

目录

一、基础知识与编程环境：	1
1、计算机发展史	2
2、计算机的基本构成	2
3、存储单位	3
4、编程语言	3
5、计算机网络	3
6、Internet 知识	3
二、C++程序设计：	4
1、程序基本概念	4
2、基本数据类型	4
3、程序基本语句	5
4、基本运算	5
5、数学库常用函数	6
6、结构化程序设计	6
7、数组	6
8、字符串的处理	7
9、函数与递归	7
10、结构体与联合体	7
11、指针类型	7
三、数据结构：	9
1、线性结构	9
2、简单树	9
3、特殊树	10
4、简单图	11
四、算法：	12
1、算法概念与描述	12
2、入门算法	12
3、基础算法	13
4、数值处理算法	13
5、排序算法	13
6、搜索算法	15
7、图论算法	16
8、动态规划	16
五、数学与其他：	18
1、数及其运算	18
2、初等数学	18
3、初等数论	18
4、离散与组合数学	20
5.其他	20
六、阅读程序：	21
七、完善程序：	21

一、基础知识与编程环境：

1、计算机发展史

第一台计算机：ENIAC，1946 年，美国宾西法尼亚大学，数值计算（弹道计算）。

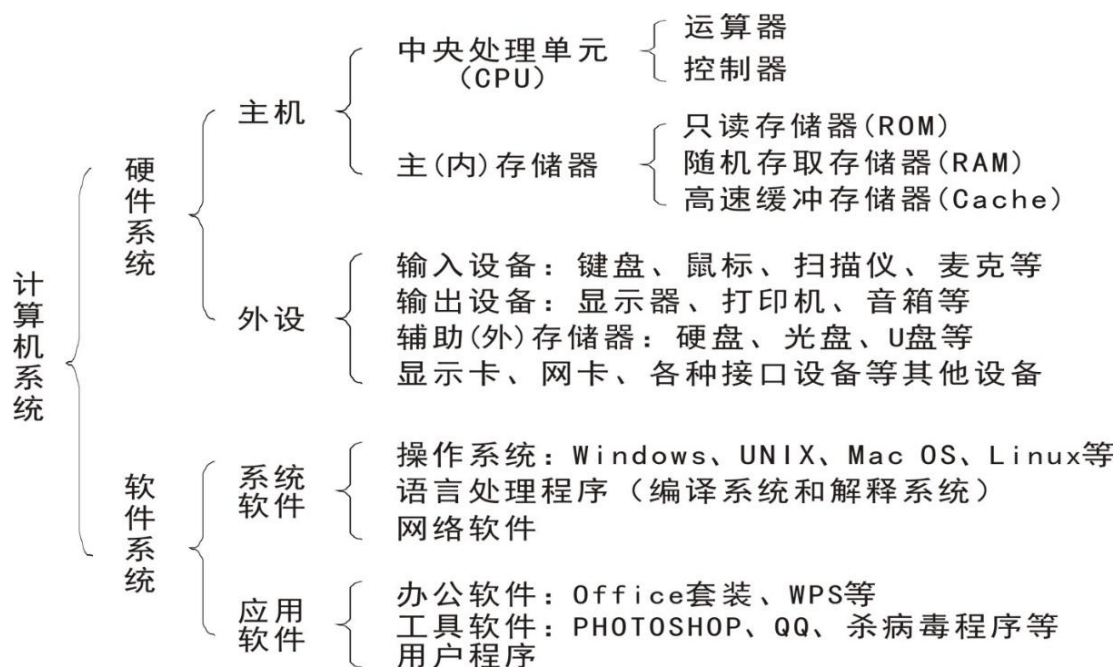
四代：电子管、晶体管、中小规模集成电路、大规模/超大规模集成电路。

人物：美籍匈牙利科学家冯·诺伊曼（计算机之父），英国科学家阿兰·图灵（人工智能之父），Ada Lovelace（第一位程序员）。

理论：冯·诺依曼架构（硬件构成：存储器、运算器、控制器、输入/输出设备），

奖项：图灵奖是计算机领域最高奖项。

2、计算机的基本构成



硬件组成：运算器、控制器、存储器、输入/输出设备。

CPU 包括运算器、控制器、寄存器。

存储器包括内、外存储器。

断电后能保存数据：ROM（只读存储器）、硬盘。

断电后不能保存数据：RAM（随机存储器）、CPU、显卡。

3、存储单位：位（bit），字节（Byte）

缩写：KB、MB、GB、TB、PB，换算关系：

1 Byte = 8 bit 1 个字节有 8 位

1 KB = 1024 Byte 1 MB = 1024 KB 1 GB = 1024 MB

1 TB = 1024 GB 1 PB = 1024 TB 注：10TB 大约为正常人脑储存量。

4、编程语言：

通常分为三类：机器语言、汇编语言和高级语言。

按转换分：编译型(C、C++、Pascal、Delphi)；解释型(Java、VB、Python、Ruby)

编译型(源代码整个翻译成机器语言，跨平台性差)，解释型(译一句执行一句，跨平台性好)。

按面向分：面向过程(C、Fortran)；面向对象(C++、Java、Python)

5、计算机网络：TCP/IP 是网络的一组通信协议。

网络主要功能：资源共享、信息传递、分布处理、综合信息服务。

网络分类：局域网（LAN）、城域网（MAN）、广域网（WAN）。

拓扑结构：星形、总线型、环形、树形、网状型(一般网都是网状型，不规则)。

体系结构：应用层、传输层、网络层、链路层、物理层。

IP 地址：形式：a.b.c.d，其中 a、b、c、d 是 0~255 中的整数，如：192.168.0.1。

三类私用地址：A 类：10.0.0.0~10.255.255.255

B 类：172.16.0.0~172.31.255.255 C 类：192.168.0.0~192.168.255.255

6、Internet 知识：

域名：.com(商业组织) .net(网络组织) .edu(教育机构) .gov(政府部门)

.cn(中国) .us(美国) .jp(日本) .hk(香港) .tw(台湾)

服务：WWW(万维网)、E-mail(电子邮件)、FTP(文件传输协议)、Telnet(远程登录)。

WWW 网页用超文本标记语言（HTML）在超文本传输协议（HTTP）支持下运行。

电子邮件：传输协议（SMTP）、扩展协议（MIME）、接收邮件（POP3）协议、邮件访问协议（IMAP）。

二、C++程序设计：

1、程序基本概念：

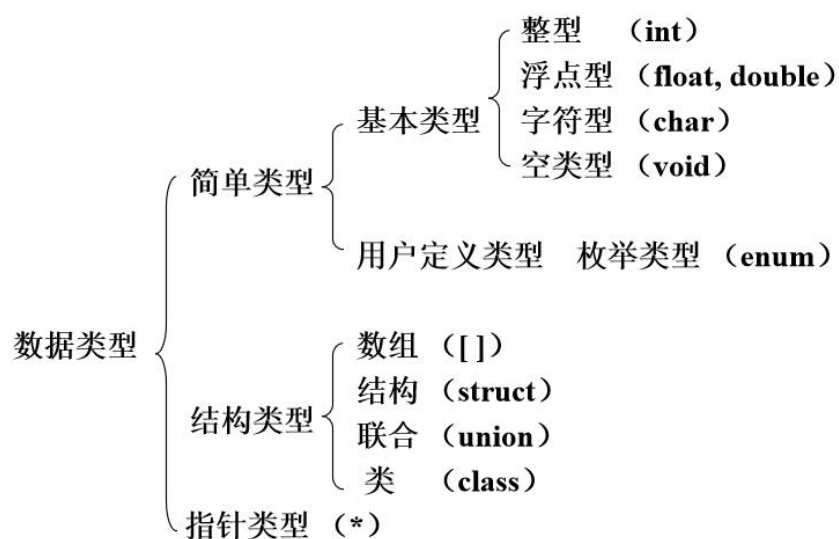
标识符：以字母或下划线开始，由字母、数字和下划线组成的符号串。

常量：`const float PI = 3.14159` 变量：`int x=666`

头文件：`#include<bits/stdc++.h>` //调用系统提供的库函数

名字空间：`using namespace std` //可在不同空间使用相同名字类或函数

2、基本数据类型：



常用类型：

类型	说明	长度 (字节)	表示范围	备注
char	字符型	1	-128~127	$-2^7 \sim (2^7 - 1)$
int	整型	4	-2147483648~2147483647	$-2^{31} \sim (2^{31} - 1)$
double	双精度型	8	$-1.7 \times 10^{308} \sim 1.7 \times 10^{308}$	15 位有效位

整型：

数据类型	定义标识符	占字节数	数值范围	数值范围
短整型	short [int]	2(16位)	-32768~32767	$-2^{15} \sim 2^{15}-1$
整型	[long] int	4(32位)	-2147483648~2147483647	$-2^{31} \sim 2^{31}-1$
长整型	long [int]	4(32位)	-2147483648~2147483647	$-2^{31} \sim 2^{31}-1$
超长整型	long long [int]	8(64位)	-9223372036854775808~ 9223372036854775807	$-2^{63} \sim 2^{63}-1$
无符号整型	unsigned [int]	2(16位)	0~65535	$0 \sim 2^{16}-1$
无符号短整型	unsigned short [int]	2(16位)	0~65535	$0 \sim 2^{16}-1$
无符号长整型	unsigned long [int]	4(32位)	0~4294967295	$0 \sim 2^{32}-1$
无符号超长整型	unsigned long long	8(64位)	0~18446744073709551615	$0 \sim 2^{64}-1$

整型十进制：第一个数字必须非 0，如 23、45

整型八进制：必须以数字 0 为前缀，如 023、045

整型十六进制：以 0x 或 0X 为前缀，如 0x2a、0x45、0xff

字符类型（char）一般用 ASCII 编码参与运算。

布尔类型（bool）用来判断真假，结果只有 true 和 false 两种。

true 转换为整型值 1，false 转换为整型值 0；非 0 值可以隐式转换为 true，而 0 转换为 false。

3、程序基本语句：

流输入：cin >> a >> b; 流输出：cout << a << b;

格式化输入：scanf("%d%d", &n, &m); 格式化输出：printf("%d", ans);

注：cin 和 cout 属于 C++ 的概念，而 scanf 和 printf 属于 C 的概念。

条件语句（if，if-else，switch）

循环语句（for，while，do-while，多层循环）

4、基本运算：

算术运算：加+、减-、乘*、除/、求余%

关系运算：大于>、大于等于>=、小于<、小于等于<=、等于==、不等于!=

逻辑运算：与（&&）、或（||）、非（!）

变量自增与自减运算：++、--

三目运算：？：如：int max = a > b ? a : b;

位运算：与（&）、或（|）、非（~）、异或（^）、左移（<<）、右移（>>）

与：有 0 则 0，全 1 才 1；

或：有 1 则 1，全 0 才 0；

异或：不同则 1，相同则 0；

左移：高位丢弃（超出类型范围时），低位补 0；

注：在高位没有 1 丢弃的情况下， $a \ll 1$ 相当于 $a * 2$ ， $a \ll 2$ 相当于 $a * 2^2 = a * 4$ 。

右移：低位丢弃，高位补 0（无符号数）或高位补符号位（有符号数）；

注： $a \gg 1$ 相当于 $a / 2$ ， $a \gg 2$ 相当于 $a / 2^2 = a / 4$ （除不尽时直接去掉小数部分）。

***原码、反码与补码：**计算机中均用补码存储。

正数：原码与反码和补码均相同，如：

$[+1] = [00000001]_{\text{原}} = [00000001]_{\text{反}} = [00000001]_{\text{补}}$

负数：反码在原码基础上，除符号位外各位取反，补码在反码的基础上+1，如：

$[-1] = [10000001]_{\text{原}} = [11111110]_{\text{反}} = [11111111]_{\text{补}}$

5、数学库常用函数：

abs 绝对值；	absolute value
sqrt 开方；	square root 开方根
ceil 向上取整；	ceil /si:l/ 装天花板
floor 向下取整；	floor 铺地板
round 四舍五入；	

6、结构化程序设计：流程图（能读懂即可）

7、数组：

默认数组下标从[0]开始，注意数组越界问题。

二维数组可视为行列表格/矩阵。

多维数组使用多层循环来输入/输出。

8、字符串的处理：

1) 字符数组：如 `char a[101], b[101];`

字符串函数：

`strcat(字符串名 1, 字符串名 2)` 连接字符串；

`strcpy(字符串名 1, 字符串名 2)` 复制字符串；

`strcmp(字符串名 1, 字符串名 2)` 比较字符串大小；

`strlen(字符串名)` 计算字符串长度；

2) 字符串 **string** 类型：

字符串的声明和初始化：`string str1= "black cat";`

获取含空格的字符串：`getline(cin, str)`

字符串赋值：`str1 = str2`

字符串拼接：`str1 += str2`

`str1.empty()` 判断字符串是否为空

`str1.size()` 返回字符串个数

`a.insert(pos, b)` 在下标 `pos` 之前插入字符串 `b`

`a.substr(pos, len)` 返回从 `pos` 开始 `len` 个字符的字符串

9、函数与递归：

函数调用：先定义后使用，或先声明后使用；

递归函数：函数自己调用自己，称为“递归”调用，这样的函数称为“递归函数”。

注：阅读程序题一般可用“打表法”来从最初的递归（终止）条件反推题目。

10、结构体与联合体：

结构体：**struct** 能储存不同类型数据。

联合体：**union** 能储存不同类型数据，但同一时间只能储存其中一种类型数据。

11、指针类型：

能够存放对象地址的变量称为**指针变量**，地址操作符“&”，如“&a”；

指针操作：

说 明	样 例
指针定义： 类型说明符 * 指针变量名；	int a=10; int * p;
取地址运算符： &	p=&a;
间接运算符： *	* p=20;
指针变量直接存取的是内存地址	cout<<p; 结果可能是:0x4097ce
间接存取的才是储存类型的值	cout<<* p; 结果是:20

1) 指针与数组：

引入指针变量后，就可以用两种方法访问数组元素了，

例如定义了 `int a[5]; int *pa=a;`

第一种方法为**下标法**，即用 `pa[i]`形式访问 `a` 的数组元素。

第二种方法为**指针法**，即采用`*(pa+i)`形式，用间接访问的方法来访问数组元素。

2) 指针与字符串：

在 C++中，我们可以用两种方式访问字符串。

①用**字符数组**存放一个字符串，然后输出该字符串。

```
char str[]="I love China!";
```

```
printf("%s\n", str);
```

②用**字符指针**指向一个字符串。

```
const char *str="I love China!";
```

```
printf("%s\n", str);
```

3) 指针与结构体

可以定义一个**指针**变量来指向一个**结构体**变量。

在引用结构体成员时，有以下三种方法可以使用：

结构体变量名.成员名

(*指针变量名).成员名

指针变量名->成员名

三、数据结构：

1、线性结构：

1) 链表：同时存储**数据域**和**指针域**，通过**结点**链接。

单链表：每个结点存储 **1** 个指针域（后继结点的存储位置）；

双向链表：每个结点存储 **2** 个指针域（前驱结点和后继节点）；

循环链表：最后一个节点的指针指向**头结点**（单向循环）；最后一个节点的指针指向头结点，且**头结点的前驱**指向**最后一个结点**（双向循环）。

2) 栈：先进后出

声明：stack<数据类型>栈名，例如 stack<int> s; 对于 stack 类型的变量 s，

s.empty(); //判断栈是否为空

s.size(); //返回栈中元素个数

s.top(); //获取栈顶元素

s.push(x); //入栈(在栈顶加入一个元素)

s.pop(); //出栈(删除栈顶元素)

3) 队列：先进先出

声明：queue<数据类型>队名，例如：queue<int>q; 对于 queue 类型的变量 q，

q.empty(); //判断队列为空

q.size(); //返回队列中元素个数

q.front(); //获取队头元素

q.back(); //获取队尾元素

q.pop(); //(出队)删除队头元素

q.push(x); //(入队)在队尾插入一个元素

2、简单树：

1) 树

度：结点的**子树个数**称为该结点的度，树中**各结点度的最大值**称为这棵树的度。

层次：一般定义一个**根节点**的层次为 **1**，其他结点的层次等于它的父结点层次+1。

深度：一棵树中**所有的结点的层次的最大值**称为树的深度(depth)。

2) 二叉树：**度数为 2** 的树 （可以为空二叉树、仅有根节点的二叉树）。

【性质 1】在二叉树的第 i 层上最多有 2^{i-1} 个结点($i \geq 1$)。

【性质 2】深度为 k 的二叉树至多有 $2^k - 1$ 个结点($k \geq 1$)。

【性质 3】对任意一棵二叉树，如果其叶结点数为 n_0 ，度为 2 的结点数为 n_2 ，则一定满足： $n_0 = n_2 + 1$ 。

满二叉树：一棵深度为 k 且有 $2^k - 1$ 个结点的二叉树称为满二叉树。

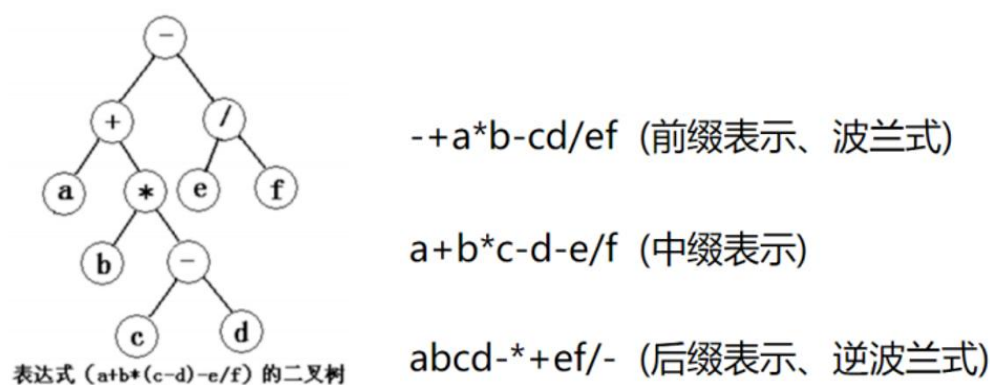
3) 二叉树的遍历

先序（根）遍历：根节点→左子树→右子树；

中序（根）遍历：左子树→根节点→右子树；

后序（根）遍历：左子树→右子树→根节点；

表达式树：



前缀、中缀、后缀表达式的转换：

1) 中缀表达式运算规则：

先乘除，后加减，从左到右计算，

先括号内，后括号外。从而能构造出二叉树。

2) 中缀表达式转二叉树：

从最后被计算的运算符开始，

运算符为树根，运算符左边为左子树，右边为右子树。

3、特殊树：

1) 完全二叉树：有 n 个结点的二叉树当且仅当其每一个结点都与深度为 k 的满二叉树中编号从 1 到 n 的结点一一对应时，称为完全二叉树。

注：约定编号从根结点起，自上而下，从左到右。

【性质 4】具有 n 个结点的完全二叉树的深度为 $\text{floor}(\log_2 n) + 1$ 。

【性质 5】 对于一棵 n 个结点的完全二叉树，对任一结点(编号为 i)，有：

①如果 $i=1$ 则结点 i 为根，无父结点;如果 $i>1$ 则其父结点编号为 $i/2$ 。如果 $2*i>n$ ，则结点 i 无左孩子(当然也无右孩子,即结点 i 为叶结点);否则左孩子编号为 $2*i$ 。

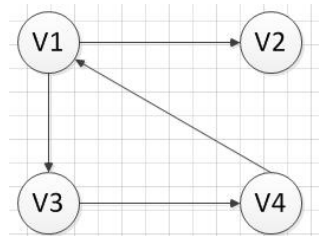
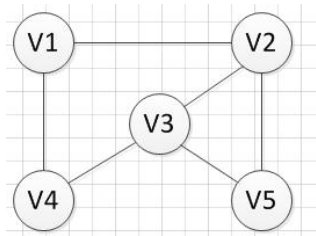
②如果 $2*i+1>n$ ，则结点 i 无右孩子;否则右孩子编号为 $2*i+1$ 。

2) 哈夫曼树与哈夫曼编码： 按“合并果子”题意来简化管理。

哈夫曼树：带权路径最短的二叉树，又称最优二叉树。

4、简单图：

1) 有向图、无向图： 若图中所有边都为无向边，则该图为无向图（左下图），反之若都为有向边，则为有向图（右下图）。



度数：无向图中每个顶点所连边的边数。

出度：有向图中每个顶点出边边数。

入度：有向图中每个顶点入边边数。

2) 连通图： 在无向图中，如果两个顶点之间有路径，则称这两个顶点是连通的。

如果图中任意两个顶点都是连通的，那么该图为连通图。

强连通：若有向图中有两点能互相到达则称这两点强连通。

强连通图：任意两点强连通的有向图称为强连通图。

强连通分量：任意有向图的极大强连通子图称为强连通分量。

四、算法：

1、算法概念与描述：

1) 时间复杂度：即算法操作步数的估算。

技巧：估算（内层）循环语句的执行次数

大O记法：抓住主要矛盾（高阶项），忽略次要因素（低阶项，常数系数）；

有一种“上界”的含义，表示该算法“最差不过”……，举例：

如何对操作步计数——（内层）循环语句的执行次数	
<pre>for i = 1 to n for j = i to n a[i,j] = i+j</pre>	<pre>for i = 1 to n for j = i to n a[i,j] = i+j b[i] = b[i] + 1 for i = 1 to n a[i,1] = b[i]</pre>
$n + (n-1) + \dots + 1 = \frac{n(n+1)}{2}$	
它们都是 n^2 算法， $O(n^2)$	$n(n+1)+n$ 高阶项为 n^2 , $O(n^2)$

2) 算法描述：自然语言、流程图、伪代码

自然语言描述的语句	伪代码描述的语句
- 重复下述步骤（2，3，4，5）两次 - 将7升桶装满； - 将它的水倒向5升桶（满为止）； - 倒空5升桶； - 将7升桶中剩的水倒向5升桶； - 将7升桶装满； - 将它的水倒向5升桶（满为止）	<pre>1. for i = 1 to 2 2. A ← 7 3. A ← A - (5-B) 4. B ← A 5. A ← 7 6. A ← A - (5-B)</pre> A和B需要适当初始化，这里尤其是B

2、入门算法：

枚举算法：枚举出所有可能的情况，然后判断或者统计，从而解决问题。

模拟算法：指让程序完整的按照题目叙述的方式执行运行得到最终答案。

3、基础算法：

1) **贪心法**：将局部最优解合成为全局最优解。

注：前提是局部最优策略能产生全局最优解，典型题目“合并果子”。

2) **递推法**：是指从已知的初始条件出发，依据某种递推关系，逐次推出所要求的各中间结果及最后结果。

注：首要问题是得到相邻的数据项间的关系（即递推关系），典型题目“斐波那契”数列。

3) **递归法**：一种函数直接或间接地调用自身的算法。

注：有递归边界（即递归终止条件），典型题目“辗转相除法求最大公约数”“递归求阶乘”。

4) **二分法**：将问题分解成两个较小问题求解时的分治方法称之为二分法。

注：仅处理有序数据；算法时间复杂度为 $O(\log n)$ 。

5) **倍增法**：即成倍的增长，指在进行递推时，如果状态空间很大，通常的线性递推无法满足时间和空间复杂度的要求，就可以通过成倍的增长，只递推状态空间中在 2 的整数次幂位置上的值作为代表。

4、数值处理算法：

高精度加法、减法、乘法、除法。

统一处理步骤：

- 1) 输入字符数组：将高精度数输入为“字符数组”或“字符串”（string）；
- 2) 转存整数数组：转存至“整数数组”（ASCII 码减'0'转换，逆序存储）；
- 3) 循环按位计算：加法“满十进一”，减法“为负借位补十”，乘除法按模拟。
- 4) 逆序输出结果：因为步骤 2) 是逆序存储的，输出结果时需要再逆序。

5、排序算法：

1) **冒泡排序**：重复“从序列左边开始比较相邻两个数字的大小，再根据结果交换两个数字的位置”该操作的算法。

5 6 3 4 1 2	5 3 4 1 2 6
5 3 6 4 1 2	3 5 4 1 2 6
5 3 4 6 1 2	3 4 5 1 2 6
5 3 4 1 6 2	3 4 1 5 2 6
5 3 4 1 2 6	3 4 1 2 5 6
第一趟结束，最后的6固定下来	第二趟结束，5固定下来

注：总的比较次数为 $(n-1) + (n-2) + \cdots + 1 = (n^2 - n) / 2$ ，时间复杂度为 $O(n^2)$ 。

2) 选择排序：重复“从待排序的数据中寻找最小值（或最大值），将其与序列最左边的数字进行交换”该操作的算法。



注：总的比较次数是 $(n-1)+(n-2)+\dots+1 \approx n^2/2$ 次，时间复杂度为 $O(n^2)$ 。

3) 插入排序：重复“从右侧未排序区域内取出一个数据，然后将它插入到左侧已排序区域内合适的位置上”该操作的算法。



首先，我们假设最左边的数字 5 已经完成排序，所以此时只有 5 是已归位的数字。

接下来，从待排数字（未排序区域）中取出最左边的数字 3，将它与左边已归位的数字进行比较。若左边的数字更大，就交换这两个数字。重复该操作，直到左边已归位的数字比取出的数字更小，或者取出的数字已经被移到整个序列的最左边为止。



由于 $5 > 3$ ，所以交换这两个数字。



对数字3的操作到此结束。此时3和5已归位，还剩下右边7个数字尚未排序。

注：在最糟糕的情况下，第2轮需要操作1次，第3轮操作2次……第n轮操作n-1次，所以时间复杂度和冒泡排序的一样，都为 $O(n^2)$ 。

4) 计数排序：略

*排序算法比较：

(1)稳定性比较：

插入排序、冒泡排序及其他线形排序是稳定的。

选择排序、快速排序是不稳定的。

(2)时间复杂性比较

插入排序、冒泡排序、选择排序的时间复杂性为 $O(n^2)$;快速排序、堆排序、归并排序的时间复杂性为 $O(n\log_2 n)$;

在最好情况下，直接插入排序和冒泡排序最快;在平均情况下，快速排序最快;在最坏情况下，堆排序和归并排序最快。

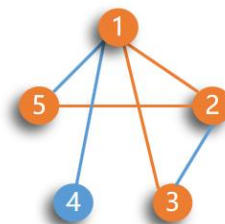
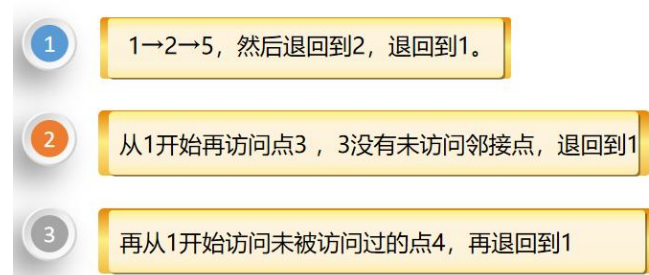
6、搜索算法：

1) 深度优先搜索 (DFS)：沿着一条路径不断往下搜索直到不能再继续为止，然后再折返，开始搜索下一条候补路径。

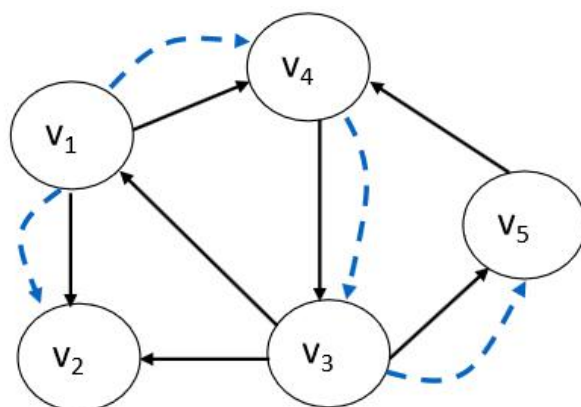
2) 广度优先搜索算法 (BFS)：每次都先将搜索树每一层所有节点全部访问完后再访问下一层，因此也被称作“按层搜索”。

7、图论算法：

1) 深度优先遍历：



2) 广度优先遍历：



BFS次序是：v1→v2→v4→v3→v5

3) 泛洪算法 (flood fill)

【概念】：洪水填充算法，用于将格点图某一块连通区域内的所有格点状态修改为目标状态。

【实现】：状态通常用颜色表示，从一个起始顶点开始把附近与其连通的顶点填充成新的颜色，直到连通区域内的所有顶点都被处理过为止。算法可以使用深度优先遍历、广度优先遍历...

1) 四邻域填充法（上、下、左、右）；

2) 八邻域填充法（四邻域四个方向+对角线四个方向）。

8、动态规划：

【渊源】：将原问题分解为若干子问题，自底向上先求解最小的子问题，并把结果存储起来，在求解大的问题时直接查询已存储的小的子问题的解，以避免重复计算，从而提高效率。

【使用场景】：求解最优化问题，尤其是带有多步决策的最优化问题，具备以下3个要素：

1) 最优子结构：问题的最优解所包含的子问题的解也是最优的，即一个问题的最优解只取决于其子问题的最优解；

2) 无后效性：将原问题分解为若干子问题，每个子问题的求解过程作为一个阶段，当前阶段的求解只与之前阶段有关，与之后阶段无关，即某阶段的状态一旦确定，就不受该状态后续决策的影响。

3) **重叠子问题（非必要）**：求解过程中每次产生的子问题并不总是新问题，会有大量子问题重复，此时只需在已存储的解中查询，无须再次求解。此性质非必要但凸显动态规划优势。

【设计模式】：

- 1) **划分阶段**：按照问题的**时间或空间**特性，将问题分为**若干个阶段**；注意阶段一定要有序或可排序，以保证各状态的**无后效性**。
- 2) **状态表示**：将问题发展到**各个阶段**时所处的各种情况用**不同的状态**表示出来，通常状态的表示可以设为问题**最终结果的一般化表示**，并将**最优解**进行**递归定义**。
- 3) **决策与状态转移方程**：根据**上一阶段的状态和决策**来导出**本阶段的状态**就是**状态转移**。
通常做法是根据**相邻**两个阶段的状态**之间的关系**来确定**决策方法**和**状态转移方程**。
- 4) **边界条件**：给出的状态转移方程是一个递推式，需要一个递推的终止条件或边界条件。
- 5) **答案**：问题的求解目标。

【进阶内容】：

1) **简单一维**：

概念：在一维数组上标识状态及进行转移的动态规划。

典型问题：**最长上升子序列 (LIS)**；IOI1994 Number Triangles (洛谷 P1216)

2) **简单背包**：

概念：有 n 种物品和一个容量为 V 的背包，第 i 种物品的**费用**（如体积）是 v_i 、数量是 n_i 、每个价值是 c_i ，可以选择任意给出的物品中的若干个放入背包中，求满足：所选择物品的**体积之和不超过 V** 的情况下物品**价值的和最大**是多少？

典型问题：**0-1 背包**（每种物品数量为 1 时）；NOIP2001 普及组 装箱问题（洛谷 P1049）

3) **简单区间**：

概念：在区间上进行动态规划，即求解一段区间上的最优解。

差异：一维动态规划只有一维，而区间动态规划解决的问题与区间有关，**状态有二维**，分别表示区间的左端点与右端点；一个状态通常由被它包含且比它更小的区间状态转移而来。

实现：区间动态规划通常以区间长度作为动态规划的阶段。

阶段（长度）、状态（左右端点）、决策三者按照由外到内的顺序构成三层循环。

典型问题：**NOI1995 石子合并**（洛谷 P1880）

五、数学与其他：

1、数及其运算：

1) 进制转换：

二进制（八进制、十六进制）**转十进制**：统一“按权展开”；

十进制转二进制（八进制、十六进制）：

整数部分，除 2（8、16）**取余**，自底向上；

小数部分，乘 2（8、16）**取整**、自顶向下。

二进制转八进制：3 位并 1 位，不足补 0； 八进制转二进制：1 位拆 3 位；

二进制转十六进制：4 位并 1 位，不足补 0； 十六进制转二进制：1 位拆 4 位；

2、初等数学：略

3、初等数论：

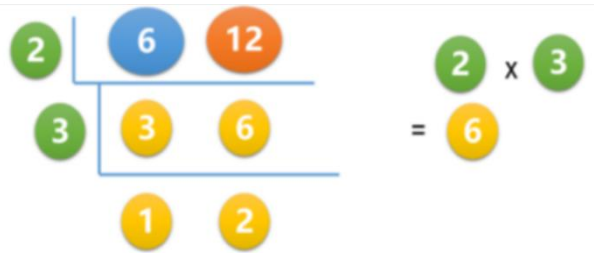
1) 数列规律：

数列类型	规律归纳	
	通项公式	求和公式
等差数列	$a_n = a_1 + (n-1) \cdot d$	$S_n = \frac{1}{2}n \cdot (a_n + a_1)$
等比数列	$a_n = a_1 \cdot q^{n-1}$	$S_n = a_1(1-q^n)/(1-q)$
$1^2 + 2^2 + 3^2 + \dots$	$a_n = n^2$	$S_n = \frac{1}{6}n(n+1)(2n+1)$
$1^3 + 2^3 + 3^3 + \dots$	$a_n = n^3$	$S_n = [\frac{1}{2}n(n+1)]^2$
$1^2 + 3^2 + 5^2 + \dots$	$a_n = (2n-1)^2$	$S_n = \frac{1}{3}n(4n^2-1)$
斐波那切数列	$a_n = a_{n-1} + a_{n-2}$	

2) 最大公约数求法：

枚举法：略

短除法：先用这几个数的公约数连续去除，一直除到所有的商互质为止，然后把所有的除数连乘起来，所得的积就是这几个数的最大公约数。



***辗转相除法（欧几里德算法）：**用较大数除以较小数，再用出现的余数（第一个余数）去除除数，再用出现的余数（第二余数）去除第一余数，如此反复，直到最后余数是 0 为止，两个数的最大公约数，就是最后的除数。



3) 约数个数定理：

对于一个正整数 a ，可以分解质因数：

$$a = a_1^{r_1} * a_2^{r_2} * a_3^{r_3} \dots$$

则 a 的约数的个数 = $(r_1 + 1) * (r_2 + 1) * (r_3 + 1) \dots$



4) 埃氏筛法求质数：把从 1 开始某一范围内的正整数从小到大顺序排列，1 不是质数，首先筛掉。剩下的数中选择最小的质数，然后去掉它的倍数，以此类推，直到筛子为空时结束。

4、离散与组合数学：

集合：略

加法原理：做一件事情，完成它有 n 类方式，第一类方式有 M_1 种方法，第二类方式有 M_2 种方法，……，第 n 类方式有 M_n 种方法，那么完成这件事情共有 $M_1 + M_2 + \dots + M_n$ 种方法。

乘法原理：做一件事，完成它需要分成 n 个步骤，做第一步有 M_1 种不同的方法，做第二步有 M_2 种不同的方法，……，做第 n 步有 M_n 种不同的方法。那么完成这件事共有 $N = M_1 \times M_2 \times M_3 \times \dots \times M_n$ 种不同的方法。

排列：

$$A_n^m = \underbrace{n(n-1)(n-2)\cdots(n-m+1)}_{m\text{个因子}} = \frac{n!}{(n-m)!}$$

注： $0! = 1$

组合：

$$C_n^m = \frac{A_n^m}{m!} = \frac{n!}{m!(n-m)!} \quad C(n, m) = C(n, n-m), \text{ 其中 } n \geq m$$

解题技巧：

①**捆绑法：**7 个学生站成一排，甲、乙必须站在一起有多少种排法？

解法：将甲、乙捆绑在一起， $A(6, 6) * 2 = 1440$ 。

②**插空法：**7 个学生站成一排，甲、乙互不相邻有多少种排法？

解法：把甲、乙单独拿出来，剩下 5 名学生有 6 个空隙， $A(5, 5) * A(6, 2) = 3600$ 。

③**排除法：**重新排列 1234 使得每一个数字都不在原来的位置上，共有几种排法？

解法：4 个数字都在原来位置：1234

3 个数字都在原来位置：没有

2 个数字都在原来位置：1243 1432 1324 4231 3214 2134

1 个数字在原来位置：1423 1342 4213 3241 4132 2431 3124 2314

$A(4, 4) - 15 = 4 * 3 * 2 * 1 - 15 = 9$ 。

5.其他：

1) ASCII 码：

10进制ASCII码表

ASCII	字符	ASCII	字符	ASCII	字符
000	NUL	:	:	:	:
001	SOH	032	space	065	A
002	STX	033	!	066	B
003	ETX	034	“	067	C
004	EOT	:	:	:	:
005	ENO	048	0	097	a
006	ACK	049	1	098	b
007	(振铃)	050	2	099	c
:	:	:	:	:	:
010	LF	063	?	254	■
012	FF	064	@	255	空格FF

六、阅读程序：

1、**模拟代入**：第一、二题甚至第三题都可以**暴力模拟**算出，技巧：

- 小样例数据代入（优先题目中出现的数据），快速总结出规律；
- 根据题目选项差异（如：首尾值、字符位数等），局部代入来匹配或排除；
- 每列表示核心变量/数组名称，每行表示程序阶段运行结果

2、**总结规律**：如果数据很大只能找规律，可使用类似数学归纳法，通过小样例数据代入再找规律。

3、**读懂程序**：变量名、函数名有利于理解程序，如：max（最大）、min（最小）、len（长度）、ans（答案）、encoder（编码）、decoder（解码）、search（查找）、solve（解决<方案>）、dfs（深度优先搜索）、prev（前一个）、valid（有效的）、cur（光标）。

七、完善程序：

技巧：从**题目需求**出发，思考如果是自己会如何解题，**求正解**，切忌乱猜答案。

- 1、注意已给的变量名和函数名，根据含义推导用途。
- 2、程序上下文互相参照，如分治、DFS 前后会有类似的语句。