函数
- 头文件: `string.h`
- 常用的字符串操作函数
 - 字符串长度: `strlen()`
 - 字符串连接: `strcat()`, `strncat()`
 - 字符串比较: `strcmp()`, `strncmp()`
 - 字符串拷贝: `strcpy()`, `strncpy()`

```
#include <stdio.h>
#include <stdlib.h>
// 传地址（指针），修改指针所间接访问的内容
void swap1(int * x, int * y)
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
    printf( "子函数: *x=%d *y=%d\n" , *x, *y);
}
```

✓ 交换成功

// 传值，修改的子函数空间的变量值

void swap2(int x, int y)

```
{   int temp;
    temp = x; x = y; y = temp;
    printf( "子函数: x=%d y=%d\n" , x, y);
}
```

✗ 交换失败

// 传地址（指针），修改指针本身

void swap3(int *x, int *y)

```
{   int *p;
    p = x; x = y; y = p;
    printf( "子函数: *x=%d *y=%d\n" , *x, *y);
}
```

✗ 交换失败

// C++的引用调用方式

```
void swap4(int &x, int &y)
```

```
{
```

```
    int temp;
```

```
    temp = x;
```

```
    x = y;
```

```
    y = temp;
```

```
    printf( "子函数: x=%d y=%d\n" , x, y);
```

```
}
```

✓ 交换成功

指向数组的指针

```
#include <stdio.h>

int main()
{ float score[ ][4]={{60,70,80,90}, {56,89,67,88},{34,78,90,66}};
  float *search(float (*pointer)[4],int n);
  float *p; int i, k;
  scanf("%d",&k);
  printf("The scores of No.%d are:\n",k);
  p=search(score, k);                                返回k号学生课程首地址
  for(i=0;i<4;i++)
    printf("%5.2f\t",*(p+i));
  printf("\n");
  return 0;
}
```

```
float *search(float (*pointer)[4],int n)
{ float *pt;
  pt=*(pointer+n);
  return(pt);
}
```


使用指针数组处理结构

- 如何使用指针数组完成排序?

- 如何声明指向结构的指针数组

```
struct student *p[4]={&Room[0], &Room[1],  
                      &Room[2], &Room[3] };
```

- 如何使用指针完成排序

```
for (int i=0; i<3; i++)  
for (int j=0; j<=3-i; j++)  
    if ( p[j]->height < p[j+1]->height ) {  
        struct student *q = p[j];  
        p[j] = p[j+1];  
        p[j+1] = q;    }
```

二、编程必须掌握的基础模块

- 输入（数值、字符与数值）
- 变量交换（值交换、地址）
- 整数取整(四舍五入)、数位提取、数位和、字符串与小数转换
- 字母大小写判断，加密、解密
- 排序（冒泡、快排、选择、插入、多排序）
- 素数、最小公倍数、最大公约数、因数分解
- 字符串操作（相等、约等、复制、插入、子串）
- 集合操作（交集、并集）
- 全排列、半排列、组合
- 回文
- 二分查找

三、题型分类

- 数值计算
- 统计排序
- 数组字符
- 循环枚举
- 递归搜索



中國人民大學
RENMIN UNIVERSITY OF CHINA



3.1 数值计算

输入格式

输入样例 ↵

3-4 ↵

3-8-9-10 ↵

2-5--3-5 ↵

7-0--1-4 ↵

```
int main()
{
    int n, m, sum = 0, x;
    scanf("%d-%d", &n, &m);
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
        {
            scanf("%d", &x);
            ....
        }
    return 0;
}
```

输入样式 ↵

3-5 ↵

2017101000-10-2.1-1.2-4.3-2.4-2.5-5.1-5.2-1.6-4.2-4.4 ↵

2017101001-9-2.1-2.2-2.5-3.2-1.1-1.3-4.3-4.4-5.2 ↵

2017101002-10-1.4-1.5-2.1-2.2-3.4-3.4-4.1-4.6-5.4-5.5 ↵

```
scanf("%d-%d", &n, &k);
for (s = 0; i = 0; i < n; i++)
{
    scanf("%s", &id[i]);
    scanf("%d", &num[i]);
    for (j = 0; j < num[i]; j++)
    {
        scanf("%d.%d", &date[s], &class1[s]);
        s += 1;
    }
}
```

```

/*
输入样例 //第一行以
5
aaaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbb
ccccccccccccccccccc
ddddddddddddddddddd
eeeeeeeeeeeeeeeeeee
*/

```

```

int i, n;
char a[1000][20];
scanf("%d", &n);
getchar();
//如果没有上面getchar(), 下面gets就出错
//第一个gets获得是空串, 导致最后一个字符串没有获得
for (i = 0; i < n; i++)
    gets(a[i]); //用gets出错

```

```

int i, n;
char a[1000][20];
scanf("%d", &n);
for (i = 0; i < n; i++)
    scanf("%s", a[i]); //正常,但如果是gets就会出错。
//但用scanf的缺点是输入字符串不能有空格,
//用gets的好处是可以输入包含有空格的字符串

printf("%d\n", n);
for (i = 0; i < n; i++)
    puts(a[i]);

```

■ 遇 “空格” 都结束

#161 数据加密

某个公司采用公用电话传递数据，数据是四位的整数，数据在传递过程中是加密的：每位数字都加上 5，得到的结果除以 10 的余数代替该数字，再将第一位和第四位交换，第二位和第三位交换。请你编写程序按照上述规则加密数据。

输入格式

□□输入只有一行，包括一个 4 位数的正整数 d ($1000 \leq d \leq 9999$)，表示加密前的数据。

输出格式

□□输出只有一行，也是一个 4 位数的正整数，表示加密后的数据。

输入样例

1235

输出样例

876

特殊提示

□□【样例 1 说明】

□□1235 每位上数字加 5 后模 10 得到的新数字是 6780，按照要求第一位第四位交换，第二位第三位交换后是 876（先导 0 不输出）。

读取每一位

#304 整数计数

编写一个程序计算整数区间[a, b]内，其个位数是 n，且能被 k 整除的 m 位正整数共有多少个。

【输入格式】

输入只有一行，输入 5 个整数 a、b、n、k、m，空格分隔，其中： $1 \leq a \leq b \leq 1,000,000$ ，且 $0 \leq n, k, m \leq 9$ 。

【输出格式】

输出一行，为符合要求的整数个数。

【样例输入 1】

1019-1-2-2

【样例输出 1】

0

【样例输入 2】

1050-4-2-2

【样例输出 2】

4

```
int main()
{
    int a,b,c,d,e,s=0,i,f;
    scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
    f=(pow(10,e-1));
    for(i=a;i<=b;i++)
        if(i%10==c)
            if(i%d==0)
                if(i/(10*f)<1&&i/f>=1) s++;
    printf("%d",s);
    return 0;
}
```


#462 统计字符

```
#include <stdio.h>
```

```
int main() {
```

```
    char c;
```

```
    int i;
```

```
    int cnt[26] = {0};
```

```
    while ((c = getchar()) != '\n') {
```

```
        if (c >= 'A' && c <= 'Z') {
```

```
            cnt[c - 'A']++;
```

```
        }
```

```
    }
```

```
    for (i = 0; i < 26; i++) {
```

```
        char s;
```

```
        s = 'A' + i;
```

```
        printf("%c:%d\n", s, cnt[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

#379149 #462 统计字符 Time Limit Exceeded 0 3068 ms 384 KB cpp / 288 B 余力(yuli) 2021/10/22 下午7:05

▶ 测试点 #0	2	得分: 0	用时: 1007 ms	内存: 384 KiB
▶ 测试点 #1	2	得分: 0	用时: 1009 ms	内存: 384 KiB
▶ 测试点 #2	2	得分: 0	用时: 1052 ms	内存: 264 KiB

```
#include <stdio.h>
```

```
#include <string.h>
```

```
int main() {
```

```
    char s[10000];
```

```
    int a[26] = {0};
```

```
    gets(s);
```

```
    for (int i = 0; i < strlen(s); i++)
```

```
        if (s[i] >= 65 && s[i] <= 90)
```

```
            a[s[i] - 'A']++;
```

```
    for (int i = 0; i < 26; i++)
```

```
        printf("%c:%d\n", 'A' + i, a[i]);
```

```
    return 0;
```

```
}
```

#161 数据加密

```
int main() {
```

```
→ int i, j, n;
```

```
→ scanf("%d", &n);
```

```
→ int x[4], y[4];
```

```
→ x[3] = n % 10;
```

```
→ x[2] = n / 10 % 10;
```

```
→ x[1] = n / 100 % 10;
```

```
→ x[0] = n / 1000 % 10;
```

```
→ for (i = 0; i <= 3; i++) {
```

```
→     x[i] += 5;
```

```
→     if (x[i] >= 10) x[i] = x[i] % 10;
```

```
→     x[i] = x[i] % 10;
```

```
→ }
```

数位模块

```
y[0] = x[3];
```

```
y[1] = x[2];
```

```
y[2] = x[1];
```

```
y[3] = x[0];
```

```
for (i = 0; i <= 3; i++)
```

```
→ if (y[i] == 0) → continue;
```

```
→ else → break;
```

```
for (j = i; j <= 3; j++)
```

```
→ printf("%d", y[j]);
```

```
return 0;
```

条件输出模块

筛法求素数

```
int main() {  
    int n, prime[10000];  
    scanf("%d", &n);  
    prime[1] = 0;  
    for (int i = 2; i <= n; i++)  
        prime[i] = 1;  
    for (int i = 2; i <= n; i++) {  
        if (prime[i] == 1)  
            for (int j = 2; j * i <= n; j++)  
                prime[i * j] = 0;  
    }  
    for (int i = 2; i <= n; i++)  
        if (prime[i])  
            printf("%d ", i);  
    return 0;  
}
```

筛法求素数.cpp

#299 质因数分解

给定一个整数 n ，请给出它的质因数分解。如 28 可以如下分解： $28=2^2\times 7$

请你输出的每个质因数以及该质因数的指数。

输入格式

一行，包含一个整数 n ($2\leq n\leq 10000$)。

输出格式

若干行，每行输出整数 n 的一个质因数，以及该质因数的指数，之间用一个冒号：隔开，且按质因数从小到大的顺序输出。

输入样例

28

输出样例

2:2

7:1

数值计算类

#161 数据加密

某个公司采用公用电话传递数据，数据是四位的整数，数据在传递过程中是加密的：每位数字都加上 5，得到的结果除以 10 的余数代替该数字，再将第一位和第四位交换，第二位和第三位交换。请你编写程序按照上述规则加密数据。

输入格式

□□输入只有一行，包括一个 4 位数的正整数 d ($1000 \leq d \leq 9999$)，表示加密前的数据。

输出格式

□□输出只有一行，也是一个 4 位数的正整数，表示加密后的数据。

输入样例

1235

输出样例

876

特殊提示

□□【样例 1 说明】

□□1235 每位上数字加 5 后模 10 得到的新数字是 6780，按照要求第一位第四位交换，第二位第三位交换后是 876（先导 0 不输出）。

读取每一位

#310 同构数

【问题描述】 ↵

计算正整数 $[a,b]$ 之间的全部“同构数”之和。所谓“同构数”，是指一个正整数 n 是它平方数的尾部，则称 n 为同构数。如 6 的平方是 36，6 出现在 36 的右端，6 就是同构数。76 的平方数是 5776，76 是同构数。↵

【输入格式】 ↵

输入只有一行，输入两个正整数 a 和 b ，中间由一个空格分隔，其中： $1 \leq a \leq b \leq 10000$ 。↵

【输出格式】 ↵

输出一行，一个正整数，为 a 、 b 之间同构数之和。↵

【输入样例 1】 ↵

1 100↵

【输入样例 2】 ↵

80 100↵

【输出样例 1】 ↵

113↵

【输出样例 2】 ↵

0↵

【数据规模说明】 ↵

$1 \leq a \leq b \leq 10000$ 。↵

76的平方数是5776，76是同构数

```
#include <stdio.h>↵
```

```
#include <math.h>↵
```

```
int main() {↵
```

```
→ int from, to, n, m, weishu, sum=0, i;↵
```

```
→ scanf("%d%d", &from, &to);↵
```

```
→ for (n = from; n <= to; ++n) {↵
```

```
→ → i = n;↵
```

```
→ → weishu = 0;↵
```

```
→ → do {i = i / 10;↵
```

```
→ → → weishu++;↵
```

```
→ → } while (i != 0);↵
```

```
→ → m = n * n;↵
```

```
→ → for (i = 0; i < weishu; i++)↵
```

```
→ → → m = m / 10;↵
```

```
→ → for (i = 0; i < weishu; i++)↵
```

```
→ → → m = m * 10;↵
```

```
→ → if (n == n * n - m) · sum += n;↵
```

```
→ }↵
```

```
→ printf("%d", sum);↵
```

```
→ return 0;↵
```

```
}↵
```

数位模块

```
for (n = from; n <= to; ++n) {↵
```

```
→ i = n;↵
```

```
→ weishu = 0;↵
```

```
→ do {i = i / 10;↵
```

```
→ → weishu++;↵
```

```
→ } while (i != 0);↵
```

```
→ m = n * n;↵
```

```
→ for (i = 0; i < weishu; i++)↵
```

```
→ → m = m / 10;↵
```

```
→ for (i = 0; i < weishu; i++)↵
```

```
→ → m = m * 10;↵
```

```
→ if (n == n * n - m) · sum += n;↵
```

```
}↵
```



中國人民大學
RENMIN UNIVERSITY OF CHINA



3.2 统计排序

```
scanf ("%d", &n);
```

```
for (i = 0; i < n; i++) { // 输入n个信息
```

```
    scanf ("%d %d", &StuNo, &KemuNum);
```

```
    for (sum = 0, j = 0; j < KemuNum; j++) {
```

```
        scanf ("%d", &tmp);
```

```
        sum = sum + tmp; }
```

```
    if (KemuNum >= 2) { count++ ; }
```

```
}
```

```
// 输出结果
```

```
if ( count == 0 )
```

```
    printf ("NO");
```

```
else {
```

```
    for (i = 0; i < count - 1; i++)
```

```
    for (j = 0; j < count - i - 1; j++)
```

```
        {
```

```
            for (i = 0; i < count; i++)
```

```
                printf( );
```

```
    }
```

统计分析类
框架

```

→ if (count == 0) {
→     → printf("NO");
→     → return 0; }
→ else {
→     → for (i = 0; i < count - 1; i++)
→         → for (j = 0; j < count - i - 1; j++)
→             → if ((fabs(aver[j] - aver[j + 1]) < 1e-7 && id[j] > id[j + 1]) || aver[j] < aver[j + 1]) {
→                 → tmp = id[j]; → id[j] = id[j + 1]; → id[j + 1] = tmp;
→                 → tmpf = aver[j]; → aver[j] = aver[j + 1]; → aver[j + 1] = tmpf; }
→     → for (i = 0; i < count; i++)
→         → printf("%d %.2lf\n", id[i], int(aver[i] * 100) / 100.0);

→     → return 0;
→     → }

```

a[0]---a[n-1]·n 个数排序

```
for (i = 0; i < n-1; i++)  
    → for (j = 0; j < n-1-i; j++)  
        → if (a[j] < a[j+1])  
            → { tmp = a[j]; a[j] = a[j+1]; a[j+1] = tmp; }
```

a[1]---a[n]·n 个数排序

```
for (i = 1; i < n; i++)  
    → for (j = 1; j < n-i; j++)  
        → if (a[j] < a[j+1])  
            → { tmp = a[j]; a[j] = a[j+1]; a[j+1] = tmp; }
```

多排序依据

```
for (i = 0; i < n - 1; i++)  
→ for (j = 0; j < n - 1 - i; j++)  
→ → if (aver[j] < aver[j + 1] || (fabs(aver[j] - aver[j + 1]) < 1e-7 && id[j] > id[j + 1])) {  
→ → → tmp = id[j]; → id[j] = id[j + 1]; → id[j + 1] = tmp;  
→ → → tmpf = aver[j]; → aver[j] = aver[j + 1]; → aver[j + 1] = tmpf; }
```

**(aver[j] < aver[j + 1] || (fabs(aver[j] -
aver[j + 1]) < 1e-7 && id[j] > id[j + 1])**

#91 相似乐曲

所谓“与 Q 最相似的 k 首乐曲”，即与 Q 的欧几里得距离最小的前 k 首乐曲。

【输入格式】

- 第 1 行，表示查询乐曲，一个正整数 n_0 ($1 \leq n_0 \leq 100$)，表示乐曲长度，后面有 n_0 个整数，每个整数在 $[0, 255]$ 内，表示一个频率。
- 第 2 行，两个整数 n 和 k ，用空格隔开。表示有 n 首乐曲， $1 \leq n \leq 100$ ，查找最相似的 k ($1 \leq k \leq n$) 首乐曲。
- 第 3 行到 $n+2$ 行，表示编号从 0 到 $n-1$ 的 n 首乐曲。每行一个正整数 n_i ($1 \leq n_i \leq 100$)，表示该乐曲长度，后面 n_i 个整数，每个整数在 $[0, 255]$ 内，表示一个频率。

【输出格式】

输出 k 个整数，与查询乐曲最相似的乐曲的编号，注意乐曲编号范围是 $[0, n-1]$ 。按相似度从高到低（即：欧式距离从小到大）的顺序输出。若两首乐曲与 Q 的距离相同，则编号小的排名靠前。

【输入样例】

4 10 12 245 245

3 1

6 2 4 2 250 250 250

1 189

4 10 12 245 245

【输出样例】

2

```
int main(){  
    int n,k,n0,i,j,count0[16]={0},m;  
    int a[100],b[100][3],d[100][100],count[100][16]={0}; //a 存曲子,  
    float sim[100]; //c[i]存相似度.  
    scanf("%d",&n0); //曲长.  
    for(i=0;i<n0;i++){  
        scanf("%d",&a[i]);  
        scanf("%d%d",&n,&k); //n 首曲子找 k 个最相近.  
        for(i=0;i<n;i++){  
            b[i][2]=i; //编号.  
            for(j=0;j<b[i][1];j++){  
                scanf("%d",&d[i][j]);  
            }  
        }  
    }  
}
```

```

... for(i=0;i<16;i++){
...     for(j=0;j<n0;j++){
...         if(a[j]>=0+i*16&& a[j]<=15+i*16){
...             count0[i]++; //识别曲每个区间的频率
...         }
...     }
... }

```

..

```

... for(i=0;i<n;i++){
...     for(j=0;j<16;j++){
...         for(m=0;m<b[i][1];m++){
...             if(d[i][m]>=0+j*16&& d[i][m]<=15+16*j){
...                 count[i][j]++; //每首曲子每个区间的频率
...             }
...         }
...     }
... }

```

//上面的可以简化换成如下的.

```

... //for(i=0;i<n;i++){
... //    for(m=0;m<b[i][1];m++){
... //        j=d[i][m]%16;
... //        count[i][j]++; //每首曲子每个区间的频率
... //    }
... }

```

```

---- int sum;
---- for(i=0;i<n;i++){
----- sum=0;
----- for(j=0;j<16;j++){
sum=sum+(count0[j]-count[i][j])*(count0[j]-count[i][j]);
----- sim[i]=sqrt(sum); //第 i 首曲子的相似度
    }
}

```

```

---- float max; int t;
---- for(i=0;i<n-1;i++){
----- for(j=0;j<n-1-i;j++){
----- if(sim[j]>sim[j+1] || ((abs(sim[j+1]-sim[j])<1e-6)&&b[j][2]>b[j+1][2])){
----- max=sim[j]; sim[j]=sim[j+1]; sim[j+1]=max;
----- t=b[j][2]; b[j][2]=b[j+1][2]; b[j+1][2]=t;
----- }
}

```

```

---- for(i=0;i<k;i++){
----- printf("%d-",b[i][2]); //要有空格
---- return 0;
}

```

#91 相似乐曲.cpp

#307 生辰八字

大富商杨家有一女儿到了出阁的年纪，杨老爷决定面向全城适龄男士征婚。杨老爷遵循传统，决定按生辰八字作为选女婿的依据。每个应征的男士须提供自己的生辰八字，亦即八个正整数，每个数的取值范围均在[1, 24]。杨老爷特意聘请了黄半仙来算命，选择和女儿最契合的前 k 个男士作为候选。黄半仙采用的算命方法是西洋传入的余弦相似度，具体做法如下：给定两个生辰八字 $A = [a_1, a_2, \dots, a_8]$, $B = [b_1, b_2, \dots, b_8]$, 则

【样例输出】

1012345678

$$\text{Similarity}(A, B) = \sum_{i=1}^8 a_i * b_i / (\sqrt{\sum_{i=1}^8 (a_i)^2} * \sqrt{\sum_{i=1}^8 (b_i)^2})$$

例如，若 $A = [10, 1, 1, 1, 1, 1, 1, 1]$ ， $B = [1, 1, 1, 1, 1, 1, 1, 1]$ ，则 $\text{Similarity}(A, B) = (10*1 + 1*1 + \dots + 1*1) / (\text{sqrt}(10^2 + 1^2 + \dots + 1^2) + \text{sqrt}(1^2 + 1^2 + \dots + 1^2)) = 1.29$

黄半仙认为一个男士和小姐两人的生辰八字的余弦相似度越大，两人就越契合。

【输入格式】

第 1 行两个整数， n 和 k ，($1 \leq n \leq 100, 1 \leq k \leq n$)，表示有 n 个男士应征，以及需要选择 k 人进入候选名单。

第 2 行 8 个整数，表示杨小姐的生辰八字。

后面 n 行，每行 9 个整数，第一个数字是某男士的身份证号，8 位整数；后面 8 个数字是该男士的生辰八字。整数之间均以空格隔开。

【样例输入】

2 1

1 1 1 1 1 1 1 1

1012345678 1 1 1 1 1 1 1 1

1087654321 1 1 1 1 1 8 8 8

【输出格式】

k 个整数，表示契合度最好的前 k 位男士的身份证号，按相似度从高到低排列。

注意：若两个男士和小姐的相似度相同，则身份证号码大的一个排在前面。当相似度相差小于 $1e-10$ 时，认为相同。

```

int main(){
    → int top_k, num, i, j, k, woman[9], man[9], man_id[101], tmp_id;
    → double sim[101], tmp1, tmp2, tmp3, tmp_sim;
    → scanf("%d%d", &num, &top_k);
    → for(k = 1; k <= 8; k++){
    →     → scanf("%d", &woman[k]); //女生生辰八字
    →     for(i = 1; i <= num; i++){
    →         → scanf("%d", &man_id[i]); //男生身份证号
    →         → for(k = 1; k <= 8; k++){
    →             → scanf("%d", &man[k]); //男生生辰八字
    →             → tmp1 = 0; tmp2 = 0; tmp3 = 0;
    →             → for(k = 1; k <= 8; k++){
    →                 → tmp1 = tmp1 + woman[k] * man[k];
    →                 → tmp2 = tmp2 + woman[k] * woman[k];
    →                 → tmp3 = tmp3 + man[k] * man[k];
    →             → }
    →             → sim[i] = tmp1 / (sqrt(tmp2) * sqrt(tmp3)); //相似度
    →         }
    →     }
    →     for(j = 1; j <= num; j++){
    →         → for(j = 1; j <= num - i; j++){
    →             → if(sim[j] < sim[j + 1] || (fabs(sim[j] - sim[j + 1]) < 1e-10 && man_id[j] < man_id[j + 1])){
    →                 → tmp_id = man_id[j]; man_id[j] = man_id[j + 1]; man_id[j + 1] = tmp_id;
    →                 → tmp_sim = sim[j]; sim[j] = sim[j + 1]; sim[j + 1] = tmp_sim;
    →             }
    →         }
    →     }
    →     for(i = 1; i <= top_k; i++){
    →         → printf("%d", man_id[i]);
    →     }
    →     return 0;
}

```

#301 成绩统计

【问题描述】

期中考试后，老师需要对班里学生的成绩进行统计分析，了解各个分数段的人数。

给定所有学生的成绩和成绩分段方式，请你编程帮忙**以下任务**：

- 1) 统计各个分数段的人数，并按照分数段人数从大到小输出；
- 2) 将按分数段统计的结果用**模拟的直方图**显示出来。

【输入格式】

第 1 行包含 2 个整数 n 、 m 、 g ，分别表示学生人数、分数段个数、需解决的任务编号。

m 表示将成绩区间 $[0, 100]$ 平均分成 m 个分数段。例如 $m = 5$ 表示分为 5 个分数段，分别是 $[0, 20)$ 、 $[20, 40)$ 、 $[40, 60)$ 、 $[60, 80)$ 、 $[80, 100]$ 。

g 一共有 3 种取值，分别是 0、1、2，

- 0 表示任务 1、2 均需完成，且先完成任务 1，再完成任务 2；
- 1 表示只完成任务 1；
- 2 表示只完成任务 2。

第 2 行包含 n 个 $[0, 100]$ 之间的整数，为 n 个学生的期中考试成绩。

【输出格式】

任务 1 格式:

若干行，每行一个分数段以及该分数段的人数，分数段表示为 $[L,R)$ 或者 $[L,100]$ 的形式，其中 L 占 2 个字符的宽度， R 需要占用 3 个字符的宽度，分数段后面跟一个字符 $:$ ，之后输出一个空格，再输出该分数段的人数。

注意：如果该分数段的人数为 0，则不输出该分数段；按照分数段的人数从大到小的顺序输出，如果 2 个分数段人数相同，先输出分数段比较低的那一个。

任务 2 格式:

分数段的输出格式与任务 1 相同， $:$ 之后输出若干个字符 $*$ ，具体个数为该分数段的实际人数，或者该分数段人数归一化后的数量。人数最多的分数段能在显示在一行以内，归一化的方式为人数最多那个分数段最多输出 50 个字符 $*$ ，即如果人数多于 50，则输出 50 个字符 $*$ ，其他分数段按比例减少字符 $*$ 的个数（下取整）；否则直接输出实际数量的字符 $*$ 。

注意：任务 1 和任务 2 之间请输出一个空行。

【输入样例 1】

```
20 10 0.
88 90 75 68 77 85 86 99 100 95 60 55 32 93 63 60 78 96 70 80.
```

【输出样例 1】

```
[90,100]: 6.
[60, 70]: 4.
[70, 80]: 4.
[80, 90]: 4.
[30, 40]: 1.
[50, 60]: 1.

[ 0, 10]:.
[10, 20]:.
[20, 30]:.
[30, 40]:*.
[40, 50]:.
[50, 60]:*.
[60, 70]:****.
[70, 80]:****.
[80, 90]:****.
[90,100]:*****.
```

```

#include <stdio.h>
#include <math.h>
int main(){
    int n, m, g, i, j, sc[5001], a, b, c, d, t = 0, k[11] = {0}, ch[11][3];
    scanf("%d %d %d", &n, &m, &g);
    for (i = 0; i < n; i++)
        scanf("%d", &sc[i]);
    a = 100 / m;
    for (i = 0; i < n; i++) { // 每个分数
        for (j = 0; j < m; j++) // 每个分数段
            if (sc[i] >= j * a && sc[i] < (j + 1) * a)
                k[j] += 1;
        if (sc[i] == 100) k[m - 1] += 1;
    }
    for (i = 0; i < m; i++) {
        ch[i][0] = i;
        ch[i][1] = k[i];
    }
    for (i = 0; i < m; i++) // 统计结果排序
        for (j = 0; j < m; j++)
            if ((ch[i][1] > ch[j][1]) || (ch[i][1] == ch[j][1] && ch[i][0] < ch[j][0])) {
                c = ch[i][1]; ch[i][1] = ch[j][1]; ch[j][1] = c;
                d = ch[i][0]; ch[i][0] = ch[j][0]; ch[j][0] = d;
            }
}

```

```

if (g == 1) {
    for (i = 0; i < m; i++) {
        if (ch[i][1] != 0) {
            if (ch[i][0] == 0) {
                printf("[0, %d]", (ch[i][0] + 1) * a);
            }
            else {
                if ((ch[i][0] + 1) * a != 100) {
                    printf("[%d, %d]", ch[i][0] * a, (ch[i][0] + 1) * a);
                }
                else {
                    printf("[%d, 100]", ch[i][0] * a);
                }
            }
            printf(": %d", ch[i][1]);
            printf("\n");
        }
    }
}

```

```

if (g == 2) {
    if (ch[0][1] > 50) {
        for (j = 0; j < m; j++) {
            k[j] = k[j] * 50 / ch[0][1];
        }
    }
    for (i = 0; i < m; i++) {
        if (i == 0) {
            printf("[0, %d]", a);
        }
        else {
            if (i != m - 1) {
                printf("[%d, %d]", i * a, (i + 1) * a);
            }
            else {
                printf("[%d, 100]", i * a);
            }
        }
        printf(":");
        for (j = 0; j < k[i]; j++) {
            printf("**");
        }
        printf("\n");
    }
}

```

```

if (g == 0) {
    for (i = 0; i < m; i++) {
        if (ch[i][1] != 0) {
            if (ch[i][0] == 0) {
                printf("[0, %d]", (ch[i][0] + 1) * a);
            }
            else {
                if ((ch[i][0] + 1) * a != 100) {
                    printf("[%d, %d]", ch[i][0] * a, (ch[i][0] + 1) * a);
                }
                else {
                    printf("[%d, 100]", ch[i][0] * a);
                }
            }
            printf(": %d", ch[i][1]);
            printf("\n");
        }
    }
    printf("\n");
    if (ch[0][1] > 50) {
        for (j = 0; j < m; j++) {
            k[j] = k[j] * 50 / ch[0][1];
        }
    }
    for (i = 0; i < m; i++) {
        if (i == 0) {
            printf("[0, %d]", a);
        }
        else {
            if (i != m - 1) {
                printf("[%d, %d]", i * a, (i + 1) * a);
            }
            else {
                printf("[%d, 100]", i * a);
            }
        }
        printf(":");
        for (j = 0; j < k[i]; j++) {
            printf("**");
        }
        printf("\n");
    }
}

```

```

int n, m, g, i, j, sc[5001], a, b, c, d, t = 0, k[11] = {0}, ch[11][3];
int main() {
    void print_result();
    void print_figure();
    scanf("%d %d %d", &n, &m, &g);
    for (i = 0; i < n; i++)
        scanf("%d", &sc[i]);
    a = 100 / m;
    for (i = 0; i < n; i++) { // 每个分数
        for (j = 0; j < m; j++) // 每个分数段
            if (sc[i] >= j * a && sc[i] < (j + 1) * a)
                k[j] += 1;
        if (sc[i] == 100) k[m - 1] += 1;
    }
    for (i = 0; i < m; i++) {
        ch[i][0] = i;
        ch[i][1] = k[i];
    }
    for (i = 0; i < m; i++) // 统计结果排序
        for (j = 0; j < m; j++)
            if ((ch[i][1] > ch[j][1]) || (ch[i][1] == ch[j][1] && ch[i][0] < ch[j][0])) {
                c = ch[i][1]; ch[i][1] = ch[j][1]; ch[j][1] = c;
                d = ch[i][0]; ch[i][0] = ch[j][0]; ch[j][0] = d;
            }
    if (g == 1) print_result();
    if (g == 2) print_figure();
    if (g == 0) { print_result(); print_figure(); }
    return 0;
}

```

```

void print_result()
{
    for (i = 0; i < m; i++)
        if (ch[i][1] != 0) {
            if (ch[i][0] == 0)
                printf("[0, %d]", (ch[i][0] + 1) * a);
            else
                if ((ch[i][0] + 1) * a != 100)
                    printf("[%d, %d]", ch[i][0] * a, (ch[i][0] + 1) * a);
                else
                    printf("[%d, 100]", ch[i][0] * a);
            printf(": %d", ch[i][1]);
            printf("\n");
        }
}

void print_figure()
{
    if (ch[0][1] > 50)
        for (j = 0; j < m; j++)
            k[j] = k[j] * 50 / ch[0][1];
    for (i = 0; i < m; i++) {
        if (i == 0) printf("[0, %d]", a);
        else
            if (i != m - 1) printf("[%d, %d]", i * a, (i + 1) * a);
            else printf("[%d, 100]", i * a);
        printf(":");
        for (j = 0; j < k[i]; j++) printf("*");
        printf("\n");
    }
}

```

#290 购物车相似性

电商平台在购物狂欢节期间，从某类顾客中随机抽取 n 个顾客($n > 1$)，调查他们购物车中预选保存的商品类别，以掌握顾客购物偏好的共同性，得到最受欢迎的商品，为拓展市场销量服务。购物车中商品会被分类，假定种类是以整数进行编号，此问题为在 n 个集合中寻找交集的问题。

【输入格式】

第一行一个整数 n ，表示选取的顾客数。此后有 n 行，分别表示每位顾客所选商品构成，其中第一个数表示该顾客购物车中的不同商品数，其后为空格分隔的整数。调查抽取的顾客数 n 不超过 20，每个顾客所选的商品种类不超过 100。

【输出格式】

若不存在共同偏好，无交集，输出 NO；若有，把共同的商品编号从小到大输出，空格分隔。

【输入样例】

3

5 1 2 3 4 5

5 2 4 6 8 10

8 1 2 3 4 5 6 8 10

【输出样例】

2 4

统计排序类

集合运算



中國人民大學
RENMIN UNIVERSITY OF CHINA



3.3 数组字符

//输入 "345china" 或 "345<回车>china" 都正确

```
int a;  
char b[100];  
scanf("%d%s", &a, b);  
printf("a=%d b=%s", a, b);
```

//输入 "345china" 或 "345<回车>china" 都正确

```
int a;  
char b[100];  
scanf("%d%s", &a, b);  
printf("a=%d b=%s", a, b);
```

//输入 "345china" OK 但 "345<回车>china" 错误, gets获得空串

```
int a;  
char b[100];  
scanf("%d", &a);  
gets(b);  
printf("a=%d b=%s", a, b);
```

//输入china<回车> 可以有空格

```
char s[1000];  
gets(s);  
puts(s);
```

//输入 "1china" OK 但 "1<回车>china" 错误, gets获得空串

```
char a;  
char b[100];  
scanf("%c", &a);  
gets(b);  
printf("a=%c b=%s", a, b);
```

//输入china<回车> 可以有空格

```
char s[1000];  
gets(s);  
puts(s);
```

字符串到整数、浮点数

```
char s[100] = "123";  
double f;  
f = atof(s);  
printf("%lf\n", f);
```

atoi(p) 字符串转换到 int 整型

atof(p) 字符串转换到 double 浮点数

```
char s1[100] = "123";  
int n;  
n = atoi(s1);  
printf("%d\n", n);
```

atol(p) 字符串转换到 long 整型

```
char s2[100] = "123";  
long long int m;  
m = atol(s1);  
printf("%lld\n", m);
```

strncpy

4. strcpy和strncpy函数-字符串复制

- strcpy一般形式为：strcpy(字符数组1, 字符串2)
 - 作用是将字符串2复制到字符数组1中去
- 用strncpy函数将字符串2中前面n个字符复制到字符数组1中去

strncpy(字符数组1, 字符串2, n)

char str1[10],str2[]=" China" ;

str1=" China" ; 错误

str1=str2; 错误

char str1[10],str2[]="China";

strcpy(str1,str2);

```
#include<stdio.h>
#include<string.h>
int main(){
char name[]={ "Chinanet"},destin[20]={};
strncpy(destin,name,3);
printf("%s\n",destin);
}
```

str1	C	h	i	n	a	\0	\0	\0	\0	\0
------	---	---	---	---	---	----	----	----	----	----

#110 回文判断

```
int main()
{
    int n, j, i;
    char a[1000];
    gets(a);
    n = strlen(a);
    for (i = 0, j = n - 1; i < (n + 1) / 2; i++, j--)
        if (a[i] != a[j])
            { printf("No"); break; }
    if (i == (n + 1) / 2) printf("Yes");
    return 0;
}
```

```
for (i = 0, j = n - 1; i < (n + 1) / 2; i++, j--)
    if (a[i] != a[j])
        { printf("No"); break; }
    if (i == (n + 1) / 2) printf("Yes");
```

#110 回文判断.cpp

“对称循环类” 题目

[209] → #461 · 回文数等式

回文数 x 是指正读和反读都一样的正整数，如 11，121，1221。

输入一个 k ($1 \leq k \leq 1000$)，打印所有不超过 k 的数 i ($1 \leq i \leq k$) 的平方是回文数 x 的等式，即 $i*i=x$ 。输出每行一个等式。

输入格式

一行，只有一个整数 k

输出格式

$i*i=x$

输入样例

30

输出样例

1*1=1

2*2=4

3*3=9

11*11=121

22*22=484

26*26=676

```
int int_rev(int x) {
    int y = 0;
    while (x) {
        y = y * 10 + x % 10; //反转的整数扩大10倍+原数的余数
        x /= 10;
    }
    return y;
}

int main() {
    int k;
    scanf("%d", &k);
    for (int i = 1; i <= k; i++)
        if (i * i == int_rev(i * i))
            printf("%d*%d=%d\n", i, i, i * i);
    return 0;
}
```

#288 字符串之差

```
#include <stdio.h>
#include <string.h>
int fun ( char s[], char t[]) {
    int i = 0;
    while (s[i] == t[i]) {
        if (s[i] == 0)
            return (0);
        i++;
    }
    return (s[i] - t[i]);
}

int main() {
    char s1[101], s2[101];
    gets(s1);
    gets(s2);
    printf("%d\n", fun(s1, s2));
    return 0;
}
```

```
int fun ( char *s, char *t) {
    while (*s == *t) {
        if (*s == 0)
            return (0);
        ++s;
        ++t;
    }
    return (*s - *t);
}

int main() {
    char s1[100], s2[100];
    gets(s1);
    gets(s2);
    printf("%d\n", fun(s1, s2));
    return 0;
}
```

#291 大整数加减法

```
void parseStrToIntArr(char str[], int a[], int size) {  
    for(int i=0; i<size; i++)  
        a[i]=str[size-1-i]-'0';  
}  
  
int bigIntMinus(int m[], int a[], int b[], int len) {  
    for(int i=0; i<len; i++)  
        { m[i]=a[i]-b[i];  
          if(m[i]<0) { a[i+1]--; m[i]+=10; }  
        }  
    while(len>1 && m[len-1]==0) len--;  
    return len;  
}  
  
int add(int s[], int a[], int b[], int len) {  
    for(int i=0; i<len; i++)  
        { s[i]=a[i]+b[i];  
          if(s[i]>9) { s[i+1]+=s[i]/10; s[i]%=10; }  
        }  
    if(s[len]) len++;  
    return len;  
}
```

```
int main() {  
    char ac[2000], bc[2000];  
    int an[2000]={0}, bn[2000]={0};  
    char op, s1=1, s2=1;  
    cin>>op;  
    cin>>ac;  
    cin>>bc;  
    if(ac[0]=='-')  
        { s1=-1;  
          strcpy(ac, &ac[1]); //将 ac 去除第一位,  
        }  
    if(bc[0]=='-')  
        { s2=-1;  
          strcpy(bc, &bc[1]);  
        }
```


#291 大整数加减法

```
· · int lena = strlen(ac);  
· · int lenb = strlen(bc);  
· ·  
· · parseStrToIntArr(ac, an, lena);  
· · parseStrToIntArr(bc, bn, lenb);  
· ·  
· · int len = (lena > lenb) ? lena : lenb;  
· ·  
· · if(op == '-') s2 *= -1;  
· · int sum[2018] = {0};
```

```
· if(s1 == s2)  
· {  
· · · len = add(sum, bn, an, len);  
· · · · if(s1 == -1) cout << "-";  
· }  
· else  
· · · if((len > lenb) || (len == lenb && strcmp(ac, bc) > 0))  
· · · · { len = bigIntMinus(sum, an, bn, len);  
· · · · · if(s1 == -1) cout << "-";  
· · · · else  
· · · · · { len = bigIntMinus(sum, bn, an, len);  
· · · · · if(s2 == -1) cout << "-";  
· · · · }  
· · for(int i = len - 1; i >= 0; i--)  
· · · · cout << sum[i];
```



中國人民大學
RENMIN UNIVERSITY OF CHINA



3.4 循环枚举

#492-4 加法表达式

给定一个数字字符串 S ，在其中添加 3 个加号使其形成一个加法表达式。遍历所有可以构成合法加法表达式的加号添加方案，计算这些合法加法表达式的值，并按从大到小的顺序输出。

例如：对于数字字符串 12345，添加 3 个加号可以构成的加法表达式包括：

$$1+2+3+45=51$$

$$1+2+34+5=42$$

$$1+23+4+5=33$$

$$12+3+4+5=24$$

按从大到小的顺序输出的结果是：51 42 33 24

【输入格式】

一行，包含一个只含有数字的字符串，长度不超过 15。

【输出格式】

一行，若干整数，依次是从大到小输出所有构造合法的加法表达式的计算结果，每 2 个整数之间用一个空格隔开。

【输入样例】

12345

【输出样例】

51 42 33 24

$$12+34+5$$

$$1+234+5$$

字符串类

枚举

- 人大附中有四位同学中的一位做了好事，不留名，表扬信来了之后，校长问这四位是谁做的好事。

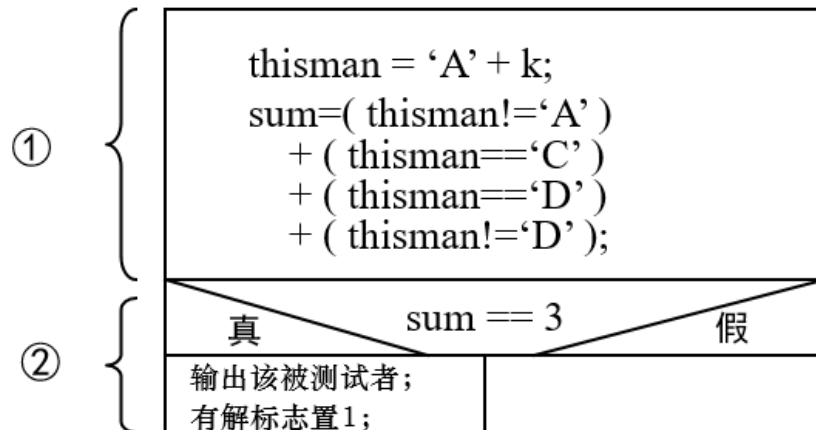
A说：不是我。

B说：是C。

C说：是D。

D说：他胡说。

已知三个人说的是真话，一个人说的是假话。现在要根据这些信息，找出做了好事的人。



```
for(k=0; k<4; k++)  
{thisman = 'A'+k;  
  sum =( thisman!='A' )  
        + ( thisman=='C' )  
        + ( thisman=='D' )  
        + ( thisman !='D' );  
  if (sum==3) {  
    printf( "%c\n ", thisman );  
    g=1;  
  }  
}
```

找出作案人

■ 某地刑侦队对涉及六个嫌疑人的一桩疑案进行案情分析：

- A、B至少有一人作案；
- A、D不可能是同案犯；
- A、E、F三人中至少有两人参与作案；
- B、C或同时作案,或与本案无关；
- C、D中有且仅有一人作案；
- 若D没参与，则E也不可能参与。

一脸懵逼

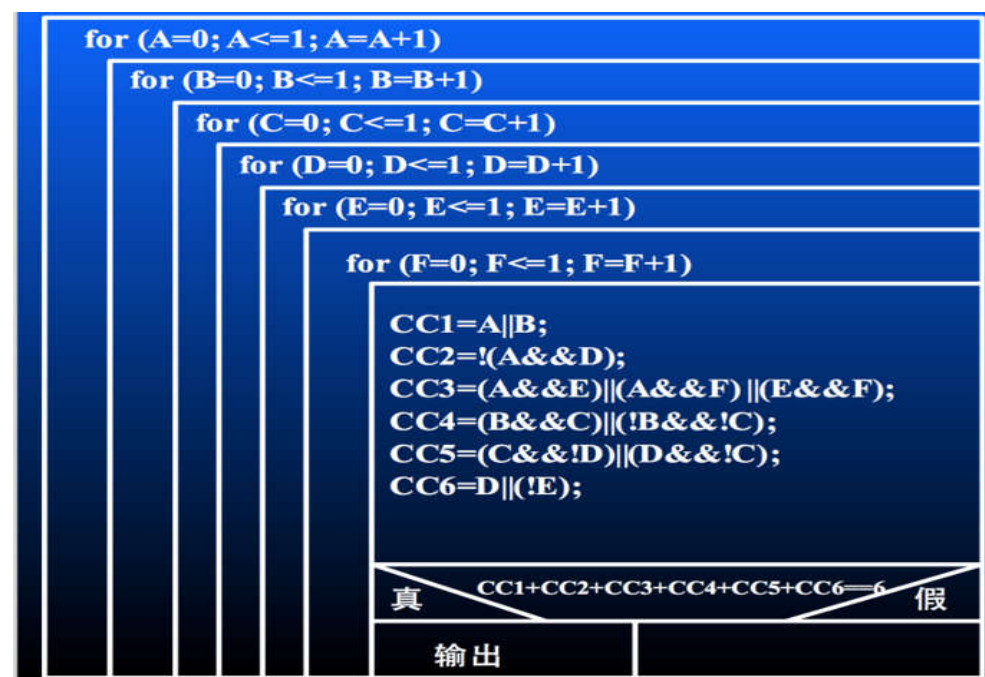


■ 已知：变量A和B分别表示A和B作案

- 怎么表示“A、B至少有一人作案”？
- 表达式：CC1 = A || B

■ 已知：变量A和D分别表示A和D作案

- 怎么表示“A、D不可能是同案犯”？
- 考虑相反事件：A和D是同案犯 → A && D
- 表达式：CC2 = !(A && D)



#147 教室排课

```
int main()
{
    int a,b,c,d,i,j,k,h;
    int s[8]={120,40,85,50,100,140,70,100};
    scanf("%d%d%d%d",&a,&b,&c,&d);
    int m=0;
    for(i=0;i<8;i++)
        for(j=0;j<8;j++)
            for(k=0;k<8;k++)
                for(h=0;h<8;h++)
                    if(a<=s[i]&&b<=s[j]&&c<=s[k]&&d<=s[h]&&i!=j&&j!=k&&k!=h&&i!=k&&i!=h&&j!=h)
                        {printf("%d%d%d%d\n",i+1,j+1,k+1,h+1); m=m+1;}
    if(m==0) printf("-1");
    return 0;
}
```

#308 猪场分配

```
for(i=0;i<Farm_Count;i++){  
    → if(rongliang[i]<P1_Num) continue;  
    → for(j=0;j<Farm_Count;j++){  
        → if(rongliang[j]<P2_Num||j==i) continue;  
        → for(k=0;k<Farm_Count;k++){  
            → if(rongliang[k]<P3_Num||k==j||k==i) continue;  
            → cost_sum=base_cost[i]+base_cost[j]+base_cost[k];  
            → cost_sum=cost_sum+each_cost[i]*P1_Num+each_cost[j]*P2_Num+each_cost[k]*P3_Num;  
            → find=0;  
            → if(cost_sum<min_cost) → find=1;  
  
            → else if(cost_sum==min_cost)  
            → if(ID[i]<P1_id) → find=1;  
            → else if(ID[i]==P1_id)  
            → if(ID[j]<P2_id) → find=1;  
            → else if(ID[j]==P2_id && ID[k]<P3_id) → find=1;  
            → if(find==1){  
                → min_cost=cost_sum;  
                → P1_id=ID[i];P2_id=ID[j];P3_id=ID[k];  
            }  
        }  
    }  
}
```



中國人民大學
RENMIN UNIVERSITY OF CHINA



3.5 递归搜索

两种

- 递归计算

- 递推式计算
- 重复计算问题

- 递归搜索

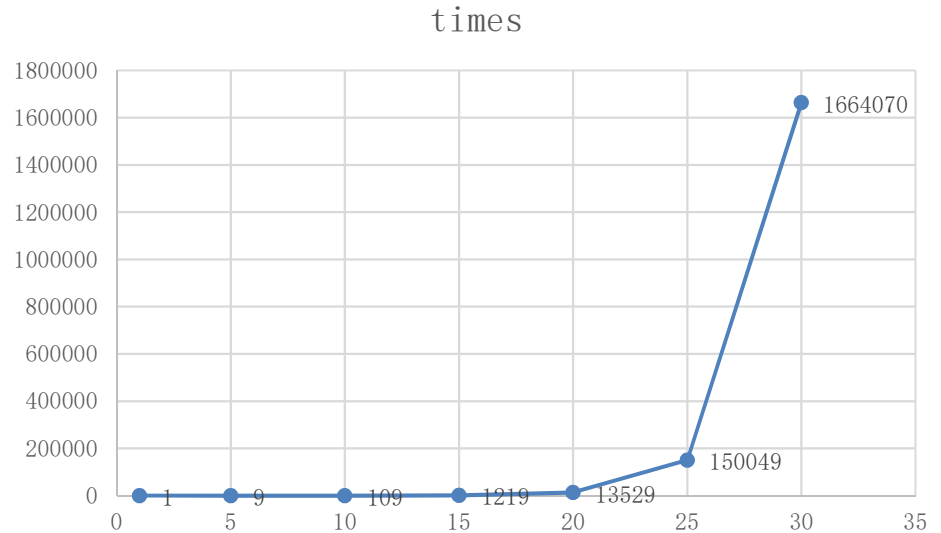
- 输出详细结果
- 输出最大值
- 输出方案数

斐波那契数列

```
#include<iostream>
using namespace std;

int times = 0;
int fib(int n) {
    times ++;
    if (n==1||n==2) return 1;
    return fib(n-1)+fib(n-2)+fib(n-3);
}

int main () {
    cout << fib(6) << endl;
    cout << "times: " << times << endl;
}
```



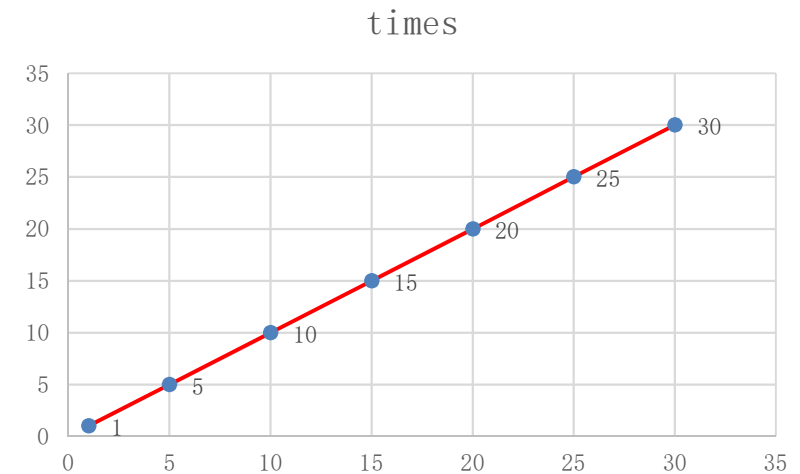
指数级增长!

Fibonacci.cpp

斐波那契数列

```
#include<iostream>
using namespace std;
int times = 0;
int fib(int n, int memo[]) {
    if (memo[n]!=-1)
        return memo[n];
    else{
        times ++;
        if (n==1||n==2) memo[n]=1;
        else memo[n]=fib(n-1,memo)+fib(n-2,memo);
        return memo[n];}
}
int main () {
    int memo[100];
    cout << fib(5,memo) << endl; for (int i = 0; i < 100; i ++) memo[i]=-1;

    cout << "times: " << times << endl;
}
```



```
int zai(int x, int y)
```

```
{ int topleft, topright;
```

```
  if (x==1) return a[x][y];
```

```
  else
```

```
    if(y==1)
```

```
    {  
      return a[x][y]+zai(x-1,y)  
    }
```

```
  else
```

```
  {  
    左上: a[x][y]+zai(x-1,y-1);  
    右上: a[x][y]+zai(x-1,y)  
    return max(左上、右上)  
  }
```

```
}
```

```
for(i=2;i<=tier;i++){  
  for(j=1;j<=i;j++){  
    if(a[i-1][j-1]>a[i-1][j]) num=a[i-1][j-1];  
    else num=a[i-1][j];  
    a[i][j]+=num;  
  }  
}
```

纯数组

#117 摘桃子

纯递归

计算超时！！

```
int zai(int x, int y) {
```

```
    int topleft, topright;
```

```
    if (taozi[x][y] == -1) {
```

```
        if (x == 1)
```

```
            taozi[x][y] = a[x][y]
```

```
        else if (y == 1)
```

```
            taozi[x][y] = a[x][y] + zai(x-1,y)
```

```
        else {
```

```
            左上 :  $a[x][y] + zai(x-1, y-1);$ 
```

```
            右上 :  $a[x][y] + zai(x-1, y)$ 
```

```
            taozi[x][y] = 比较 左上、右上
```

```
        }
```

```
    }
```

```
    return taozi[x][y];
```

```
}
```

```
int taozi[101][101];
```

-1

#117 摘桃子

分三种情况

直接或间接

告诉答案

递归+数组

递归求二进制表示位数

给定一个十进制整数，返回其对应的二进制数的位数。例如，输入十进制数9，其对应的二进制数是1001，因此位数是4。

$$f(0)=0$$

$$f(n)=1+f(n/2)$$

#278 爬楼梯

假如你正在爬一个有 n 个台阶的楼梯，一次只能爬1个台阶或者2个台阶，请问一共有多少种爬到顶端的走法？

```
int f(int n)
{
    if(n==1) return 1;
    if(n==2) return 2;
    return f(n-1)+f(n-2);
}
```

瓷砖铺放

有一长度为 N ($1 \leq N \leq 10$) 的地板，给定两种不同瓷砖：一种长度为1，另一种长度为2，数目不限。要将这个长度为 N 的地板铺满，一共有多少种不同的铺法？

例如，长度为4的地面一共有如下5种铺法：

$$4 = 1 + 1 + 1 + 1$$

$$4 = 2 + 1 + 1$$

$$4 = 1 + 2 + 1$$

$$4 = 1 + 1 + 2$$

$$4 = 2 + 2$$

void Try(int i)

{

是否到站

for (int j=1; j<=N; j++) // 逐个每一种情况

{ **if 第j位置是否可放**

标记占第j位置

Try(i+1)

释放第j位置

}

}

搜索

void Try(int i)

{ for (int j=1; j<=N; j++) // 逐个试每一种情况

{ if 第j位置是否可放

标记占第j位置

Y: 是否到站

N: Try(i+1)

释放第j位置

}

搜索

if 第j位置是否可放

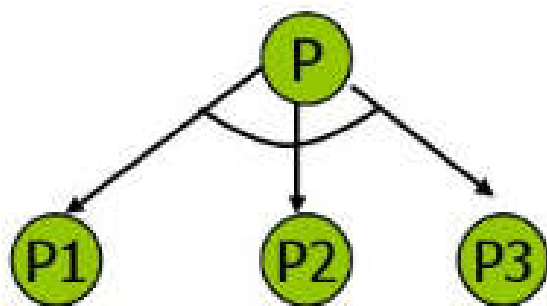
```
if (不能放) continue;  
else
```

```
if ( ) continue;
```

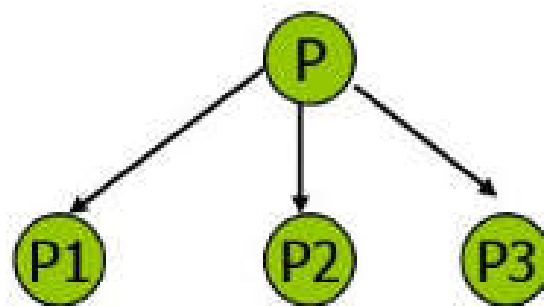
```
if ( 能放 )
```

与或图

- 与图: 把一个原问题分解为若干个子问题, P_1, P_2, P_3, \dots 可用“与图”表示; P_1, P_2, P_3, \dots 对应的子问题节点称为“与节点”。
- 或图: 把一个原问题变换为若干个子问题, P_1, P_2, P_3, \dots 可用“或图”表示; P_1, P_2, P_3, \dots 对应的子问题节点称为“或节点”。



与



或

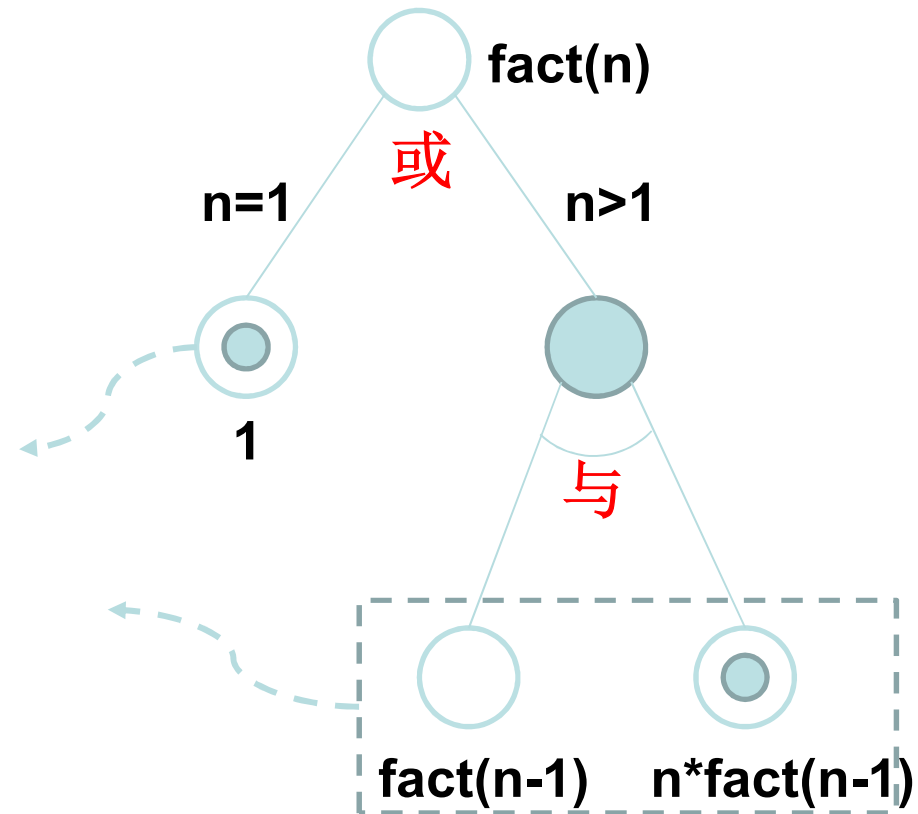
与或图与递归程序编写

```
#include<iostream>

using namespace std;

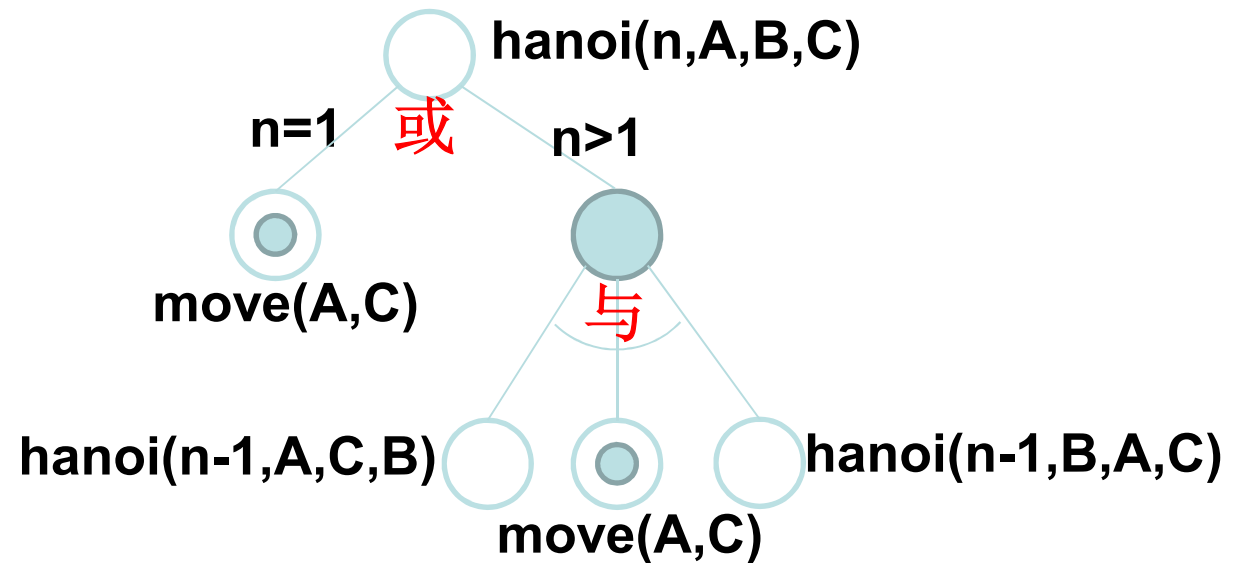
int fact (int n) {
    if (n==1) return 1;
    else {
        int fn1 = fact(n-1);
        return n*fn1;
    }
}

int main () {
    cout << fact(3);
}
```



Fact(阶乘).cpp

汉诺塔



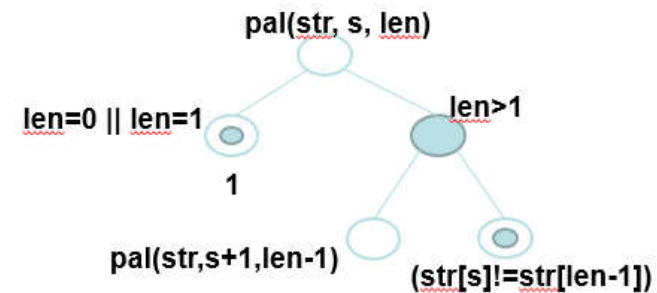
```
void move (int n, char A, char B, char C) {  
    if (n == 1)  
        cout << "move from " << A << " to " << C << endl;  
    else {  
        move (n-1, A, C, B);  
        cout << "move from " << A << " to " << C << endl;  
        move (n-1, B, A, C);  
    }  
}  
  
int main () {  
    int n = 4;  
    char A = 'A', B = 'B', C = 'C';  
    move (n, A, B, C);  
}
```

#123 汉诺塔.cpp

回文判断

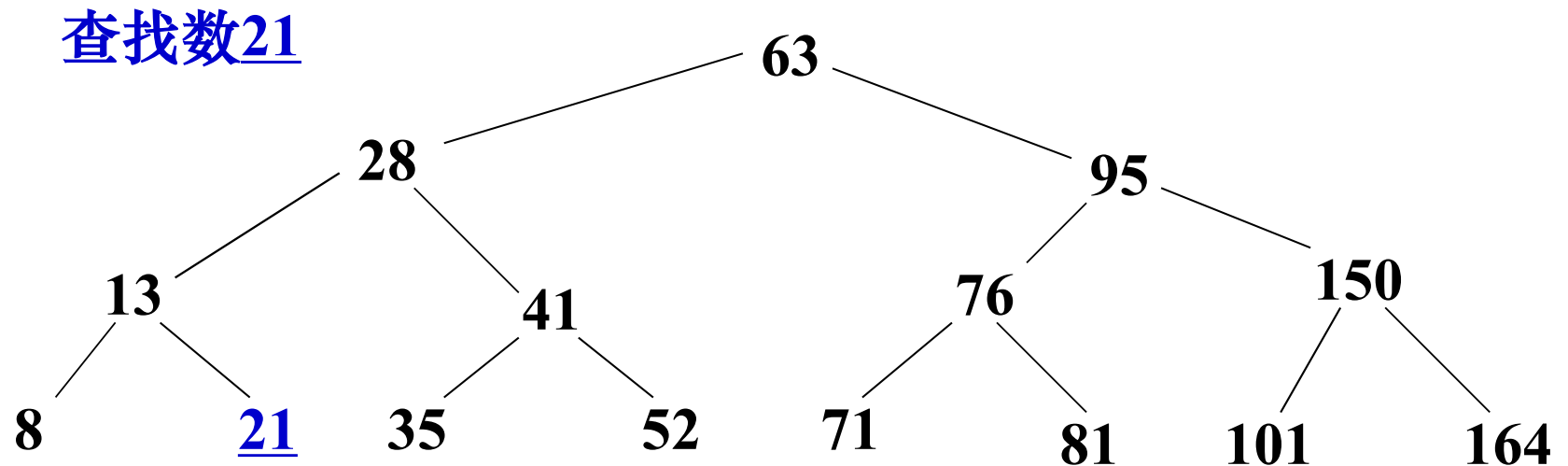
```
int main()
{
    int n,j,i;
    char a[1000];
    gets(a);
    n=strlen(a);
    for ( i = 0,j=n-1; i < (n+1)/2; i++,j--)
        if (a[i]!=a[j])
            { printf("No"); break;}
    if (i==(n+1)/2) printf("Yes");
    return 0;
}

int pal(char str[], int low, int high) {
    if (high <= low)
        return 1; // base case
    else if (str[low] != str[high])
        return 0;
    else
        return pal(str, low + 1, high - 1);
}
```



二分查找

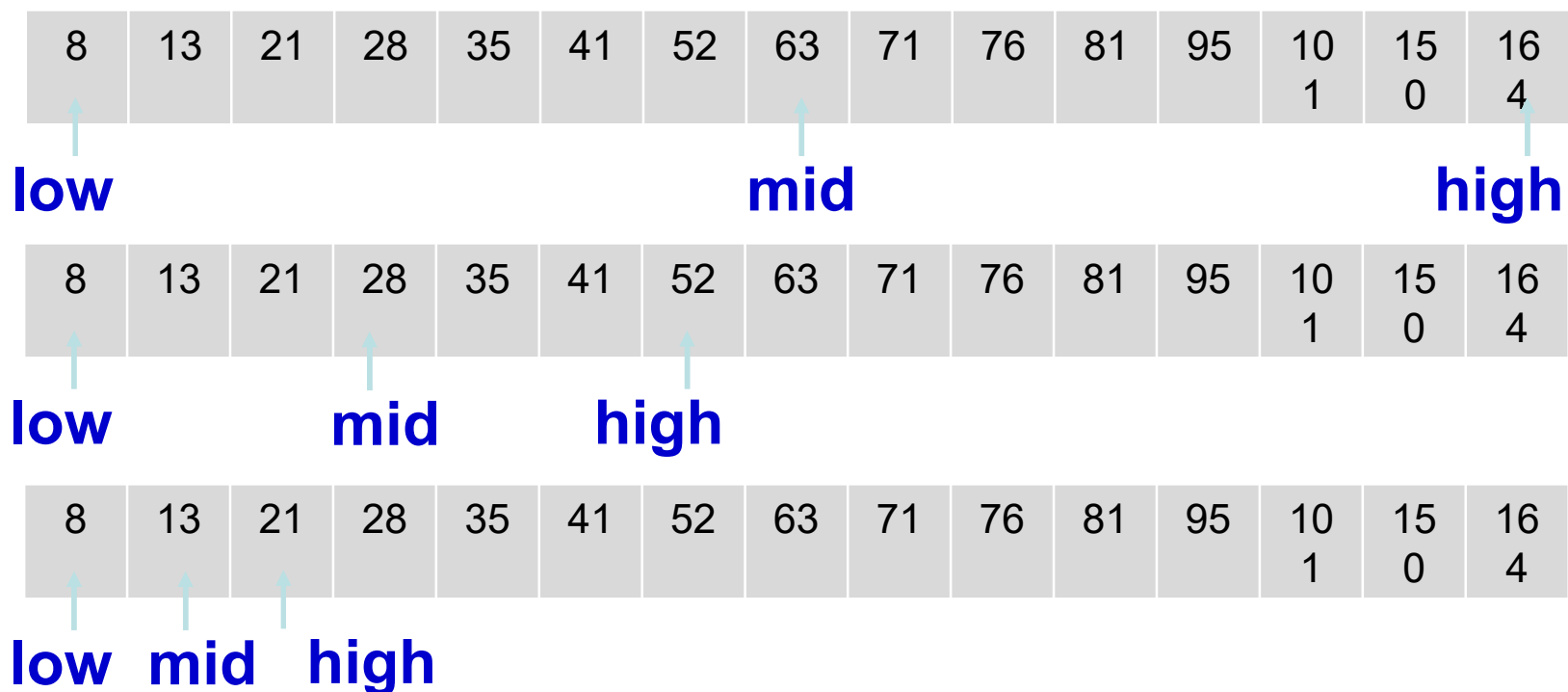
- 如果把排好序的数据看成一棵树.....



思想： 折半查找

二分查找的循环实现 (1)

- 核心难点：实现 “折半”
 - 解决：设置下标变量low, high, mid



循环方式

递归方式

Diagram 1: Initial array split. The array is [8, 13, 21, 28, 35, 41, 52, 63, 71, 76, 81, 95, 101, 150, 164]. The pivot is 63. The low segment is [8, 13, 21, 28, 35, 41, 52] and the high segment is [71, 76, 81, 95, 101, 150, 164].

Diagram 2: Low segment split. The low segment [8, 13, 21, 28, 35, 41, 52] is split with pivot 28. The low segment is [8, 13, 21] and the high segment is [35, 41, 52].

Diagram 3: Low segment split. The low segment [8, 13, 21] is split with pivot 13. The low segment is [8] and the high segment is [21].

```
int bsearch(int ary[], int low, int high, int m) {
    if ( low > high )           //没找到
        return -1;
    int mid = (low + high) / 2;
    if (ary[mid] == m) //找到
        return mid;
    else if ( ary[mid] > m ) //找左半边
        return bsearch(ary, low, mid - 1, m);
    else //找右半边
        return bsearch(ary, mid + 1, high, m);
}
```

Bsearch(二分查找).

Bisearch(二分查找).cpp

快速排序函数设计

```
void quicksort(int ary[], int low, int high) {
```

```
if (low >= high) return;
```

```
int p = partition(ary, low, high);
```

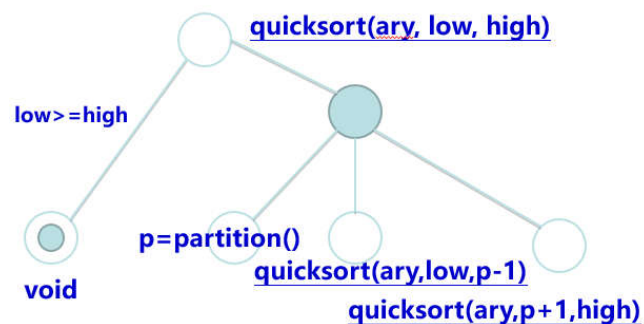
```
//递归调用：处理左侧分区
```

```
quicksort(ary, low, p - 1);
```

```
//递归调用：处理右侧分区
```

```
quicksort(ary, p + 1, high);
```

```
}
```



```
int partition(int ary[], int low, int high)
```

```
{
```

```
int p = ary[low];
```

```
while( low < high ) {
```

```
while( low < high && ary[high] >= p ) high--;
```

```
ary[low] = ary[high];
```

```
while( low < high && ary[low] <= p ) low++;
```

```
ary[high] = ary[low]; }
```

```
ary[low] = p;
```

```
return low;
```

```
}
```

全排列

```
void perm_impl1 (int ary[ ], int selected[ ], int k, int n) {
```

```
    if (k == n) {
```

```
        for (int i = 0; i < n; i ++)
```

```
            cout << selected[i] << " ";
```

```
        cout << endl; }
```

```
    else
```

```
        for (int i = k; i < n; i ++)
```

```
            selected[k] = ary[i];
```

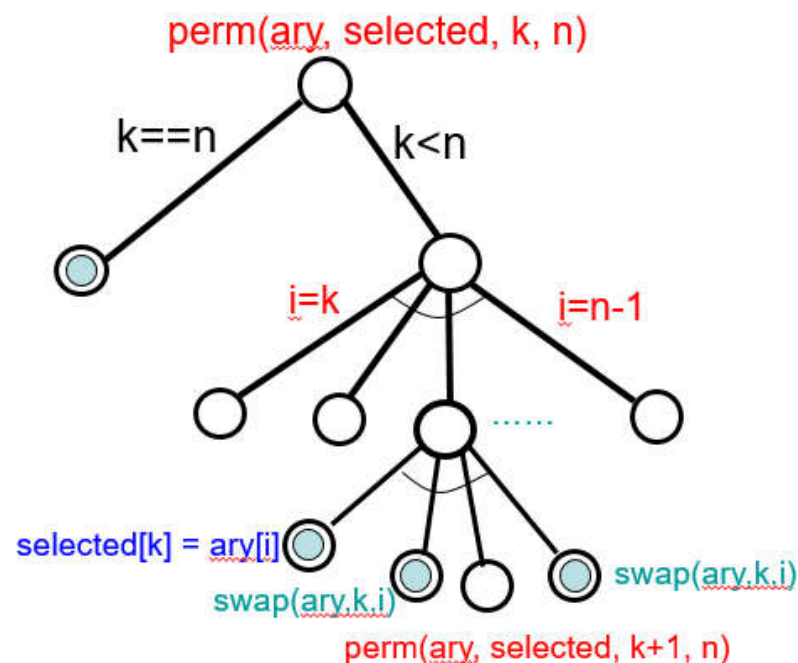
```
            swap (ary, k, i);
```

```
            perm_impl1(ary, selected, k+1, n);
```

```
            swap (ary, k, i);
```

```
        }
```

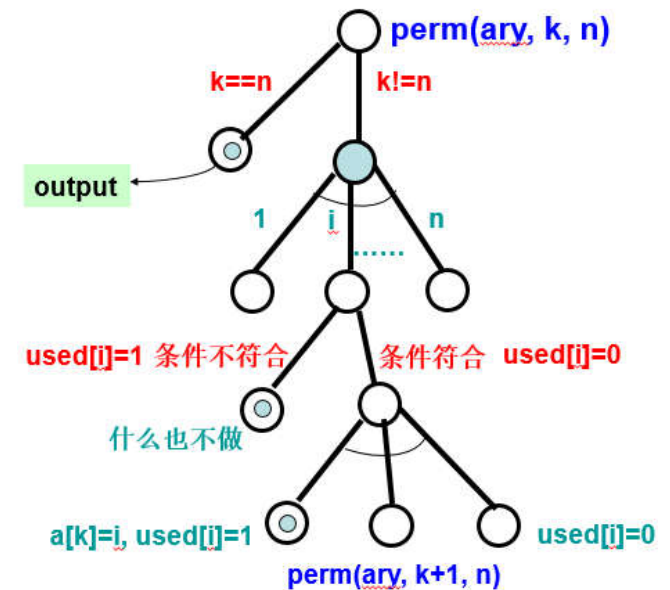
```
    }
```



#109 全排列问题.cpp

全排列2

```
void perm_impl2 (int ary[ ], int indices[ ], int used[ ], int k, int n) {  
    if (k == n) {  
        for (int i = 0; i < n; i ++)  
            cout << ary[indices[i]] << " ";  
        cout << endl;    }  
    else  
        for (int i = 0; i < n; i ++){  
            if (used[i] == 1) continue;  
  
            used[i] = 1;  
            indices[k] = i;  
            perm_impl2(ary, indices, used, k+1, n);  
            used[i] = 0;  
        }  
}
```



#320 物体称重

有 n 种砝码，它们的标准重量为 w_1, w_2, \dots, w_n 克，相应的数量分别为 c_1, c_2, \dots, c_n 。砝码标准重量和数量可灵活组合，但已知它们的总重不会超过 10000 克。现有 m 个不同重量的物体需称重，请根据给定的砝码标准和数量，1) 确定每个重物被准确称量时，可采用的砝码组合方案数；2) 以及所有方案中，砝码最少用量是多少。

这里的组合方案指不同标准的砝码各有多少个，不区分具体是哪些砝码组合而成。比如，假设物体重 10 克，用 2 个 5 克的砝码是一种方案，用 5 个 2 克的是另一种方案，用 1 个 10 克的则是砝码用量最少的方案。

【输入格式】

共 4 行，第 1 行两个数，分别为标准砝码的种类数 n 和物体数量 m ；

第 2 行， n 个整数，分别为砝码的标准重量；

第 3 行， n 个整数，与第 2 行不同标准砝码对应，依次为它们的数量；

第 4 行， m 个整数，分别为 m 个待称重物体的重量，无重复。

所有数据均为空格分隔。

5 5

1 2 3 4 5 砝码重量

5 4 3 2 1 砝码数量

4 5 6 7 8 待称物体

【输出格式】

根据物体称重的砝码组合方案数从多到少排序，逐行输出每个物体的重量及组合出该重量的砝码组合方案数；若组合数相同，按物体重量从小到大排。

8:16,2

7:12,2

6:9,2

5:7,1

4:5,1

//探索第OBJ_ID个物体，而且还剩下重量为NeedToWeight，可以用的砝码
NoUsed_FM_ID，方案数存储在Solution_Num[OBJ_ID]

```
void Try(int NeedToWeight, int OBJ_ID, int NoUsed_FM_ID) {
```

```
    int NoUsed_FM_Weight = 0;
```

```
    for (int i = NoUsed_FM_ID; i < n; i++)
```

```
        NoUsed_FM_Weight += FaMa[i][0] * FaMa[i][1];
```

```
    if (NeedToWeight == 0) k = Solution_Num[OBJ_ID]++;
```

```
    else if (NeedToWeight <= NoUsed_FM_Weight)
```

```
        for (int i = NoUsed_FM_ID; i < n; i++)
```

```
            if (NeedToWeight - FaMa[i][0] >= 0 && FaMa[i][1] > 0) {
```

```
                FaMa[i][1]--;
```

```
                Try(NeedToWeight - FaMa[i][0], OBJ_ID, i);
```

```
                FaMa[i][1]++;    }
```

```
}
```

主函数: **for (i = 0; i < m; i++)**
Try(Object[i], i, 0);

递归搜索

优化:

砝码按重量从大到小排序

递归计算（动态规划）

```
int FM_wei[10],FM_num[10]; //砝码重量、数量
```

```
int FangAn[15001] = {0};
```

根据方案数的递推关系

```
FangAn[0] = 1;
```

```
//每增加一种砝码，方案数重新计算
```

无法输出具体方案

```
for (i = 0; i < n; i++)
```

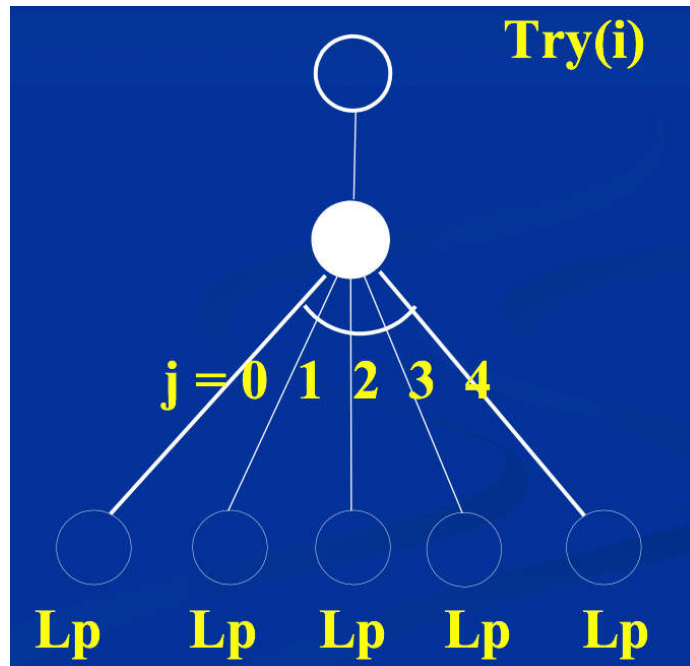
```
    for (w = total_weight; w >= 0; w--)
```

```
        for (k = 1; k <= min(FM_num[i], w / FM_wei[i]); k++)
```

```
            FangAn[w] += FangAn[w - k * FM_wei[i];
```


分书问题

- 定义函数 Try(i), 表示给第i个人分书
- 画出与或图



```
void Try(int i) {  
    for (int j = 0; j < BNUM; j++) {
```

给第i个人
分配第j本书

```
    }  
}
```

```

void Try(int i) {
    if (book[j] == 1 || like[i][j] == 0) continue;
    if (book[j] != 1 && like[i][j] != 0) { }

    for (int j = 0; j < BNUM; j++) { // for each book
        if (book[j] == 1) continue; // already taken
        if (like[i][j] == 0) continue; // not like

        take[i] = j; // take the book
        book[j] = 1; // update the flag

        if (i < PNUM - 1 )
            Try(i+1);
        else {
            n++;
            cout << "Assignment Plan #" << n << endl;
            for(int k=0; k < PNUM; k++)
                cout << "Person " << char(k+'A') << " takes Book " << take[k] << endl;
            cout << endl;
        }

        take[i] = -1; // return the book
        book[j] = 0; // update the flag
    }
}

```

给第i个人
分配第j本书

边搜索，边判断

```

void Try(int i) {
    if (i == PNUM) {
        n++;
        cout << "Assignment Plan #" << n << endl;
        for(int k=0; k < PNUM; k++)
            cout << "Person " << char(k+'A') << " takes Book " << take[k] << endl;
        cout << endl;
    } else

        for (int j = 0; j < BNUM; j ++) { // for each book
            if (book[j] == 1) continue; // already taken
            if (like[i][j] == 0) continue; // not like

            take[i] = j; // take the book
            book[j] = 1; // update the flag
            Try(i+1);
            take[i] = -1; // return the book
            book[j] = 0; // update the flag
        }
}

```

先Base, 后搜索

#202 分书问题.cpp

八皇后问题

```
void Try(int i)
```

```
{ for (int j=1; j<=8; j++) // 逐个试每一列
```

```
{ if (如果可以放在第j列)
```

第i个皇后
放在第j列

```
}
```

```
}
```

```

#include <stdio.h>    // 预编译命令
const int Normalize = 9;    // 定义常量, 用来统一数组下标
int Num;                // 整型变量, 记录方案数
int q[9];                // 记录8个皇后所占用的列号
int C[9];                // C[1]~C[8], 布尔型变量, 当前列是否安全
int L[17];                // L[2]~L[16], 布尔型变量, (i-j)对角线是否安全
int R[17];                // R[2]~R[16], 布尔型变量, (i+j)对角线是否安全

```

```

void Try(int i)                // 被调用函数

```

```

{  int j;                // 循环变量, 表示列号

```

Try(1);

```

    int k;                // 临时变量

```

```

    for (j=1; j<=8; j++)    // 循环

```

```

    {  if ( C[j]==1 && R[i+j]==1 && L[i-j+Normalize]==1 )

```

```

        // 表示第i行, 第j列是安全的
    }
}

```

```

{   q[i] = j;           // 第一件事, 占用位置(i,j)
    C[j] = 0;           // 修改安全标志
    L[i-j+Normalize] = 0;
    R[i+j] = 0;
    if ( i<8 )           // 第二件事, 判断是否放完8个皇后
        Try(i+1);
    else
    {
        Num++;
        printf("方案%d: ", Num);
        for ( k=1; k<=8; k++)
            printf("%d ", q[k]);
        printf ("\n" );
    }
    C[j] = 1;           // 第三件事, 修改安全标志, 回溯
    L[i-j+Normalize] = 1;
    R[i+j] = 1;
}

```

第i个皇后
放在第j列

```

void Try(int i) {
    int j, k;
    if (i == 9) {                // 已经放完8个皇后
        Num++;                    // 方案数加1
        printf("方案%d: ", Num); // 输出方案号
        for (k = 1; k <= 8; k++)
            printf("%d ", q[k]); // 输出具体方案
        printf("\n"); }
    else
        for (j = 1; j <= 8; j++) // 循环
            if ( C[j] == 1 && R[i + j] == 1 && L[i - j + 9] == 1 ) {
                q[i] = j; // 第一件事, 占用位置(i,j)
                C[j] = 0; // 修改安全标志, 包括所在列和两个对角线
                L[i - j + Normalize] = 0;
                R[i + j] = 0;
                Try(i + 1); // 则继续放下一个
                C[j] = 1; // 第三件事, 修改安全标志, 回溯
                L[i - j + Normalize] = 1;
                R[i + j] = 1;
            }
}

```

Try(1);

最大岛屿

```
#include <stdio.h>
int land[101][101] = {0};
int area = 0;
int m, n;
int max_area = 0;
```

```
for (i = 0; i < n; i++)
for (j = 0; j < m; j++)
    if (land[i][j] == 1) {
        area = 0;      Try(i, j);
        if (area > max_area ) max_area = area;
        count++;
    }
```

```
void Try(int x, int y) {
```

```
    if (land[x][y]==1) {
        area++;
        land[x][y] = 0;
        if (y < m - 1 && land[x][y + 1] ) Try(x, y + 1);
        if (y > 0 && land[x][y - 1])      Try(x, y - 1);
        if (x < n - 1 && land[x + 1][y])    Try(x + 1, y);
        if (x > 0 && land[x - 1][y])        Try(x - 1, y);
        if (x < n - 1 && y < m - 1 && land[x + 1][y + 1]) Try(x + 1, y + 1);
        if (x > 0 && y > 0 && land[x - 1][y - 1] )      Try(x - 1, y - 1);
        if (x > 0 && y < m - 1 && land[x - 1][y + 1])    Try(x - 1, y + 1);
        if (x < n - 1 && y > 0 && land[x + 1][y - 1])    Try(x + 1, y - 1);
    }
```

area--; 不能写!!!

```
}
```


只探最大面积

```
void Try(int x, int y) {
```

```
    area++; used[x][y] = 1;
```

```
    if (area > max) max = area;
```

```
    if (y < m - 1 && land[x][y + 1] ) Try(x, y + 1);
```

```
    if (y > 0 && land[x][y - 1]) Try(x, y - 1);
```

```
    if (x < n - 1 && land[x + 1][y]) Try(x + 1, y);
```

```
    if (x > 0 && land[x - 1][y]) Try(x - 1, y);
```

```
    if (x < n - 1 && y < m - 1 && land[x + 1][y + 1]) Try(x + 1, y + 1);
```

```
    if (x > 0 && y > 0 && land[x - 1][y - 1] ) Try(x - 1, y - 1);
```

```
    if (x > 0 && y < m - 1 && land[x - 1][y + 1]) Try(x - 1, y + 1);
```

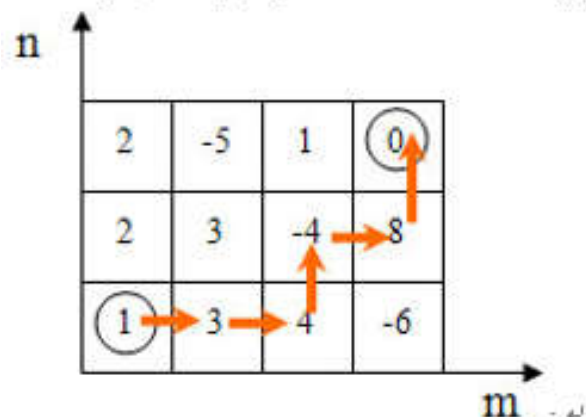
```
    if (x < n - 1 && y > 0 && land[x + 1][y - 1]) Try(x + 1, y - 1);
```

```
    area--; used[x][y] = 0;
```

```
}
```

#126 Travel

有一个 $n \times m$ 的棋盘，如图所示，骑士 X 最开始站在方格(1,1)中，目的地是方格(n,m)。他的每次都只能移动到上、左、右相邻的任意一个方格。每个方格中都有一定数量的宝物 k （可能为负），对于任意方格，骑士 X 能且只能经过最多 1 次（因此从(1,1)点出发后就不能再回到该点了）。



你的任务是，帮助骑士 X 从(1,1)点移动到(n,m)点，且使得他获得的宝物数最多。

输入格式

□□ 输入共有 $n+1$ 行。

□□ 第一行为两个整数 n 和 m ($1 \leq n, m \leq 8$)。

□□ 接下来 n 行，每行有 m 个整数，每 2 个整数之间由一个空格分隔。第 $i+1$ 行第 j 个整数表示方格(i, j)中的宝物数目 k ($-100 \leq k \leq 100$)。

输出格式

□□ 输出数据仅一个整数，即为骑士获得的宝物数。

void Try(int x, int y) {

0、排队越界+不可用

● if (**x < 1 || x > n || y < 1 || y > m || used[x][y]**) return;

● mine += value[x][y];
used[x][y] = 1;

1、先用

Try(1, 1) ?

if (**x == n && y == m**) // 当前探索完毕

if (sum < mine) sum = mine;

2.1、终点到站

else {

Try(x + 1, y); // 上

Try(x, y - 1); // 左

Try(x, y + 1); // 右

2.2、直接探索

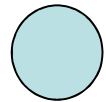
● mine -= value[x][y];
used[x][y] = 0;

3、回溯

}

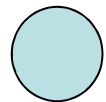
void Try(int x, int y) {

Try(1, 1)



```
mine += value[x][y];  
used[x][y] = 1;
```

1、先用



```
if ( x == n && y == m )
```

2.1、终点到站

```
if (sum < mine) sum = mine;
```

```
else {
```

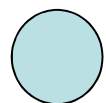
2.2、判断探索

```
if (x + 1 <= n && !used[x + 1][y]) Try(x + 1, y);
```

```
if (y - 1 >= 1 && !used[x][y - 1]) Try(x, y - 1);
```

```
if (y + 1 <= m && !used[x][y + 1]) Try(x, y + 1);
```

```
}
```



```
mine -= value[x][y];  
used[x][y] = 0;
```

3、回溯

}

```
void Try(int x, int y) {
```

```
    if (x == n - 1 && y == m - 1)
```

```
        { find = 1; //判断条件
```

```
          return;    }
```

```
for (i = 0; i < R; i++)  
for (j = 0; j < C; j++)  
    search(i, j, 0);
```

#122 迷宫

不回朔

```
used[x][y] = 0; //使用 这一点标记搜索过
```

分别探索四个方向

条件：不会出界、是道路

```
if (y != 0 && used[x][y - 1] == 1)    search(x, y - 1);
```

```
if (y != m - 1 && used[x][y + 1] == 1)    search(x, y + 1);
```

```
if (x != 0 && used[x - 1][y] == 1)    search(x - 1, y);
```

```
if (x != n - 1 && used[x + 1][y] == 1)    search(x + 1, y);
```

```
}
```



```
void walk(int x, int y, int goal) {
```

#437 寻找一卡通

```
    if (a[x][y] == goal)
```

```
        if (step < min_distance) min_distance = step;
```

```
    else if (used[x][y] == 0) {
```

```
        step++; used[x][y] = 1;
```

先用

探索
四个
方向

```
        if (y < n - 1 && a[x][y + 1] != 1) walk(x, y + 1, goal);
```

```
        if (x < n - 1 && a[x + 1][y] != 1) walk(x + 1, y, goal);
```

```
        if (y > 0 && a[x][y - 1] != 1)      walk(x, y - 1, goal);
```

```
        if (x > 0 && a[x - 1][y] != 1)      walk(x - 1, y, goal);
```

```
        step--; used[x][y] = 0;
```

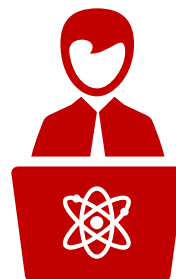
回溯

```
    }
```

```
}
```



中國人民大學
RENMIN UNIVERSITY OF CHINA



谢谢大家!

