



计算机/软工相关专业 考研复试指导

2020 第一版

- 1.机试百题（取自 34 所高校近年真题）
- 2.面试指南（专业面&英语面一本解决）
- 3.只要不打老师基本都是稳的！

主编：灰灰



前言

首先皮皮灰恭喜大家顺利通过初试的独木桥，进入复试！

复试的成绩的好坏直接决定了你能否考上研究生以及学业奖学金的等级（第一年的学业奖学金通常由考研总成绩决定）。现在离成功只有一步之遥，我们切不可掉以轻心：

（1）复试占比区间一般在 50%-20%之间。

（2）复录比一般为 1.2（12 个人进复试刷掉 2 人），但热门学校复录比皮皮灰看到最高的可以达到 4: 1，另国家规定必须差额复试，这就意味着总会有人被刷掉。

（3）非全日制大行其道，导致分数线一降再降。学校变了法子的让尽可能多的人参加复试，推荐分数不够的人调剂非全日制，甚至分数低的组面试一进去就问你愿不愿意调剂非全日制。

（4）100%的学校都有着复试成绩不合格不予录取的规定。

复试流程通常由体检（查血、胸透）、政治审查（不计入总成绩）、心理测试（不计入总成绩）、机试（部分学校没有该项）、笔试、面试组成，只要有一项不合格均算不合格。

本书针对复试的各个环节进行全方法的复习指导规划，意在给各位小皮皮提供一个精准有方向的复试准备。

（1）建议复习开始准备时间：1 月初

（2）参加复试前需要准备的物品：

外省的同学建议提前预定宾馆，规划好交通路线。

| | | | |
|-----|--------------------|--------------------|----------------|
| 应届生 | 准考证 考完试别丢了 | 有效身份证件 | 复试志愿申请表 |
| | 学生证 | 本科成绩单 需要有学校的章 | 政审表 辅导员能解决 |
| | 1 寸照片若干 体检用 | 现金 200+ 复试要交现金的 | |
| 往届生 | 准考证 考完试别丢了 | 有效身份证件 | 复试志愿申请表 |
| | 毕业证和学位证书 原件，复印件 | 本科成绩单 需要有学校的章 | 政审表 档案所在地盖章 |
| | 1 寸照片若干 体检用 | 现金 200+ 复试要交现金的 | |

(3) 体检相关

体检主要查血、胸透和血压。

查血：查转氨酶，肝好不好，一查便知。

如果转氨酶超出正常范围过多，需要在学校指定的学院复查。

对自己肝不放心的同学，可以事先吃药降酶（小葵花护肝片）。

关于乙肝的问题：国家规定，学校不得以学生有乙肝为依据拒绝入学。

胸透：主要查有没有阴影。

血压：高血压什么自己控制点==。

对于有重大疾病影响入学的同学，可以申请休学，学校保留一年入学资格。

(4) 心理测试

通常是在微信上回答几百个选择题，别乱选就好。

(5) 机试

建议使用语言：**C++**

如果评分的是老师，一般会给你几个测试用例，看你程序输出的结果。时间复杂度与空间复杂度通常不在考虑的范围之内，只考虑算法的正确性。

编程完成后不要急着叫老师评分，花点时间对每个变量与关键点写注释，评分过程中老师很可能会看你的源程序，这有可能成为扣分的点。如果实在想不出来，**切记不要交空的东西**，最后给老师看一份写满注释的程序也是会有辛苦分的。

如果评分的是 OJ 系统，请尽可能给出最优解提交。

本书前篇为**机试指导**：由上机题要点、C++在上机中的应用，机试精选 100 题（题目精选自 34 所 985 高校近年试题）组成。

(6) 面试

综合面试在 20 分钟左右（部分学校会严格计时）

面试的老师人数为 5 人以上，通常包含专业面试老师、英语面试老师、面试记录人员等。

通常的顺序为英语面试->专业面试

按教育部要求，所有考生面试将进行全程录像录音。

参加面试时，考生可提供反映能力与水平的获奖证书、各类证明等相关材料。

几乎所有的学校都存在着：面试不及格不予录取。

本书后篇为**面试指导**：包含家常扯淡篇，英语面试篇，专业面试常考题目汇总。

只要不打老师基本都是稳的

(7) 政治审查表

XX 大学 2018 年研究生复试录取政审表

| | | | | |
|---|--|--------|------------|---|
| 姓 名 | 皮皮灰 | 性 别 | 皮 |  |
| 民 族 | 汉 | 出生日期 | 1997/12/06 | |
| 政治面貌 | 群众 | 入党时间 | 无 | |
| 宗教信仰 | 无 | 联系电话 | / | |
| 报考院系 | 计算机学院 | Email | / | |
| 本科毕业时间、院校及专业 | 2017 年 XX 大学软件工程专业 | | | |
| 硕士毕业时间、院校及专业 | 无 | | | |
| 是否拥护党的路线方针政策 | <input checked="" type="checkbox"/> 是 <input type="checkbox"/> 否 | | | |
| 是否参加过非法组织或活动 | <input type="checkbox"/> 是 <input checked="" type="checkbox"/> 否 | | | |
| 是否受过违法违纪处分 | <input type="checkbox"/> 是 <input checked="" type="checkbox"/> 否 | | | |
| <p>思想政治表现自述：</p> <p>我在大学本科学历学习期间用心上进，遵纪守法，无任何违法违纪行为，品学兼优，多次获得各类奖项，诚实守信，没有任何不良行为；担任过 XXX 等职务，对社会工作尽职尽责，并于 XX 年加入了光荣的中国共产党。我期望 XX 大学这个人才荟萃的大熔炉里，能够继续创造佳绩，早出成绩、早日成才，早日为国家和社会贡献自己的才智和力量。</p> <p style="text-align: right;">签 名：皮皮灰 2018 年 3 月 3 日</p> | | | | |
| 以下由单位填写（暂无工作学习单位的，由户籍所在地村委、居委会党组织或档案保管单位填写） | | | | |
| <p>思想政治品德表现情况：</p> <p>该生在各方面表现突出，思想上用心进取，认真学习马克思列宁主义、毛泽东思想、邓小平理论，政治思想觉悟高，学习刻苦发奋，成绩优秀。曾获二等奖学金、“三好学生”称号，用心参加各项活动，全面发展，用心参加社会实践活动，群众基础较好。</p> | | | | |
| 以上政审情况属实。 | | | | |
| 负责人签名：XXX | | 党组织公章： | | 2018 年 3 月 3 日 |

备 注： 请用黑色水笔填写政审表内容，以便今后存档，而且所有栏目不能为空，没有则填“无”；

目录

| | |
|-----------------------|----|
| 1. 机试篇 | 1 |
| 1.1 推荐使用 C++ | 1 |
| 1.1.1 基础篇 | 1 |
| 1.1.2 常见算法问题 | 3 |
| 1.2 机试 100 题 | 7 |
| 1.2.1 中南大学上机题 | 7 |
| 1.2.2 北京理工大学上机题 | 13 |
| 1.2.3 东华大学上机题 | 18 |
| 1.2.4 福州大学复试上机题 | 23 |
| 2. 综合面试 | 31 |
| 2.1.家常扯淡篇 | 32 |
| 2.2 专业面试常考问题汇总 | 36 |
| 2.2.1 C 语言 | 37 |
| 2.2.2 数据结构 | 41 |
| 2.2.3 操作系统 | 41 |
| 2.2.4 计算机网络 | 41 |
| 2.2.5 计算机组成原理 | 42 |
| 2.2.6 数据库 | 49 |
| 2.2.7 软件工程 | 49 |
| 2.2.8 软件测试 | 49 |
| 2.2.9 面向对象 | 49 |
| 2.2.10 UML | 49 |
| 2.2.11 离散数学 | 49 |
| 2.2.12 编译原理 | 49 |
| 2.2.13 其他问题 | 49 |

1. 机试篇

1.1 推荐使用 C++

为什么选择 C++? 相比于 c 或者 Java 有什么好处?

(1) 在一般的情况下, C++ 一般比 Java 的运行速度快上许多, 若某些学校的程序有 Limit time 的限制, C++ 是一个不错的选择;

(2) C++ 简单而功能强大, 有些东西用 C++ 实现起来更为方便, 如指针的操作。

(3) 现在对 C++ 提前了解会对以后的工作有好处, 以后去公司上手也会比较快。

(4) C/C++ 的编辑器很多, 且支持性较好, 若程序中存在 bug, 可以借助编辑器强大的 DeBug 功能来快速解决问题。

(5) 学会 C/C++ 以后的工作选择的可能性更多, 可以去搞硬件, 也可以去取搞软件。

C++ 相对于 C, 可以调用更多的函数, 比如常见的 sort 函数, 调用可以直接对所想要的数组进行排序, 而不用自己去实现底层; 同时 C++ 引入了类和对象的概念, 采用封装、继承、多态, 程序的可用性, 安全系更好。

1.1.1 基础篇

1. C++ 版 Hello World!

```
#include <iostream>
using namespace std;
int main(void){
    cout << "Hello World!" << endl;
    return 0;
}
```

2. 基本的输入输出

1) 常用的输入输出函数

C++ 中输入输出首先要加上头文件 `#include <iostream>`, 至于 `using namespace std;` 就不用管它的意思, 当成固定格式写上就行了。

注: 在 C++ 中也可以使用 C 语言中的库函数, 但是写法略有不同。

例如: 在 C 语言中调用 `math.h`, `#include <math.h>`, 而在 C++ 中要调用时的写法是 `#include <cmath>`。基本上就是去掉 `.h` 后缀, 在前面加上 `c` 即可。

C++ 中的标准输出函数:

例: `cout << x1 << x2 << endl;` // `x1`、`x2` 是变量名, `endl` 表示一行结束, 会进行换行

通过 `cout` 可以输出变量、常量等, 而不用像 C 语言中 `printf()` 一样写一长串。

C++ 中的输入函数:

例: `cin >> x1 >> x2;` // 输入变量 `x1`、`x2`

注: 当进行字符串读入时, 若要连空格一起读入要采用以下方式:

a) `string str;`

`getline(cin, str);`

b) `char str2[1024];`

`cin.getline(str2, 1024);` // 第二个参数 1024 为要读取的字符串的长度

2) 带格式控制符的输出函数

举例介绍:

```
#include <iostream>
#include <iomanip>
using namespace std;
void main() {
    float f1 = 2.99;
    float f2 = 8.90909;
    int i = 10;
    cout << setfill('*');
    cout << setw(10) << f1 << endl;
    cout << setw(10) << f2 << endl;
    cout << setw(10) << i << endl;
}
```

其中 `setw` 是设置域宽，就是输出数据所占几列，如果在默认的情况下，输出的数据不能达到所规定的域宽就用空格填充，当然也可以用 `setfill` 来设置填充物，如上例中设置的就是用 “*” 来填充空的列。在使用操纵符时，需要包含头文件 `iomanip`。

需要注意的是，所有的操纵符除了 `setw` 之外，对流的影响都是永久的，除非用户再次进行设置。而 `setw` 默认是 `setw(0)`，每次使用完 `setw` 之后，域宽又被重新置为 0，这就需要在每一次使用时都进行设置。

如上例中，3 次输出就设置了 3 次，而 `setfill` 却只设置了一次。`setw` 还有一点需要说明，如果 `setw` 设置的域宽小于实际数据的域宽时，仍然按照数据的实际宽度来显示

补充:

`setprecision` 用来设置浮点数的精度，默认精度为 6 位。

`left` 和 `right` 用来控制左对齐和右对齐。

`showpoint` 用来强制显示小数点及全部尾部的 0。

`fixed` 强制浮点数以定点格式输出。

这部分读者可以自己根据编辑器尝试得到结果。

说明：如果对于 C++ 的带格式输出函数不熟悉的话，那么也可以直接采用自己熟悉的 C 的输出函数，如 `printf(“%5d”, u)` 等照样能够得到想要的答案，皮皮灰的观点就是不惜一切的得分。

3. 结构体

结构体在 C 语言中就有的，但是前面 C 语言笔试部分中没有进行介绍。

结构体定义方式:

```
struct Student{
    int id;
    char name[20];
};
```

定义结构体变量:

```
struct Student s;
```

更常见的一种方式是使用 typedef 给结构体起一个别名，使操作更方便：

```
typedef struct Student{
    int id;
    char name[20];
}Student;
```

定义的时候就可以更简洁了：

```
Student s;
```

4. 关于字符串

在 C 语言中，字符串的结束标志是 '\0'，我们可以通过下标来逐一访问字符串中的字符：

```
int i = 0;
while(str[i] != '\0'){
    cout << str[i] << endl;
}
```

5. 关于数字和字符之间的转换

字符都是以 ASCII 码进行存储的，'0'到'9'的 ASCII 码是逐个递增的，将数字字符和'0'作差就可以得到对应的数字。同理，将'0'加上对应的数字，就可以得到该数字对应的字符。

例如，将字符 '7' 转换成数字 7：

```
char c = '7';
int num = c - '0';
```

例如，将数字 5 转换成字符 '5'：

```
int num = 5;
char c = '0' + 5;
```

1.1.2 常见算法问题

1. 排序问题

排序的算法有多种，复习数据结构的时候大家肯定也掌握了很多种，因为在大多数机试中并没有时间的要求，只要能运行出正确的结果即可，所以这里就以最简单的冒泡排序为主。

冒泡排序：(以非递减为例)

```
for(int i = 0; i < n; i++){    //第 i 趟排序找出第(i+1)大的元素
    for(int j = 0; j < n-i-1; j++){
        if(num[j] > num[j+1]){
            //交换第 j 和第 j+1 个元素
        }
    }
}
```


2. 进制转换

将十进制转换成二进制是最常见的了，这里以十进制转换成 8 进制为例：

```
char result[100]; //结果通过字符数组来存储，存储是逆序的
int length = 0; //记录结果的长度
while(num/8 != 0){
    result[length] = num%8;
    length++;
    num /= 8;
}
result[length] = num;
length++;
```

3. 质数的判定

1) 直观判断法

最直观的方法，根据定义，因为质数除了 1 和本身之外没有其他约数，所以判断 n 是否为质数，根据定义直接判断从 2 到 $n-1$ 是否存在 n 的约数即可。C++代码如下：

```
bool isPrime_1( int num ){
    int tmp =num- 1;
    for(int i= 2;i <=tmp; i++)
        if(num %i== 0)
            return 0 ;
    return 1 ;
}
```

2) 直观判断法改进

上述判断方法，明显存在效率极低的问题。对于每个数 n ，其实并不需要从 2 判断到 $n-1$ ，我们知道，一个数若可以进行因数分解，那么分解时得到的两个数一定是一个小于等于 \sqrt{n} ，一个大于等于 \sqrt{n} ，据此，上述代码中并不需要遍历到 $n-1$ ，遍历到 \sqrt{n} 即可，因为若 \sqrt{n} 左侧找不到约数，那么右侧也一定找不到约数。C++代码如下：

```
bool isPrime_2( int num ){
    int tmp =sqrt( num);
    for(int i= 2;i <=tmp; i++)
        if(num %i== 0)
            return 0 ;
    return 1 ;
}
```

4. 常见的字符串函数介绍

函数名: strcpy

功 能: 将参数 src 字符串拷贝至参数 dest 所指的地址

用 法: char *strcpy(char *dest, const char *src);

返回值: 返回参数 dest 的字符串起始地址

说 明: 如果参数 dest 所指的内存空间不够大, 可能会造成缓冲溢出的错误情况, 在编写程序时需特别留意, 或者用 strncpy() 来取代;

实例:

```
#include <stdio.h>
#include <string.h>
int main(void) {
    char string[10];
    char *str1 = "abcdefghi";

    strcpy(string, str1);
    printf("%s\n", string); // 输出: abcdefghi
    return 0;
}
```

函数名: strcat

功 能: 字符串拼接函数

用 法: char *strcat(char *dest, const char *src);

返回值: 返回 dest 字符串起始地址

说 明: strcat() 会将参数 src 字符串复制到参数 dest 所指的字符串尾部;

dest 最后的结束字符'\0' 会被覆盖掉, 并在连接后的字符串的尾部再增加一个'\0';

dest 与 src 所指的内存空间不能重叠, 且 dest 要有足够的空间来容纳要复制的字符串;

实例:

```
#include <string.h>
#include <stdio.h>
int main(void) {
    char destination[25];
    char *blank = " ", *c = "C++", *Borland = "Borland";
    strcpy(destination, Borland);
    strcat(destination, blank);
    strcat(destination, c);
    printf("%s\n", destination); // 输出: Borland C++
    return 0;
}
```

函数名: strcmp

功 能: 字符串比较

用 法: int strcmp(const char *s1, const char *s2);

返回值: 根据 ASCII 码比较, 若参数 s1 和 s2 字符串相同则返回 0, s1 若大于 s2 则返回大于 0 的值, s1 若小于 s2 则返回小于 0 的值

说 明: 它是区分大小写比较的, 如果希望不区分大小写进行字符串比较, 可以使用 stricmp 函数

实例:

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

int main(void)
{
    char *a = "aBcDeF";
    char *b = "AbCdEf";
    char *c = "aacdef";
    char *d = "aBcDeF";
    printf("strcmp(a, b) : %d\n", strcmp(a, b)); // 输出: 1
    printf("strcmp(a, c) : %d\n", strcmp(a, c)); // 输出: -1
    printf("strcmp(a, d) : %d\n", strcmp(a, d)); // 输出: 0
    return 0;
}
```

函数名: strlen

功 能: 计算指定的字符串 s 的长度, 不包括结束字符 '\0'

用 法: size_t strlen(const char *s);

返回值: 返回字符串 s 的字符数

说 明: strlen() 函数计算的是字符串的实际长度, 遇到第一个 '\0' 结束;

如果你只定义没有给它赋初值, 这个结果是不定的, 它会从首地址一直找下去, 直到遇到 '\0' 停止;

sizeof 返回的是变量声明后所占的内存数, 不是实际长度, 此外 sizeof 不是函数, 仅仅是一个操作符, strlen() 是函数;

实例:

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

int main() {
    char str[5] = "abcd";
    printf("strlen(str)=%d, sizeof(str)=%d\n", strlen(str), sizeof(str));
    // 输出: strlen(str)=4, sizeof(str)=5
    return 0;
}
```

1.2 机试 100 题

1.2.1 中南大学上机题

1. Problem Description

大家都关心考试的难易程度。K 老师出题有一个规律，在出题之前，他会随机写下一个字符串，只要在这个字符串中能按顺序找到 E, A, S, Y 四个字母，他出题就会比较简单。你拿到了字符串，请你告诉别人题目难不难吧。

Input

输入的数据有多组，每组占一行，由一个字符串组成（字符串的长度不超过 1000）。

Output

对于每组输入数据，输出一行，对应一个要求的答案（题目简单就输出 easy，难就输出 difficult）

Sample Input

```
eAsy
SEoAtSNY
```

Sample Output

```
difficult
easy
```

算法思想：由于要比较的内容已经确定为EASY，那么可以另外设计一个辅助字符串，里面存储的即是这四个字母，设计两个指针i, j，指针i指向给定的字符串，指针j指向辅助的字符串，当配时，指针i, j均向后移动一位，则

（1）若i指向了给定了字符串的尾部，而j未指向尾部，那么输出difficult；

（2）若j指向了辅助字符串的尾部，则输出easy。

在本题中，我们加入一个输入的字符串的个数来统计随机的个数

Code:

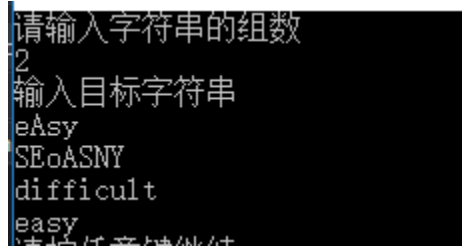
```
#include <iostream>
#include <string>
using namespace std;
bool match(char a[]) {
    int i = 0, j = 0;
    char tmp[] = { 'E', 'A', 'S', 'Y' };
    while (i < 1000 && j < 3) {
        if (a[i] == tmp[j]) {
            i++;
            j++;
        }
        else
            i++;
    }
    if (i >= 1000)
        return false;
    else
```

```

        return true;
    }
    int main() {
        cout << "请输入字符串的组数" << endl;
        int n = 0;
        cin >> n;
        //输入字符串数组
        cout << "输入目标字符串" << endl;
        char str[][1000] = {'0'};
        for (int i = 0; i < n; i++)
            cin >> str[i];
        for (int i = 0; i < n; i++) {
            if (match(str[i]))
                cout << "easy" << endl;
            else
                cout << "difficult" << endl;
        }
        system("pause");
    }
}

```

运行结果：



```

请输入字符串的组数
2
输入目标字符串
eAsy
difficult
SEoASNY
easy
请输入字符串的组数

```

2. 在太平洋的一个小岛上，岛民想要建立一个环岛的堤坝，我们可以将小岛简化为一个二维平面，你需要使用 K 条边（这些边要么是水平或者垂直长度为 1 的边，要么是倾斜 45 度的长度为 $\sqrt{2}$ 的边）围城一个多边形，多边形的顶点必须位于整点，然后要让围成的多边形面积最大，你要求出最大面积是多少。

Input

输入包含多个测试实例，每组实例给出一个数 K ($3 \leq k \leq 2,000,000,000$)

Output

每一组对应一个要求的答案（保留一位小数）

Sample Input

3

4

5

6

Sample Output

0.5

2.0

2.5

4.0

算法思想：不要死死的抓住小岛的面积不放，看清题意怎么围成的。怎样保留一位小数？

点睛 1：抓清题意，看看边的条数与边长的要求；

点睛 2：看看多边形顶点放置的要求。

本题的关键：通过画图的方式，找到边数与所画的最小的三角形的个数之间的关系，是主要考察分析能力，代码较为简单。

$$(k-3)/2=a\dots b$$

若 $b=0$, 则 $s=(1+4a)*0.5$

若 $b\neq 0$, 则 $s=(1+4a+3)*0.5$

Code:

```
#include<iostream>
#include<algorithm>
#include<math.h>
using namespace std;
int main() {
    int k, x, y;
    double s = 0;
    while (scanf_s("%lld", &k) != EOF) {
        if (k < 3) {
            cout << "边数输入错误" << endl;
            break;
        }
        x = (k-3)/ 2;
        y = (k-3)% 2;
        if (y == 0) {
            s = (1 + 4 * x)*0.5;
            cout << "面积最大值为: ";
            printf("%.1lf\n", s);
        }
        else {
            s = (1 + 4 * x+3)*0.5;
            cout << "面积最大值为: ";
            printf("%.1lf\n", s);
        }
    }
    return 0;
}
```

运行结果：

```
3
面积最大值为: 0.5
4
面积最大值为: 2.0
5
面积最大值为: 2.5
6
面积最大值为: 4.0
```

3. Problem Description

鲁大师和他的朋友们经常去一家奇怪的餐厅,为什么说奇怪呢,一是餐厅提供的菜品比较奇怪,二是餐厅的付费规则比较奇怪,每个人有不同的折扣率和折扣上限(单人从总价里折算的最高金额),超过折扣上限的原价付费。这次鲁大师和蔚然晨风以及朋友们一共N个人去这家餐厅吃饭,他们点的菜品总价是T,现在告诉你每个人的折扣率z和折扣上限H请告诉他们最少需要支付多少钱。

Input

输入数据有多组,每组占 N+1 行,第一行是 N 和 T,接下来 N 行,每行两个数字 z 和 H(输入数据保证最后结果为 int 型, $0 < N < 100$)

Output

对于每组输入数据,输出一行,对应一个要求的答案。

Sample Input

```
2 100
0.7 70
0.6 50
3 500
0.6 100
0.8 200
0.7 100
1 100
0.6 100
```

Sample output

```
65
390
60
```

算法思想:设计一个二维数组 a, 第一列 a[0] 存储折扣率, 第二列 a[1] 存储折扣上限. 将折扣率按从小到大进行排序; 下面进行分类讨论:

- (1) 若 $T < \text{最小折扣率的上限}$, 那么最后的价格直接是 $T * \text{折扣率}$;
- (2) 若不够, 则选择第二小的折扣率来计算, 看其折扣上限与第一个相加是否满足 $> T$, 若满足, 则将剩余的以第二小的折扣率付钱,
- (3) 若最后算了所有人的折扣上限加起来小于消费金额 T, 那么重复 (1) (2), 剩余的部分按原价付钱。

Code:

```
#include <iostream>
#include<algorithm>
using namespace std;
int compute(double a[][2],int n, int T){
    //若所有人折扣的金额加起来小于 T
    int sum1 = 0, sum=0, sum2=0;
    for (int i = 0; i < n; i++)
        sum1 += (int)a[i][1];
    if (sum1 > T) {
        for (int j = 0; j < n-1; j++) {
            T -= (int)a[j][1];
```

```

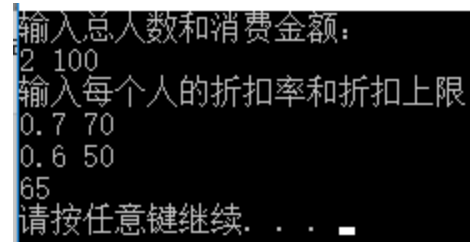
        if(T>0)
            sum += (int)(a[j][0] * a[j][1]);
        int i = j + 1;
        if(i<n&&T<a[i][1])
            sum += (int)(T * a[i][0]);
    }
}
else {
    for (int k = 0; k < n; k++)
        sum2 += (int)(a[k][1]*a[k][0]);
    T = T - sum1;
    sum = sum2 + T;
}
return sum;
}
int main() {
    cout << "输入总人数" << endl;
    int n = 0;
    scanf_s("%d", &n);
    double a[100][2]; //初始化
    cout << "输入每个人的折扣率和折扣上限" << endl;
    for (int i = 0; i < n; i++) {
        cin >> a[i][0];
        cin >> a[i][1];
    }
    int sum = 0;
    //按照折扣率大小进行排序, 在这里选择冒泡排序进行按折扣率大小进行排序,
    //即第1列进行排序
    for(int i=0;i<n-1;i++)
        for (int j = n - 1; j > i; j--)
            if (a[j - 1][0] > a[j][0]) { //若为逆序则交换
                double tmp = a[j - 1][0];
                a[j - 1][0] = a[j][0];
                a[j][0] = tmp; //交换第一列的值
                double tmp1 = a[j - 1][1];
                a[j - 1][1] = a[j][1];
                a[j][1] = tmp1; //交换第二列的值
            }
    for (int i = 0; i < n; i++) {
        cout << a[i][0] << ",";
        cout << a[i][1] << endl;
    }
    cout << "请输入消费的总金额 T: ";
    int T = 0;

```



```
    cin >> T;  
    sum = compute(a, n, T);  
    cout << sum << endl;  
    system("pause");  
    return 0;  
}
```

运行结果：



```
输入总人数和消费金额:  
2 100  
输入每个人的折扣率和折扣上限  
0.7 70  
0.6 50  
65  
请按任意键继续. . .
```

1.2.2 北京理工大学上机题

1. 身份证号的校验身份证号码共 18 位，最后一位是校验位 A[18] : aaaaaabbbbbbbcccd

校验的规则是如下

前十七位的权值分别是：W[17]:7 9 10 5 8 4 2 1 6 3 7 9 10 5 8 4 2

$x = (A[0]*W[0] + A[1]*W[1] + A[2]*W[2] + \dots + A[16]*W[16]) \bmod 11$

x 和校验位 y 的对应规则对应如下：

x:0 1 2 3 4 5 6 7 8 9 10

y:1 0 x 9 8 7 6 5 4 3 2

若 y 等于 d 则身份证号码正确

输出格式：aaaaaabbbbbbbcccd 正确

若 y 不等于 d 则身份证号码不正确

输出格式：应为:aaaaaabbbbbbbcccy

测试用例：拿自己身份证实验一下就知道了

例如：52242619811105565x

例如：533325199108247066

需要解决的问题：

问题 1：验证输入的身份证的位数，是否是 18 位

问题 2：身份证校验位的确定

Code:

```
#include <iostream>
using namespace std;
int main() {
    //输入权值数组
    int W[17] = { 7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2 };
    //校验位对照表
    char check[] = { '1', '0', 'x', '9', '8', '7', '6', '5', '4', '3', '2' };
    cout << "请输入您的 18 位身份证号：" << endl;
    char A[17] = { '0' };
    cin >> A;
    //验证输入的身份证号是否为 18 位
    int num = strlen(A);
    //cout << num << endl;
    if (num != 18)
        cout << "身份证号输入不完全，请再次输入。";
    //创建辅助数组，将前 17 位存到辅助数组中
    int B[17] = { 0 };
    for (int i = 0; i < 17; i++)
        B[i] = (int)A[i] - 48;

    //计算前 17 位的权值之和 MOD11
    int sum = 0;
    for (int j = 0; j < 17; j++)
        sum += W[j] * B[j];
```

```

int x = sum % 11;
int y = check[x] - 48;
if (check[x] == A[17]) {
    cout << A;
    cout << " " << "正确"<<endl;
    system("pause");
    return 0;
}
else {
    for (int i = 0; i < 17; i++)
        cout << B[i];
    if (check[x] == 'x')
        cout << 'x' << endl;
    else
        cout << y << endl;
    system("pause");
    return 0;
}
return 0;
}

```

运行结果:

```

请输入您的18位身份证号:
52242619811105565x
52242619811105565x 正确
请按任意键继续. . .

```

2. 二分查找{-36 -25 0 12 14 29 35 47 76 100}, 对上述十个数进行二分查找

测试用例:

请输入要查找的数据: 14

14 是第 5 个数, 查找次数为 1

请输入要查找的数据: -25

-25 是第 2 个数, 查找次数为 2

请输入要查找的数据: 121

查找失败

编者说: 写上机题的时候一定要分析学校的题型是怎么样的, 一定要相信, 基础算法也重要, 快速的写出此类题型才能更好的去写难题。

Code:

```

#include <iostream>
using namespace std;
//直接是二分查找的应用
int count_num = 0; //记录查找的次数
int binarySearch(int a[], int n, int key) {
    int low = 0, high = n - 1;
    while (low <= high) {
        count_num++;
    }
}

```

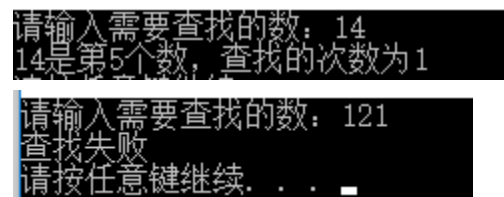
```

        int mid = (low + high) / 2;
        if (a[mid] == key)
            return mid;
        if (a[mid] > key)
            high = mid - 1;
        else
            low = mid + 1;
    }
    return -1;
}

int main() {
    int a[10] = {-36, -25, 0, 12, 14, 29, 35, 47, 76, 100 };
    cout << "请输入需要查找的数: ";
    int num = 0;
    cin >> num;
    int n = binarySearch(a, 10, num);
    if (n >= 0)
        cout << num << "是第" << (n+1) << "个数, 查找的次数为" << count_num <<
endl;
    else
        cout << "查找失败" << endl;
    system("pause");
    return 0;
}

```

运行结果:



```

请输入需要查找的数: 14
14是第5个数, 查找的次数为1

请输入需要查找的数: 121
查找失败
请按任意键继续. . .

```

3. 建立一个学生信息系统, 输入学生信息, 输出有挂科同学的信息, 再按照平均成绩从高到低排序输出

输入:

5

zhaoyi 70 80 90 240

qianer 65 32 77 174

sunsan 100 55 68 223

lisi 86 77 90 253

wangwu 100 59 66 225

输出:

*[qianer] 65 32 77

*[sunsan] 100 55 68

*[wangwu] 100 59 66

lisi 86 77 90
zhaoyi 70 80 90
wangwu 100 59 66
sunsan 100 55 68
qianer 65 32 77

Code:

```
#include<iostream>
using namespace std;
typedef struct {
    char name[20];
    float score1;
    float score2;
    float score3;
    int total;
}Student;
void swap1(Student s[], int i, int j){
    Student temp = s[i];
    s[i] = s[j];
    s[j] = temp;
}
void find_fail(Student s[],int n) {
    //输出不及格的人数
    for (int i = 0; i < n; i++) {
        if (s[i].score1 < 60 || s[i].score2 < 60 || s[i].score3 < 60)
            cout << "*" << s[i].name << " " << s[i].score1 << " " << s[i].score2
            << " " << s[i].score3 << endl;
    }
    cout << endl;
}

//按照平均成绩的高低输出
void printByAverage(Student s[],int n) {
    //按照基本的冒泡排序法按照成绩排序
    for (int i = 0; i < n - 1; i++)
        for (int j = n - 1; j > i; j--)
            if (s[j-1].total>s[j].total)//逆序
                swap1(s, j-1, j);
    for (int k = n-1; k >=0; k--) {
        cout << s[k].name << " ";
        cout << s[k].score1 << " ";
        cout << s[k].score2 << " ";
        cout << s[k].score3 <<endl;
    }
}
```

```
    }  
}  
int main() {  
    cout << "输入学生的人数: ";  
    int n = 0;  
    cin >> n;  
    Student s[100]; // 假定最多输入 100 个学生的成绩  
    cout << "请依次输入学生的姓名, 第一门课的成绩, 第二门课的成绩, 第三门课的成绩以及总分: " << endl;  
    for (int i = 0; i < n; i++) {  
        cin >> s[i].name;  
        cin >> s[i].score1;  
        cin >> s[i].score2;  
        cin >> s[i].score3;  
        cin >> s[i].total;  
    }  
    find_fail(s, n);  
    printByAverage(s, n);  
    system("pause");  
    return 0;  
}
```

运行结果:

```
输入学生的人数: 5  
请依次输入学生的姓名, 第一门课的成绩, 第二门课的成绩, 第三门课的成绩以及总分:  
zhaoyi 70 80 90 240  
qianer 65 32 77 174  
sunsan 100 55 68 223  
lisi 86 77 90 253  
wangwu 100 59 66 225  
*[qianer] 65 32 77  
*[sunsan] 100 55 68  
*[wangwu] 100 59 66  
  
lisi 86 77 90  
zhaoyi 70 80 90  
wangwu 100 59 66  
sunsan 100 55 68  
qianer 65 32 77  
请按任意键继续. . .
```

1.2.3 东华大学上机题

1. 写一个程序,该程序的功能是输出 100 到 999 之间的所有水仙花数。水仙花数的特点是:它的每个位上的数字的三次幂之和等于它本身。

例如: $371=3^3+7^3+1^3$,因此 371 是水仙花数。

本题为简单题,了解水仙花数的定义即可,然后根据取模、取余运算即可得到结果。

Code:

```
#include <iostream>
using namespace std;
void findWaterFlowerNum() {
    int ge = 0, shi = 0, bai = 0;
    //100~999 的数, 包含头部不包含尾部
    for (int i = 100; i < 999; i++) {
        ge = i % 10;
        int n = i / 10;
        shi = n % 10;
        bai = n / 10;
        if (ge*ge*ge + shi * shi*shi + bai * bai*bai == i)
            cout << i << " ";
    }
    cout << endl;
}

int main() {
    findWaterFlowerNum();
    system("pause");
    return 0;
}
```

运行结果:



```
153 370 371 407
请按任意键继续. . .
```

2. 在一个递增有序的数组中,有数值相同的元素存在,程序的功能是去掉数值相同的元素,使数组中不再有重复的元素。例如 (7,10,10,21,30,42,42,42,51) 变成 (7,10,21,30,42,51)

算法要求: 尽量优化算法的时间复杂度与空间复杂度。

算法思想: 确定该数组为递增有序。那么只是需要比较相邻两个元素的值即可。在这里我们可以换一种思考方式,可以采用直接插入排序的思想,若要插入的一个元素在数组中存在,则不插入。另一种思考方式是设计一个辅助数组(较常用),设计两个指针 i, j 分别指向原数组和辅助数组,比较原数组中的值和辅助数组中的值,若相同,则 $i++$, 否则将数据插入到辅助数组中,两个指针同时后移,直到遍历完原数组。

两种排序方式比较: 采用直接排序的思想,其时间复杂度为 $O(n^2)$, 空间复杂度为 $O(1)$ 采用增加一个辅助数组的方法,时间复杂度为 $O(n)$, 空间复杂度为 $O(n)$

Code:

```

#include <iostream>
using namespace std;
//按照第一种方式-直接插入排序的方式来去除同样的元素
void deleteSame1(int a[], int n) {
    int i, j;
    for (i = 0, j = 1; j < n; j++)
        if (a[i] != a[j])
            a[++i] = a[j];
    for (int k = 0; k < (i + 1); k++)
        cout << a[k] << " ";
    cout << endl;
}
//按照方式二来去除数组中的相同的元素，需要借助额外的 n 个空间
void deleteSame2(int a[], int n) {
    int *b = new int[n];
    int i = 0, k = 0;
    //数组 b 中存储的数据为 a 中按从小到大均不相同的数据
    b[0] = a[0];
    while (i < n) {
        if (a[i] == b[k]) {
            i++;
            continue;
        }
        else {
            b[++k] = a[i++];
        }
    }
    for (int j = 0; j <= k; j++)
        cout << b[j] << " ";
    cout << endl;
}

int main() {
    int a[100] = { 0 }; //假设数组中元素的最大值为 100
    int n = 0;
    cout << "请输入数组中元素的个数: ";
    cin >> n;
    //为 n 个元素赋值
    for (int i = 0; i < n; i++)
        cin >> a[i];
    deleteSame1(a, n);
    //deleteSame2(a, n);
    system("pause");
    return 0;
}

```

运行结果:

```

请输入数组中元素的个数: 9
7 10 10 10 21 32 32 56 68
7 10 21 32 56 68
请按任意键继续

```


3. 从数据结构中树的定义可知，除根节点外，树中的每个节点都有唯一的一个双亲结点。根据这一特性，可用一组连续的存储空间（一维数组）存储树中的各结点。树中的结点除保存本身结点的信息外，还要保存其双亲结点在数组中的位置（即在数组中的下标。双亲的信息为-1 则表示该结点为根结点），树的这种表示方法称为双亲表示法。

树中的每个结点的数据类型定义如下：

```
typedef struct {
    char data;    //结点数据域
    int parent;    //结点双亲在数组中的位置
} PTNode;
```

树的数据类型定义如下：

```
#define MAX_TREE_SIZE 100
```

```
typedef struct {
    PTNode nodes[MAX_TREE_SIZE];    //存储树中所有的结点
    int n;    //树中的结点数，n 不超过 100
} PTree;
```

则下图所示的树中，按照双亲表示法存储结构，存储为图 b 所示的形式（n 为 10）。

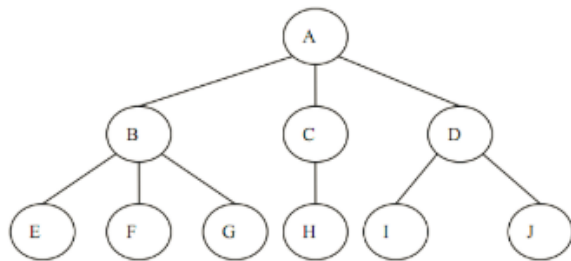


图 a 树的示意图

| 序号 | data | parent |
|----|------|--------|
| 0 | A | -1 |
| 1 | B | 0 |
| 2 | C | 0 |
| 3 | D | 0 |
| 4 | E | 1 |
| 5 | F | 1 |
| 6 | G | 1 |
| 7 | H | 2 |
| 8 | I | 3 |
| 9 | J | 3 |

图 b 双亲表示法存储

已知一棵树以存储以上形式，请编写函数 GetLeavesCount，计算叶子结点数目。

GetLeavesCount 的函数原型为：int GetLeavesCount (PTree T)

其中，形参 T 中保存了树中的结点数目以及图 b 所示的结点数组，函数返回叶子结点的数目。

比如：对图 b 的树调用函数 GetLeavesCount (T)，返回结果为 6

输入的第一个数 n 表示树中的结点数，此后有 n 行输入，每行表示一个节点的信息，第一个信息为结点的数据，第二个信息为结点的双亲结点在数组中的位置。

如输入：

```
10
a -1
b 0
c 0
d 0
e 1
f 1
```

g 1
h 2
i 3
h 3

则创建图 b 所对应的树。

对此树调用函数 `GetLeavesCount (T)` ,返回结果为 6

如输入 “

8
a -1
b 0
e 1
h 2
c 0
d 0
f 5
g 5

对此树调用函数 `GetLeavesCount (T)` ,返回结果为 4

Code:

```
#include<iostream>
using namespace std;
#define MAX_TREE_SIZE 100
typedef struct {
    char data;        //结点数据域
    int parent;       //结点双亲在数组中的位置
} PTNode;
typedef struct {
    PTNode nodes[MAX_TREE_SIZE];    //存储树中所有的结点
    int n;        //树中的结点数, n 不超过 100
} PTree;
/*
算法思想: 遍历结点数组, 当该结点的序号有另外一个节点指向时, 那么该结点为非叶节点,
当遍历完结点时, 没有一个结点的指针指向该序号的话, 那么该结点即是叶结点。
*/

//返回树中叶结点的个数
int GetLeavesCount (PTree T) {
    int count = 0; //统计非叶结点的个数, 转换一下思考方式, 否则很难直接统计叶结点个数
    for(int i=0; i<T.n; i++)
        for (int j = 0; j < T.n; j++) {
            if (i == j) //自己指向自己不算
                continue;
            else if (T.nodes[j].parent == i) {
```

```
        count++;
        break;
    }
}
return T.n-count;
}

int main() {
    cout << "请输入树的结点的个数";
    int n = 0;
    cin >> n;
    PTree pt;
    pt.n = n;
    cout << "请输入每个结点的信息: " << endl;
    for (int i = 0; i < n; i++) { // 输入结点的数据域和结点双亲在数组中的位置
        cin >> pt.nodes[i].data;
        cin >> pt.nodes[i].parent;
    }
    int leaves = GetLeavesCount(pt);
    cout << leaves << endl;
    system("pause");
    return 0;
}
```

运行结果:

```
请输入树的结点的个数10
请输入每个结点的信息:
a -1
b 0
c 0
d 0
e 1
f 1
g 1
h 2
i 3
h 3
6
请按任意键继续. . .
```

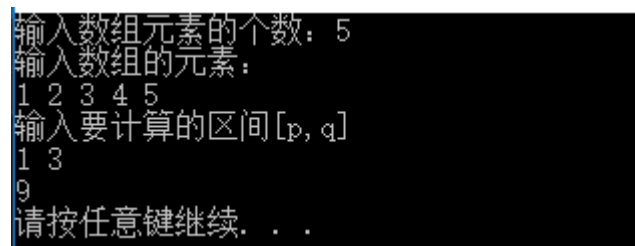
1.2.4 福州大学复试上机题

1. 输入一个数组，然后求出任意给定的区间内数值的和。

Code:

```
#include<iostream>
using namespace std;
int main() {
    cout << "输入数组元素的个数: ";
    int n = 0;
    cin >> n;
    cout << "输入数组的元素: " << endl;
    int *a = new int[n];
    for (int i = 0; i < n; i++)
        cin >> a[i];
    cout << "输入要计算的区间[p,q]" << endl;
    int p = 0, q = 0;
    cin >> p >> q;
    int sum = 0;
    for (int i = p; i < q+1; i++)
        sum += a[i];
    cout << sum << endl;
    system("pause");
    return 0;
}
```

运行结果:



输入数组元素的个数: 5
输入数组的元素:
1 2 3 4 5
输入要计算的区间[p,q]
1 3
9
请按任意键继续. . .

2. 3. 螺旋矩阵

例如 $n=4$

| | | | |
|----|----|----|---|
| 1 | 2 | 3 | 4 |
| 12 | 13 | 14 | 5 |
| 11 | 16 | 15 | 6 |
| 10 | 9 | 8 | 7 |

要求：打印出螺旋矩阵，求 i 行 j 列的数字， $0 < n < 10000$

算法思想：只要找出每一层的第一个数即可，第一个数值为上一层的第一个数 $+4*n+4$ ，循环时 n 每次减 2

```

+-----> X 轴
|  1  2  3  4
| 12 13 14  5
| 11 16 15  6
| 10  9  8  7
|
Y 轴

```

设元素 1 的坐标为(0,0)，元素 13 的坐标为 (1, 1)，……，任一元素的坐标为 (x,y) 亦可以采用递归的方式，借助于一个二维数组，从外圈开始，向内圈打印输出。

Code:

```

#include <iostream>
using namespace std;
//打印出螺旋矩阵
void printSpiralMatrix(int **matrix, int x, int y, int start, int n) {
    if (n <= 0)
        return;
    if (n == 1) {
        matrix[x][y] = start;
        return;
    }
    for (int i = x; i < x + n - 1; i++)//上部
        matrix[y][i] = start++;
    for (int j = y; j < y + n - 1; j++)//右边
        matrix[j][x + n - 1] = start++;
    for (int i = x + n - 1; i > x; i--)//底部
        matrix[y + n - 1][i] = start++;
    for (int j = y + n - 1; j > y; j--)//左边
        matrix[j][x] = start++;
    printSpiralMatrix(matrix, x + 1, y + 1, start, n - 2);
}

//查找 i 行 j 列的值, 按照人们的常识, 从 1 开始
int searchNum(int **matrix, int i, int j) {

```

```
        return matrix[i - 1][j - 1];
    }

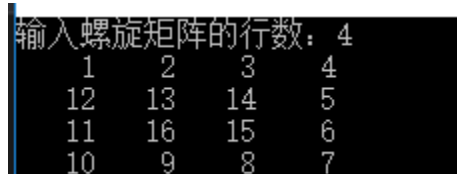
int main() {
    cout << "输入螺旋矩阵的行数: ";
    int n = 0;
    cin >> n;
    int **matrix = (int **)malloc(n * sizeof(int *));
    for (int i = 0; i < n; i++)
        matrix[i] = (int *)malloc(n * sizeof(int));

    printSpiralMatrix(matrix, 0, 0, 1, n);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            printf("%5d", matrix[i][j]);
        cout << endl;
    }
    //这里直接查找第3行第四列的数据，也可以自己指定
    int num=searchNum(matrix, 3, 4);
    cout << num << endl;

    system("pause");
    return 0;
}
```

运行结果：



```
输入螺旋矩阵的行数: 4
 1   2   3   4
12  13  14  5
11  16  15  6
10   9   8   7
```

3. 某省调查城镇交通状况，得到现有城镇道路统计表，表中列出了每条道路直接连通的城镇。省政府“畅通工程”的目标是使全省任何两个城镇间都可以实现交通（但不一定有直接的道路相连，只要互相间接通过道路可达即可）。问最少还需要建设多少条道路？

Input

测试输入包含若干测试用例。每个测试用例的第 1 行给出两个正整数，分别是城镇数目 N (< 1000) 和道路数目 M ；随后的 M 行对应 M 条道路，每行给出一对正整数，分别是该条道路直接连通的两个城镇的编号。为简单起见，城镇从 1 到 N 编号。

注意：两个城市之间可以有多个道路相通（允许存在平行边），也就是说

3 3

1 2

1 2

2 1

这种输入也是合法的

当 N 为 0 时，输入结束，该用例不被处理。

Output

对每个测试用例，在 1 行里输出最少还需要建设的道路数目。

Sample Input

测试用例 1:

4 2

1 3

4 3

测试用例 2:

3 3

1 2

1 3

2 3

测试用例 3:

5 2

1 2

3 5

测试用例 4:

999 0

0

Sample Output

结果 1:

1

结果 2:

0

结果 3:

2

结果 4:

998

算法介绍：并查集的应用

并查集是树的简单应用，它支持一下三种操作：

1. $\text{Union}(S, \text{Root1}, \text{Root2})$: 把集合 S 的子集合 Root2 并入到子集合 Root1 中，要求 Root1 和 Root2 互不相交，否则不执行合并。
2. $\text{Find}(S, x)$: 查找集合 S 中元素 x 所在的子集合，并返回该集合的名字。
3. $\text{Initial}(S)$: 将集合 S 中的每一个元素都初始化为只要一个单元素的子集合

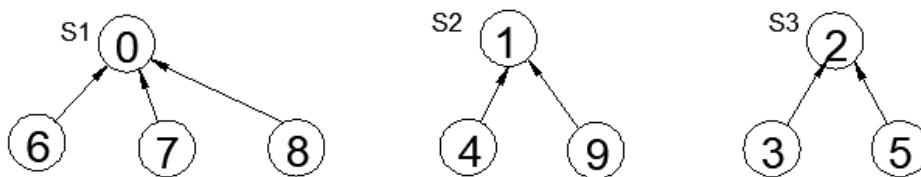
通常用树（森林）的双亲表示法作为并查集的存储结构，每个子集合用一棵树表示。

设存在一个集合 $S=\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ，执行 initial 方法后其存储结构变为：

| | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

经过计算后，集合变为： $S1=\{0, 6, 7, 8\}$ ； $S2=\{1, 4, 9\}$ ； $S3=\{2, 3, 5\}$ ；

此时并查集的树形表示和存储结构为：



| 存储结构表示 | | | | | | | | | |
|--------|----|----|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -1 | -1 | -1 | 2 | 1 | 2 | 0 | 0 | 0 | 1 |

为了得到两个子集合的并，只要将其中一个子集合根节点的双亲指向另外一个集合的根节点即可。此就是 Union 算法的思想。

并查集的模板 code

1. 结构体定义

```

#define SIZE 100
int UFsets[SIZE];

```

2. 并查集的初始化操作

```

void initial(int s[]){
    for(int i=0;i<size;i++){
        s[i]=i;
    }
}

```


3.find (x) 操作：查找包含元素 x 的树的根

```
int find (int s[],int x) {
    while(s[x] != x)
        x=s[x];
    return x;
}
```

4.Union 操作--求两个不相交的集合的并集

```
void union(int s[],int root1,int root2){
    s[root2]=root1;
}
```

Code:

```
#include<iostream>
using namespace std;
int find(int s[],int x) {
    while (s[x]!=x)
        x = s[x];
    return x;
}

void initial(int * a,int n){
    for (int i = 1; i <= n; i++)
        a[i] = i;
}

void union1(int s[], int x, int y) {
    int fx = find(s, x);
    int fy = find(s, y);
    if (fx != fy)
        s[fx] = fy;
}

//确定连通分量的个数
int find_ans(int s[], int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++)
        if (s[i] == i)
            ++sum;
    return sum;
}

int main() {
    cout << "输入城镇的数目和道路数目：";
    int m = 0, n = 0;
    cin >> n>>m;
```

```
//创建一个双亲表示法的数组
int *arr = new int[n+1];
//执行 initial 方法
initial(arr,n);
//将集合分类—每个节点都归到自己的根节点上
int a = 0, b = 0;
while (n != EOF) {
    if(!n)
        break;
    initial(arr, n);
    for (int i = 0; i < m; i++) {
        cin >> a >> b;
        union1(arr, a, b);
    }
    printf("%d\n", find_ans(arr, n));
}

system("pause");
return 0;
}
```

运行结果：

```
输入城镇的数目和道路数目：4 2
1 3
4 3
2
```

**更多编程题请参考完整版
共 30 所学校 100 道题**

2. 综合面试

综合面试在 20 分钟左右（部分学校会严格计时）

面试的老师人数为 5 人以上，通常包含专业面试老师、英语面试老师、面试记录人员等。

按教育部要求：所有考生面试将进行全程录像录音。

参加面试时，考生可提供反映能力与水平的获奖证书、各类证明等相关材料。

几乎所有的学校都存在着：面试不及格不予录取

面试评分标准（样例）：

| 序号 | 考核项目 | | 分值 |
|----|-----------|--|-----|
| 1 | 外语听力及口语测试 | | 20 |
| 2 | 专业素质和能力 | 2.1大学阶段学习情况及成绩 | 40 |
| | | 2.2对本学科(专业)理论知识和应用技能掌握程度,发现、分析和解决问题的能力 | 15 |
| | | 2.3对本学科发展动态的了解以及在本专业领域发展的潜力 | 15 |
| | | 2.4 实验和操作技能,解决实际问题的能力 | 15 |
| | | 2.5创新精神和创新能力 | 15 |
| 3 | 综合素质和能力 | 3.1 社会实践(学生工作、社团活动、志愿服务等)或实际工作表现情况 | 10 |
| | | 3.2事业心、责任感、纪律性、协作性和心理健康情况 | 5 |
| | | 3.3人文素养、举止、表达和礼仪 | 5 |
| 合计 | | | 140 |

面试分数大多学校由三个部分组成：

1. 外语听力与口语测试：英语面试。

2. 专业素质和能力：专业面试。

3. 综合素质和能力：主要考察本科阶段的社会实践、获奖经历等。

大学本科期间的成绩与竞赛经历对面试的成绩有一定程度的影响，但不是关键因素。

进入综合面试复试考场，请先敲门，进入之后环顾各位老师，先向导师团问好。

整个回答问题环节请注视对方，手机调为静音。

结束时请向导师团道谢，切勿与复试老师争执。

可提前到复试院校熟悉环境，积极调整心态。

无论是用英语提问还是用汉语提问，亦或是用哪种方式提问，这个自我介绍是必不可少的，如果老师用英语提问就用英语回答，用汉语提问就用汉语回答。对于自我介绍最重要的不是答案。自我介绍要进行一定的事前准备，你的经历比你的名字更重要，重点介绍你做过什么研究、论文题目是什么等学术方面的经验，其他证明自己能力的事情可以简要介绍。介绍内容讲究实际，不要胡编乱造。因为导师很可能根据你的介绍接着问问题，当然如果有和老师的相同爱好，可以提一下，拉近与老师的距离。所以，提前准备一些不错的句子，记住切勿不停炫耀自己以前取得的成果。

初次见面印象分是很重要的，所以对于面试礼仪，考生们应该从以下几方面准备：

着装，不管你穿什么衣服，都应该做到整洁、得体、大方、朴素，并且要使自己觉得自在，否则因为穿着不适影响发挥将得不偿失。建议，学生就保留“学生样”，工作了就应该有“工作过的样子”，男生最好不要留长发，女生不要穿吊带、高跟鞋，耳环、首饰，不要涂指甲。面试当天把头发梳理好，保持面部整洁，面带微笑。

2.1.家常扯淡篇

计算机/软工复试面试中高频问题:

信心满满的适当装【哔】非常重要

请务必在确认老师不可能查证且能自圆其说的条件下进行

总体上来说,老师会首先抛出一个基础问题,很可能跟你的项目/毕设相关。

如果你答的好,会继续加深难度,探测你的水平和对问题的理解深度,这个会不停的加分。

如果你答的不好,老师会问更简单的问题,如果还不会,会更简单,这个过程会一直减分。

面试过程中请时刻注意自己的站姿,眼神与面部表情等,切忌紧张抖腿,眼神闪躲。紧张属于正常现象,但不可过于明显,与面试官交流过程中面带微笑,尽可能看着面试官回答完所有的问题。

自我介绍时间不宜过长,时间最好控制在 2~3 分钟,自我介绍的相关内容尽量与自身实际情况相符,尤其是对于中文式介绍来说(毕竟都是 Chinese)。英文介绍过程尽可能不要太过于流畅,偶尔停顿两秒很 ok,背书式的回答反倒会降低考官给你的印象分哟!

1. 中文自我介绍

【解惑】介绍内容呢,就与个人简历尽量保持一致啦(装 B 者请慎重哈),表述方式上呢就尽量口语化,貌似不需要太正式。内容呢,就最好是切中要害,不谈无关、无用的内容,条理清晰,层次分明就看个人写作水平啦(挑重点说,要不然老师都该嫌你的自我介绍,嗯,还是中文版,翻白眼)。另外需要注意的就是,大哥大姐你一定要事先以文字的形式写好背熟,一百遍走起。

【自我介绍示例】

各位老师好:

我叫 XXX, 今年 23 岁, 是常熟理工学院软件工程专业的一名大四学生。很高兴今天能够站在这里, 下面我给老师们介绍一下我自己。

我性格开朗, 待人真诚, 踏实稳重。在校期间, 学习刻苦认真, 对学业知识能够探索钻研, 曾获得国家奖学金。同时, 我的逻辑思维能力、动手能力以及学习能力也较强, 曾参加了全国大学生信息安全竞赛。另外, 参加学校实验室时, 我深入学习了 java 和数据库, 参与实验室老师的一些科研项目, 并且自学了 PS, 帮助老师完成一些项目的软件测试和 UI 设计。我还参加了青年志愿者协会, 组织了不少的大小型志愿者活动, 具有一定的组织和策划能力。

之所以选择 XX 大学, 是因为我想和更加优秀的人站在一起并肩作战, 让自己也变得更加强大和优秀。

谢谢大家!



2. 英文自我介绍

【解惑】这个都逃不掉的哈，基本上学校都是采用英文介绍哒，网上模版一大堆，所以可借鉴资源那是相当多的，写英文介绍的时候请务必记住不要太过于统一哈(等会面试的时候老师一听，嗯？怎么都一样，扣分，淘汰，啊哈哈)，适当的自己修改一下很有必要的，记住熟读熟背呀(单词句子发音一定要好好练练，特别是各位大哥，小姐姐一般发音比你好多啦)，相信我到了真正面试的时候你不可能流利顺畅说给老师听的。

【自我介绍示例】

Good morning. Dear teachers! I am glad to be here for this interview.

At firstly, allow me to introduce myself. My name is XX, 23 years old. I come from Yancheng of Jiang Su province and I am gonging to graduate from the Software Engineering department in Changshu Institute of Technology this summery. Last year, I took part in the postgraduate examination, and got 331 points. Now I am glad to have a chance to be here, although I feel some nervous now, because this is my first time to have such an interview.

During my undergraduate years, I am interested in computer science and majored in Software Engineering. Then I studied the courses of “Java” and ”Database”, and read some related papers. Each year I got the first scholarship. Besides, I took part in the research project of my teacher and national information security competition for college students, which improve my professional ability, especially in the aspects of information security and software programming.

Recently, I focus on the new technologies such as block chain, and read some papers of XXX (from YYY labs). In order to research these new technologies better, I hope to take part in the master education in XX University. Through my diligence and persistence, I believe I could do the best in the postgraduate study. If I got the chance. Thanks.

注：XXX 为一些经典区块链文章的作者；YYY 为一些研究区块链的知名科研机构名字。

3. 你为什么选择我们学校？【为什么选择这座城市（例如深圳）】？

【解惑】这个问题其实就是想让你使劲夸一下报考的目标院校啦，挑重点夸就是这个学校这个专业的实力呀，师资力量呀，学习氛围之类的【即使这个学校不是第一志愿也给我昧着良心夸，老师爽歪歪了你不久也就爽歪歪了？？？】

4. 有女朋友/男朋友吗？

【解惑】一般来说呢，研究生导师被大家私下称作为老板的居多，站在老板的角度考虑那当然是希望你一天 24h 都给他做事为他“卖命”呀，所以当回答这个问题时最佳回答是说我结婚了，不会耽误学习和工作的，请老师放心!!! 对于往届生来说这个回答的可信度还是相当可以的，但应届生就不太符合了。那怎么办呢，我觉得应届生回答这个问题的最佳答案就是说有女/男朋友啦，而且我们也一起考取了贵校的研究生，目前 TA 和我不在同一个学院。当然啦，你也是可以本着自己的实际情况去回答，但请见机行事啦啦啦!!!【不排除老师看你不错想给你介绍对象的情况，哈哈】

5. 谈谈你为什么考计算机/软工专业研究生？

【解惑】假如这是你的本专业你就可以很大胆的说通过本科四年的学习你发现你对这个专业一如既往的very感兴趣呀，所以想读研究生来进一步自己的能力呀专业水平之类的话，那如果这不是你的本专业你就可以说通过本科四年的学习你觉得相比较本专业而言你对计算机软工的兴趣更浓，而且要表明自己本科期间有自学其相关的课程，更觉得这个专业适合自己之类的，最后点明希望来贵校进一步提升自己的能力。

6. 你是否有过项目经验？如果有项目经验，你在该项目中所担当的角色、本人完成的工作、该项目的亮点是什么？如果没有项目经验，你今后该如何弥补，才能胜任今后的开发工作？

【解惑】有项目经验的话当然爽歪歪了，这样你就可以挑一个你自认为最屌炸天的项目介绍给老师听一下(务必自己提前再回顾一下这个项目，以防老师从里面挑刺哈，另外再提前组织好语言，如果对自己参与的项目介绍起来都结巴的话那就不太好了)。如果没有项目的话，你可以开始介绍一下自己为啥没有参加过任何项目，当然原因不可以往损自己或损别人的地方说呀，你可以说你的经验不是很足，为了不给别人拖后腿就自己私下自学花的时间比较多，基础的话掌握的比较扎实，目前还在进一步提高自己的能力。今后怎么弥补的话，你可以说前期的话准备去跟着网络上那些优秀的教学视频学习如何搭建一个项目，尽可能多的去阅读 GitHub 上优秀的开源项目，汲取经验，之后再自己尝试着从小项目做起，循序渐进。

【项目介绍示例】在大三的时候我做过一个基于 J2EE 开发的图书管理系统，系统底层框架采用的是 SSH，用户界面使用的是 JSP 网页页面，采用 MVC 设计模式，后端数据库使用的是 MySQL。系统的整个功能模块包括图书借阅、图书信息、系统登录以及读者的管理等。图书借阅管理完成的是读者对图书的一系列操作，比如说借阅和浏览图书，图书管理主要指的是对图书的增删改查等功能，用户登录管理功能是通过判定用户和管理员之间的权限再对登录者能操作的范围进行划分的，此外，每个读者也都对应唯一的一个编号，这样后台管理员就能方便对读者实行增删改查等操作。在系统的设计上，Struts2 是作为系统的最基本的一层，Hibernate 是系统的持久层，用来实现与数据库的交互，而 Spring 主要是负责管理 Struts2 和 Hibernate。在系统的实现上，系统主要是通过 JSP 来完成与用户之间的交互，通过将接收到的用户的 request 请求，将数据响应给 Model 层，Model 层负责系统的业务逻辑处理，持久层中，Hibernate 通过与数据库打交道，处理请求的数据并将返回处理结果到表示层中。(注意注意注意啦!!! 在回答项目或专业相关的问题时请务必务必别说错这些专业术语呀，这个时候能专业就别搞的自己像业余一样哈，高大上的专业术语是可以给老师留下好印象哒，嗯，这个学生专业学的不错嘛，我收啦!!! 美滋滋有没有???)

7. 本科毕设做的怎么样了？请详细说明。

【解惑】这个一般都会涉及提问的，你当然要本着实际情况说啦，做的基本 ok 的话你就可以好好介绍一下毕设的思路以及相关技术工具的应用啦(请务必保证毕设你都是全程认真对待的呀，如果老师往下追问你不知道就 KO 了，好惨)，假如没怎么做的话你就说一下自己的毕设的大概内容，然后可以解释说毕业论文的事情一切都要听从学校导师的安排，导师考虑到我一直都在忙着准备贵校的初试复试就让我复试完后再着手认真准备毕设。(这个解释老师基本上就挑不出刺啦!)

【毕设介绍示例】我的毕业设计选题是基于 Web 的房屋中介管理系统的设计与实现，这个系统主要是通过解决房屋中介内部相关的管理问题来提高相关工作人员的工作效率。开发过程中我采用的是比较流行的 B/S 结构和 ASP.NET 动态网页开发技术，使用的语言是 C#，而数据库选用的是 SQL Server。ASP.NET 可以使用 .NET 平台快速的部署三层架构，因为在 .NET 中可以很方便的实现组件的装配，后台代码通过命名空间来使用自己定义的组件，显示层放在 ASP 页面中，数据库操作和逻辑层用组件来实现。针对于这个系统的具体设计，我主要是从普通用户和系统管理员两个角色所涉及到的相关功能来进行开发设计的，具体说来就是普通用户可以浏览、查看房屋信息，可以注册成为网站会员、登陆网站，发布个人房屋信息以及查看该网站相关的新闻公告，而管理员可以实现对网站会员、房屋信息以及网站的新闻公告的相关管理操作。

更多请参考完整版

2.2 专业面试常考问题汇总

本节汇集了皮皮灰整理的专业面试中常考的问题汇总。

本节内容部分参考答案来源于书籍、网络，仅供大家参考学习。

如有冲突，一切以专业参考书籍为准。

该节包含的科目有：

1.C 语言

2.数据结构

3.操作系统

4.计算机网络

5.计算机组成原理

6.数据库

7.软件工程

8.软件测试

9.面向对象

10.UML

11.离散数学

12.编译原理

13.其他问题

由于篇幅有限，所列举的知识点不可能涵盖所有内容，

如果时间充足，建议把上述列举的 12 门科目参考书籍认真学习一遍。

鉴于本书有被缴获的风险，反压题或许也不错？

2.2.1 C 语言

1. 编译 C 语言程序的基本步骤？

解析：

每个 C 语言程序写完后，都是先编译，后链接，最后运行。（.c→.obj→.exe）这个过程中注意后缀名为.c 的文本文件和后缀名为.obj 的目标文件是无法运行的，只有后缀名为**.exe 的可执行文件**才可以运行。

编译程序：编译是将编辑好的 C 语言源程序**翻译成二进制目标代码**的过程。编译过程是使用 C 语言提供的编译程序（编译器）完成的。编译时，要对源程序中的每一个语句检查语法错误，直至排除所有的语法错误后在磁盘上生成目标文件（后缀名为.obj）。

链接程序：链接就是把目标文件和其他分别进行编译生成的目标程序模块以及系统提供的标准库函数“合成”在一起，生成可以运行的可执行文件（后缀名为.exe）的过程。链接过程使用 C 语言提供的链接程序（链接器）完成，生成的可执行文件存在磁盘中。

编写一个实现某种功能的 C 语言程序，必须经历编辑→编译→链接→运行这四个步骤，每个步骤将完成不同的功能并生成不同类型的文件，如果其中某一步出错必须要回到编辑重新进行调试。

2：函数

函数：是具有一定功能的一个程序块，是 C 语言的基本组成单位。

函数不可以嵌套定义。但是可以嵌套调用。

函数名缺省返回值类型，默认为 int。

C 语言由函数组成，但有且仅有一个 main 函数！是程序运行的开始！

函数的参数可以是常量，变量，表达式，甚至是函数调用。

3：指针

指针变量的本质是用来放地址，而一般的变量是放数值的。

内存中每一个存储单元都有其存储地址，根据存储地址可以准确地找到该内存单元。一个指针变量的值就是某个内存单元的地址。指针变量是一个变量，可以被赋予不同的指针值。

int *p 中 *p 和 p 的差别：简单说*p 是**数值**，p 是**地址**！

*p 可以当做**变量**来用；*的作用是取后面地址 p 里面的**数值**。

p 是当作**地址**来使用。可以用在 scanf 函数中：scanf ("%d", p);

在声明一个指针变量的时候，“*”是类型说明符，表示其后变量是指针类型，而表达式中出现的“*”则是一个运算符用以表示指针变量所指的变量。

4：数组的指针

数组的指针其实就是数组在内存中的起始地址。数组在内存中的起始地址就是数组的变量名，也是数组第一个元素在内存中的地址。

5：静态内存分配

定义了变量或者数组以后，系统就会自动依据他们的数据类型和大小，为他们分配相应的内存单元。这些内存存在程序运行前就分配好了，不可改变。这种内存分配方式称为静态内存分配。

6: 动态内存分配

在程序实际的运行过程中, 根据程序的实际需要来分配内存单元, 称为动态内存分配。动态内存分配一般伴随着 `malloc` 函数。使用 `malloc` 函数来分配具体的内存单元, 返回一个指向该存储区域的指针变量。如果系统内存不足, 则函数返回 `NULL` 空指针。其函数原型为:

```
void *malloc(unsigned int size);
```

其中 `size` 参数的含义是分配的内存大小 (以字节为单位)。

7: 内存释放

在函数内部定义的静态存储类型的局部变量当函数运行结束时, 系统自动将内存释放。

在函数内部定义的动态分配的内存, 系统不会自动释放, 必须使用函数 `free` 来释放对应的内存。

8: 多级指针

`*p`: 一级指针: 存放变量的地址。

`**q`: 二级指针: 存放一级指针的地址。

常考题目: `int x=7; int *p=&x, **q=&p;`

问你: `*p` 为多少? `*q` 为多少? `**q` 为多少?

7 p 7

再问: `**q=&x` 的写法可以吗? 不可以, 二级指针只能存放一级指针的地址。

9: c 语言内存分布

将分配到的内存划分为四个区域:

栈区: 编译器自动分配释放, 存放函数的局部变量等

堆区: 由程序员分配和释放, 动态分配。

静态内存区: 存放全局变量和常量。

10: *p++ 和 (*p)++的之间的差别

`*p++` 是 地址会变化。 口诀: 取当前值, 然后再移动地址!

`(*p)++` 是数值会要变化。 口诀: 取当前值, 然后再使数值增加 1。

例题: `int *p,a[]={1,3,5,7,9};`

`p=a;`

请问 `*p++` 和 `(*p)++` 的数值分别为多少?

`*p++`: 这个本身的数值为 1。由于是地址会增加一, 所以指针指向数值 3 了。

`(*p)++` 这个本身的数值为 1。由于 `++` 表示数值会增加, 指针不移动, 但数值 1 由于自加了一次变成了 2。

11: 形式参数

程序进行编译时, 并不为形式参数分配存储空间。只有在被调用时, 形式参数才临时地占有存储空间。形式参数用关键字 `auto` 作存储类别的声明时, 关键字 “`auto`” 可以省略, `auto` 不写则隐含确定为 “自动存储类别”, 它属于动态存储方式。

12: strlen 和 sizeof 的区别

sizeof 相当于是个宏一样的东西,因为它只是一个运算符,而不是函数,编译时展开为常数,编译的时候有每个变量的定义表, sizeof 通过查表确定变量占用的空间,这是分配内存给 process 之前要确定的.其实可以简单的理解 sizeof 是征对"类型"的,而非"变量",但此时不能这样看如:sizeof("HELLO");中括号中为 const char *,而是一个"字符串"加上/0,所以结果大小为 6. 但:

```
char *ps = "HELLO";
sizeof(ps) = 4 //只是指针的大小
char as[8];
sizeof(as) = 8 //因为 as 的类型为 char [8],这个大小的确是 8
char aa[8][9];
sizeof((char*)aa) = 4 //还是 char *
char arr[100] = "HELLO";
sizeof(arr) = 100 //和赋什么值没什么关系,关键是"类型"是什么

int func(char p[100])
{   sizeof(p) = 4;
/*C/C++中不能传数组,只能传指针,所以任何数组都会隐式转成指针形式进行操作。*/}
```

strlen 就简单些了,但要搞清楚它的本质,它是一个函数,参数是 const char*,搞清楚它的实现,就是碰到 '\0' (字符串结尾,就停止计数,但不包括 '\0'。所以它不是看类型而是看变量,取决于变量赋的什么值。

13: printf 函数

printf (“ 第一部分 ”, 第二部分); 把第二部分的变量、表达式、常量以第一部分的形式展现出来!

| 格式说明 | 表示内容 | 格式说明 | 表示内容 |
|------|--------------|------|----------|
| %d | 整型 int | %c | 字符 char |
| %ld | 长整型 long int | %s | 字符串 |
| %f | 浮点型 float | %o | 八进制 |
| %f | double | %#o | 带前导的八进制 |
| %% | 输出一个百分号 | %x | 十六进制 |
| %5d | | %#x | 带前导的十六进制 |

举例说明:

```
printf (“%2d”, 123 );    第二部分有三位, 大于指定的两位, 原样输出 123
printf (“%5d”, 123 );    第二部分有三位, 小于指定的五位, 左边补两个空格 123
printf (“%10f”, 1.25 );  小数要求补足 6 位的, 没有六位的补 0,。结果为 1.250000
printf (“%5.3f”, 1.25 ); 小数三位, 整个五位, 结果为 1.250 (小数点算一位)
printf (“%3.1f”, 1.25 ); 小数一位, 整个三位, 结果为 1.3 (要进行四舍五入)
```

14: scanf 函数

特别注意指针在 scanf 的考察

例如: `int x=2; int *p=&x;`

`scanf ("%d", x);` 错误

`scanf ("%d", p);` 正确

`scanf ("%d", &p);` 错误

`scanf ("%d", *p)` 错误

指定输入的长度

终端输入: 1234567

`scanf ("%2d%4d%d", &x, &y, &z);` x 为 12, y 为 3456, z 为 7

终端输入: 1 234567 由于 1 和 2 中间有空格, 所以只有 1 位给 x

`scanf ("%2d%4d%d", &x, &y, &z);` x 为 1, y 为 2345, z 为 67

字符和整型是近亲:

`int x=97;`

`printf ("%d", x);` 结果为 97

`printf ("%c", x);` 结果为 a

输入时候字符和整数的区别

`scanf ("%d", &x);` 这个时候输入 1, 特别注意表示的是整数 1

`scanf ("%c", &x);` 这个时候输入 1, 特别注意表示的是字符 '1' ASCII 为整数 49。

15: 传数值和传地址

`void fun (int a, int b)`

`{ int t ;`

`t=a; a=b; b=t;`

`}`

`main ()`

`{ int x=1, y=3,`

`fun (x, y);`

`printf ("%d, %d", x, y);`

`}`

这个题目答案是 1 和 3。

传数值, fun 是用变量接受, 所以 fun 中的交换不会影响到 main 中的 x 和 y。

传数值, 形参的变化不会影响实参。

`void fun (int *a, int *b)`

`{ int t ;`

`t=*a; *a=*b; *b=t;`

`}`

`main ()`

`{ int x=1, y=3,`

`fun (&x, &y)`

`printf ("%d, %d", x, y);`

`}`

这个题目的答案就是 3 和 1。

传地址, fun 用指针接受! 这个时候 fun 中的交换, 就会影响到 main 中的 x 和 y。

传地址形参的变化绝大多数会影响到实参!

| 定 义 | 含 义 |
|---------------------------|-------------------------------|
| <code>int i;</code> | 定义整型变量 i |
| <code>int *p;</code> | p 为指向整型变量的指针变量 |
| <code>int a[n];</code> | 定义整型数组 a, 它有 n 个元素 |
| <code>int *p[n];</code> | 定义指针数组 p, 它由 n 个指向整型数据的指针元素组成 |
| <code>int (*p)[n];</code> | p 为指向含有 n 个元素的一元数组的指针变量 |
| <code>int f();</code> | f 为返回整型数值的函数 |
| <code>int *p();</code> | p 为返回一个指针的函数, 该指针指向整型数据 |
| <code>int (*p)();</code> | p 为指向函数的指针, 该函数返回一个整型值 |
| <code>int **p;</code> | p 是一个指针变量, 它指向一个指向整型数据的指针变量 |

2.2.2 数据结构

2.2.3 操作系统

2.2.4 计算机网络

更多请参考完整版

2.2.5 计算机组成原理

1: 计算机的组成和各部件功能

计算机由运算器、存储器、控制器、输入设备和输出设备组成。运算器用来完成算术运算和逻辑运算并将中间结果暂存在运算器内。存储器用来放数据和程序。控制器用来控制、指挥程序和数据的输入、运行以及处理运算结果。输入设备用来将人们熟悉的信息形式转换为机器能识别的信息形式，常见的有键盘、鼠标等。输出设备用来将机器运算的结果转换为人们熟悉的信息形式，如打印机输出、显示器输出等。

2: 计算机结构

冯·诺依曼结构：指令和数据放在同一个存储器。

哈佛结构：指令和数据分别放在两个存储器。

哈佛结构的计算机比冯诺依曼结构的计算机速度更快。

冯·诺依曼结构的特点：

- (1) 计算机由运算器、存储器、控制器、输入设备和输出设备五大部件组成。
- (2) 指令和数据以同等地位存放于存储器内，并可按地址寻访。
- (3) 指令和数据均用二进制数表示。
- (4) 指令由操作码和地址码组成，操作码用来表示操作的性质，地址码用来表示操作数在存储器中的位置。
- (5) 指令在存储器内按顺序存放。通常指令是顺序执行的，在特定条件下，可根据运算结果或根据设定的条件改变执行顺序。
- (6) 机器以运算器为中心，输入输出设备与存储器间的数据传送通过运算器完成。

3: 计算机性能指标

CPI：表示每条指令周期数，即执行一条指令所需的平均时钟周期数。

$CPI = \text{执行某段程序所需的 CPU 时钟周期数} \div \text{程序包含的指令条数}$

MIPS：表示平均每秒执行多少百万条定点指令数

$MIPS = \text{指令数} \div (\text{程序执行时间} \times 10^6)$

主频/时钟周期：CPU 的工作节拍受主时钟控制，主时钟不断产生固定频率的时钟，主时钟的频率叫 CPU 的主频。

外频：通常为系统总线的工作频率。

多层次的存储器

4: 数据存储的小端方式和大端方式

小端方式：先存储低位字节后存储高位字节

大端方式：先存储高位字节后存储低位字节

5: 边界对齐方式

假设存储字长为 32 位，可以按照字节、半字、字寻址。在对准边界的 32 位字长的计算机中，半字地址是 2 的整数倍，字地址是 4 的整数倍，当所存数据不能满足此要求时可以填充空白字节。这样保证对齐以后，可以使得每次取数据都是一次访存取出。

6: cache

Cache 是一种高速缓冲存储器,是为了解决 CPU 和主存之间速度不匹配而采用的一项重要技术。Cache 存储速度比主存快,通过向 CPU 高速提供指令和数据,加快了程序的执行速度。

7: 计算 cache 的命中率

在一个程序执行期间,设 N_c 表示 cache 完成存取的总次数, N_m 表示主存完成存取的总次数, h 定义为命中率, 则有

$$h = \frac{N_c}{N_c + N_m}$$

若 t_c 表示命中时的 cache 访问时间, t_m 表示未命中时的主存访问时间, $1-h$ 表示未命中率, 则 cache/主存系统的平均访问时间 t_a 为:

$$t_a = ht_c + (1-h)t_m$$

设 $r = t_m/t_c$ 表示主存慢于 cache 的倍率, e 表示访问效率, 则有

$$e = \frac{t_c}{t_a} = \frac{t_c}{ht_c + (1-h)t_m} = \frac{1}{h + (1-h)r} = \frac{1}{r + (1-r)h}$$

8: 主存与 cache 的地址映射

为了把主存块放到 cache 中,必须应用某种方法把主存地址定位到 cache 中,称做地址映射。地址映射的方式有全相联映射方式、直接映射方式、组相联映射方式。

全相联映射方式:在全相联映射中,将主存中一个块的地址与块的内容一起存于 cache 的行中,其中块地址存于 cache 行的标记部分中。这种带全部块地址一起保存的方法,可使主存的一个块直接拷贝到 cache 中的任意一行上,非常灵活。全相联 cache 中,全部标记用相联存储器来实现,全部数据用一个 RAM 来实现。

直接映射方式:直接映射是一种多对一的映射关系,但一个主存块只能拷贝到 cache 的一个特定行位置上去。Cache 的行号 i 和主存的块号 j 有如下函数关系:

$$i = j \bmod m \quad (m \text{ 为 cache 中的总行数})$$

组相联映射方式:将 cache 分成 u 组,每组 v 行。主存块存放到哪个组是固定的,至于存到该组哪一行是灵活的,即有如下函数关系:

$$m = u \times v$$

$$\text{组号 } q = j \bmod u$$

TIP:可以知道这三种方式的特点。全相联映射方式是整块空间都可以放元素,元素放置的位置不固定。直接映射方式中每一个内存块只能放置到计算好的特定位置上。组相联映射方式则是要求每个内存块只能放到固定的组中,但是在组中位置不固定。

9: 程序的局部性

时间局部性:如果一个存储项被访问,则可能该项会很快被再次访问。

空间局部性:如果一个存储项被访问,则该项及其邻近的项也可能很快被访问。

10: 虚拟存储器

虚存空间的用户程序按照虚地址(或逻辑地址)编程并存放在辅存中。程序运行时,由地址变换机构依据当时分配给该程序的实地址空间把程序的一部分调入实存。

每次访存时,首先判断该虚地址所对应的部分是否在实存中:如果是,则进行地址转换并用实地址访问主存;否则,按照某种算法将辅存中的部分程序调度进内存,再按同样的方法访问主存。

11: 各种虚拟存储器的优缺点

页式存储管理的优点是页长固定，因而便于构造页表、易于管理，且不存在外碎片。缺点是页长与程序的逻辑大小不相关。

段式存储管理的优点是段的逻辑独立性使其易于编译、管理、修改和保护，也便于多道程序的共享。段长可以根据需要动态改变，允许自由调度，以便有效利用主存空间。缺点是主存空间分配比较麻烦。容易在段间留下许多外碎片，造成存储空间利用率降低。必须用加法操作通过段起址与段内偏移量的求和运算求得物理地址。因此，段式存储管理比页式存储管理方式需要更多的硬件支持。

段页式虚拟存储器是段式虚拟存储器和页式虚拟存储器的结合。实存被等分成页，每个程序先按逻辑结构分段，每段再按照实存的页大小分页，程序按页进行调入和调出操作，但可按段进行编程、保护和共享。

12: cache 与虚存有什么区别

Cache 是高速缓存处理器，是速度很快但是容量很小的一种存储器。

用户编制程序时所使用的地址称为虚地址，其所对应的存储空间称为虚存空间。

我们可以看出 cache-主存访问机制和主存辅存访问机制是类似的。Cache 与主存构成了系统的内存，而主存和辅存依靠辅助软硬件构成了虚拟存储器。

相同：

| | |
|-------|--|
| 出发点相同 | 都是为了提高存储系统的性价比而构造的分层存储体系，都力图使存储系统的性能接近高速存储器，而价格和容量接近低速存储器。 |
| 原理相同 | 都是利用了程序运行时的局部性原理把最近常用的信息块从相对慢速而大容量的存储器调入相对高速而小容量的存储器。 |

不同：

| | Cache-主存 | 主存-辅存 |
|-----------|--|---|
| 侧重点不同 | 主要解决内存和 CPU 的速度差异问题 | 主要解决存储容量问题 |
| 数据通路不同 | CPU 与 Cache 与主存之间均有直接访问通路，cache 不命中时可以直接访问主存 | 辅存与 CPU 之间不存在直接的数据通路，当主存不命中时只能通过调页解决，CPU 最终还是要访问主存 |
| 透明性不同 | Cache 的管理完全由硬件完成，对系统程序员和应用程序员均透明 | 虚存管理由软件和硬件共同完成，由于软件的介入，虚存对实现存储管理的系统程序员不透明，而只对应用程序员透明。 |
| 未命中时的损失不同 | 主存取取时间是 cache 的 5-10 倍。 损失较小 | 主存的存取速度比辅存的存取速度快上千倍。 损失大。主存未命中时的性能损失远大于 cache 未命中时的损失。 |

13: RAM

RAM 是一种可读/写存储器，其特点是存储器的任何一个存储单元的内容都可以随机存取，而且存取时间与存储单元的物理位置无关。计算机系统的主存都采用这种随机存储器。

14: ROM

只读存储器是能读出其存储的内容，而不能对其重新写入的存储器。这种存储器一旦存入了原始信息后，在程序执行过程中，只能将内部信息读出，而不能随意重新写入新的信息去改变原始信息。通常用它存放固定不变的程序、常数和汉字字库，甚至用于操作系统的固化。它与随机存储器可共同作为主存的一部分，统一构成主存的地址域。

指令系统

15: 指令系统

CISC 复杂指令系统计算机：采用一整套计算机指令进行操作的计算机。指令系统多达几百条。

RISC 精简指令系统计算机：它精简了指令集，只保留了那些常用的指令。降低了控制器设计的难度。

16: 指令的格式

一条指令分为操作码字段和地址码字段。根据一条指令中有几个操作数地址，可将该指令称为几操作数指令或几地址指令。

零地址指令的指令字中只有操作码，没有地址码。

一地址指令只有一个地址码，它指定一个操作数，另一个操作数地址是隐含的。

二地址指令常被称为双操作数指令，它有两个地址码字段 A_1 和 A_2 ，分别指明参与操作的两个数在内存中或运算器中通用寄存器的地址，其中地址 A_1 兼做存放操作结果的地址。

三地址指令中有三个操作数地址 A_1 、 A_2 和 A_3 。 A_1 为被操作数地址。 A_2 为操作数地址。 A_3 为存放操作结果的地址。

17: 寻址方式

隐含寻址、立即寻址、直接寻址、间接寻址、寄存器寻址、寄存器间接寻址、偏移寻址、段寻址、堆栈寻址。

隐含寻址：在指令中不是明显的给出操作数的地址，而是在指令中隐含着操作数的地址。

立即寻址：指令的地址字段指出的不是操作数的地址，而是操作数本身。

直接寻址：在指令格式的地址字段中直接指出操作数在内存的地址。

间接寻址：指令地址字段中的形式地址不是操作数的真正地址，而是操作数地址的指示器。

寄存器寻址：操作数不在内存中，而是放在 CPU 的通用寄存器中。指令中给出的操作数地址不是内存的地址单元号，而是通用寄存器的编号。

寄存器间接寻址：指令中的寄存器内存不是操作数，而是操作数的地址。

偏移寻址：它要求指令中有两个地址字段。其中一个为形式地址，另一个指的是某个寄存器，寄存器的内容加上形式地址以后就能得到有效地址。

段寻址：段寄存器中的地址左移 4 位后，加上某个寄存器中存储的偏移量，即可得到所需的内存地址。

堆栈寻址：数据存储与栈顶地址有关，需要一个栈顶指示器。

中央处理器

18: CPU 基本结构

CPU 由运算器、cache、控制器组成。

控制器：由程序计数器、指令寄存器、指令译码器、时序产生器和操作控制器组成。它是发布命令的“决策机构”，即完成协调和指挥整个计算机系统的操作。主要功能有：

- (1) 从指令 cache 中取出一条指令，并指出下一条指令在指令 cache 中的位置。
- (2) 对指令进行译码或测试，并产生对应的操作控制信号，以便启动规定的动作。

运算器：由算术逻辑单元、通用寄存器、数据缓冲寄存器和状态条件寄存器组成。它是数据加工处理部件。主要功能有：

- (1) 执行所有的算术运算
- (2) 执行所有的逻辑运算，并进行逻辑测试。

19: 微程序控制器

微程序控制器主要由控制存储器、微指令寄存器和地址转移逻辑三大部分组成。

(1) 控制存储器

控制存储器用来存放实现全部指令系统的微程序，它是一种只读型存储器。一旦微程序固化，机器运行时则只读不写。其工作过程是：每读出一条微指令，则执行这条微指令；接着又读出下一条微指令，又执行这一条微指令...读出一条微指令并执行微指令的时间总和称为一个微指令周期。通常，在串行方式的微程序控制器中，微指令周期就是只读存储器的工作周期。**控制存储器的字长就是微指令字的长度，其存储容量视机器指令系统而定，即取决于微程序的数量。**对控制存储器的要求是速度快，读出周期要短。

(2) 微指令寄存器

微指令寄存器用来存放由控制存储器读出的一条微指令信息。其中微地址寄存器决定将要访问的下一条微指令的地址，而微命令寄存器则保存一条微指令的操作控制字段和判别测试字段的信息。

(3) 地址转移逻辑

在一般情况下，微指令由控制存储器读出后直接给出下一条微指令的地址，通常我们简称微地址，这个微地址信息就存放在微地址寄存器中。如果微程序不出现分支，那么下一条微指令的地址就直接由微地址寄存器给出。当微程序出现分支时，意味着微程序出现条件转移。在这种情况下，通过判别测试字段 P 和执行部件的“状态条件”反馈信息，去修改微地址寄存器的内容，并按改好的内容去读下一条微指令。地址转移逻辑就承担自动完成修改微地址的任务。

20: 影响流水线性能的因素

资源相关：指多条指令进入流水线后在同一机器时钟周期内争用同一个功能部件所发生的冲突。

数据相关：在一个程序中，如果必须要等前一条指令执行完毕后，才能执行后一条指令，那么这两条指令就是数据相关的。

控制相关：控制相关冲突是由转移指令引起的。当执行转移指令时，依据转移条件的产生结果，可能为顺序取下条指令，也可能转移到新的目标地址取指令，从而使流水线发生断流。

采用以下方法来减少转移指令对流水线性能的影响：

延迟转移法：由编译程序重排指令序列来实现。基本思想是“先执行再转移，即发生转移取时并不排空指令流水线，而是让紧跟在转移指令 **lb** 之后已进入流水线的少数几条指令继续完成。如果这些指令是与转移指令结果无关的有用指令，那么延迟损失时间片正好得到了有效的利用。

转移预测法：硬件方法来实现，依据指令过去的行为来预测将来的行为。通过使用转移取和顺序取两路指令预取队列器以及目标指令 **cache**，可将转移预测提前到取指阶段进行，以获得良好的效果。

21：控制方式

控制不同操作序列时序信号的方法，称为控制器的控制方式。常见的有同步控制、异步控制、联合控制三种方式，其实质反映了时序信号的定时方式。

同步控制方式：

在任何情况下，已定的指令在执行所需的机器周期数和时钟周期数都是固定不变的，称为同步控制方式。根据不同情况，同步控制方式可选取如下方案：

- (1) 采用完全统一的机器周期执行各种不同的指令。这意味着所有指令周期具有相同的节拍电位和相同的节拍脉冲数。显然，对简单指令和简单的操作来说，将造成时间浪费。
- (2) 采用不定长机器周期。将大多数操作安排在一个较短时间的机器周期内完成，对某些时间紧张的操作，则采取延长时间周期的办法来解决。
- (3) 中央控制与局部控制结合。将大部分指令安排在固定的机器周期完成，称为中央控制，对少数复杂指令（乘、除、浮点运算）采用另外的时序进行定时，称为局部控制。

异步控制方式

异步控制方式的特点是：每条指令、每个操作控制信号需要多少时间就占用多少时间。这意味着每条指令的指令周期可由多少不等的机器周期数组成；也可以是当控制器发出某一操作控制信号后，等待执行部件完成操作后发回“回答”信号，再开始新的操作。用这种方式形成的操作序列没有固定的 CPU 周期数（节拍电位）或者严格的时钟周期（节拍脉冲）与之同步。

联合控制方式

联合控制方式是同步控制和异步控制相结合的方式。一种情况是，大部分操作序列安排在固定的机器周期中，对某些时间难以确定的操作则以执行部件的“回答”信号作为本次操作的结束。

TIP：同步控制方式要求了每个指令只能是固定的机器周期数，即指令的时间是机器周期的倍数。经过对应的时间周期就意味着指令执行完了。异步则没有这个要求，每个指令需要多少时间就占用多少时间，发出对应信号才意味着执行完毕。

22：总线的作用

总线是构成计算机系统的互联机构，是多个系统功能部件之间进行数据传送的公共通路。

总线分为如下三类：

- (1) 内部总线：CPU 内部连接各寄存器及运算部件之间的总线。
- (2) 系统总线：CPU 同计算机系统的其他高速功能部件互相连接的总线。
- (3) I/O 总线：中、低速 I/O 设备之间互相连接的总线。

输入输出系统

23: 磁盘的结构

硬盘存储器由磁盘驱动器、磁盘控制器和盘片三大部分组成。

磁盘驱动器：包括主轴、定位驱动、数据控制等三个部分。主轴受传动机构控制，可使磁盘组作高速旋转运动。定位驱动系统可以驱动磁头沿盘面径向位置运动以寻找目标磁道位置。数据控制完成数据转换和读/写控制操作。

磁盘控制器：是主机和磁盘驱动器的接口。它接受主机发来的命令，将它转换成磁盘驱动器的控制命令，实现主机和驱动器之间的数据格式转换和数据传送，并控制驱动器的读/写。

盘片：盘片是存储信息的载体。

磁头：由软磁材料做铁芯绕有读写线圈的电磁铁。写入时利用磁头来使盘片具有不同的磁化状态，读取时又利用磁头来判别这些不同的磁化状态。

磁道：通常将磁盘片表面称为记录面。记录面上一系列同心圆称为磁道。

扇区：将磁道划分为若干个小的区段，这个区段就被称为扇区。

一次磁盘操作时间包括寻道时间、延迟时间和传输时间。

寻道时间指将磁头移动到指定磁道所需要花费的时间。

延迟时间指磁头定位到某一磁道的扇区所需要的时间。

传输时间指从磁盘读出或者向磁盘写入数据所经历的时间。

24: 中断响应过程

(1) 当中断处理的 CPU 控制权转移涉及特权级改变时，必须把当前的 SS 和 ESP 两个寄存器的内容压入系统堆栈予以保存。

(2) 将标志寄存器 EFLAGS 的内容也压入堆栈。

(3) 清除标志触发器 TF 和 IF。

(4) 当前的代码段寄存器 CS 和指令寄存器 EIP 也压入此堆栈。

(5) 如果中断发生伴随有错误码，则错误码也压入此堆栈。

(6) 完成上述终端现场保护后，从中断向量号获取的中断服务子程序入口地址分别装入 CS 和 EIP，开始执行中断服务子程序。

(7) 中断服务子程序最后的 IRET 指令使中断返回。保存在堆栈中的中断现场信息被恢复，并由中断点继续执行原程序。

TIP 中断相应过程即为保护现场、获取入口地址、中断返回恢复现场这样三个步骤。

25: DMA 传送方式

DMA 是指外部设备不通过 CPU 而直接与系统内存交换数据的接口技术。

在 DMA 方式中，一批数据传送前的准备工作，以及传送结束后的处理工作，均为管理程序承担，而 DMA 控制器仅负责数据传送的工作。

DMA 方式基本流程：从外围设备发出 DMA 请求。CPU 响应请求，把 CPU 工作改成 DMA 操作方式，DMA 控制器从 CPU 接管总线的控制。由 DMA 控制器对内存寻址，并执行数据传送。向 CPU 报告 DMA 操作结束。

DMA 采用如下方式和 CPU 分时使用内存：

A. 停止 CPU 访问内存 B.周期挪用 C.DMA 与 CPU 交替访问内存。

2.2.6 数据库

2.2.7 软件工程

2.2.8 软件测试

2.2.9 面向对象

2.2.10 UML

2.2.11 离散数学

2.2.12 编译原理

2.2.13 其他问题

更多请参考完整版