作者序言

为什么要分享软设笔记？

一直以来，我都认为人类世界最美好的事情莫过于用一个生命去影响另一个生命（前提是正面影响）。如果我的一点点努力，能够帮助你通过软考，我想，这将是很有意义的一件事情。考软设的人很多，我作为过来人，希望能够给予你们一些帮助。

为什么要写序言？

每一位读者，都不是我本人。看这本笔记，未必知道如何去看，所以我写下想说的话，希望你们在看笔记的过程中，少走弯路。

软设，我付出了许多的心血，虽然不至于头悬梁锥刺股，但也是为其折腾了许多日子，至少，是送了好多钱……这一路走来，有很多感想，写在这里，也算是将其作为一个树洞吧。

软考难吗？

相信许多初次考软设的人都喜欢问老司机这个问题，其实没有人能给你正确答案，除了你自己。对于一般人来说，上清华北大难吗？当然难，但是当中不乏 NB 之人，能够轻易过线，对他们来说就是 so easy。所以，难与不难，取决与你的实力。

为什么要考软考？

软考的中级证书是能够拿来直接评职称的（听说），也许你不知道什么是职称，其实我也不知道……感觉好像是蛮有用的东西，自行百度吧。我只是个学生，大学时间相对工作而言还是比较多的，考考证，也不是什么坏事。而且，努力久了，软设不仅仅是一门专业证书的考试，更是我的信仰。

对于求职，听说没什么大用，毕竟，计算机的发展大家也都知道，技术是第一位的。但是软设证书，如果人家有，你没有，我想这感觉应该不会太好。

对于经济，考一次 140RMB（浙江省价格）。曾经有人在群里收购软设挂证，开价 3000 元，对于学生的我，还是蛮开心的。

这个笔记与《软件设计师教程》的区别

先说说这个笔记的由来吧。人脑毕竟存储空间有限，且极易遗忘。所以我把重要的知识点都会记下来，反复看。有本《软件设计师教程》（以下简称教程），他写的非常详细。如果要买，我建议去某鱼看看，正版太贵了，个人建议阿，如果你钱多，或者坚决支持正版，当我没说。

我与这本书的主要区别在于精。教程一共 700 页，共计 100 万汉字，写得非常非常详细，但是一般人都看不了多少页，枯燥难懂。而我的笔记，根据多套试卷的考点考频来综合考虑，记录了高频的一些知识点。且用最简单易懂的语言来解释，有必要的话，我也配上了练习题与解析总共才 2 万汉字，几乎都是高频考点，如果我的笔记写的不够详细，可以再去

看对应教程中的知识点。

复习软设的注意事项

第一点，建议你们买一本《软件设计师同步辅导》，这本书是比较简短的知识点，详细度不如教程，但是他不仅仅是按章节分类，更是具备了课后习题与解析。虽然不得不说，解析不咋滴。

第二点，千万不要以为得到了这本笔记就和得到了《九阴真经》一样，这本书，最适合的读者是谁？是我！他针对的是我的弱点，每个知识点都是一针见血。对于读者，我写的知识点未必是你不会的，没写的知识点未必是你会的。只能说，最适合的读者是我。所以，我建议大家只是把这本笔记当作一个小小的参考资料，根据自己的做题经验，去写本专属自己的笔记。

第三点，不用去搜什么模拟卷，真题多的是。网上都有，另外，不要做年份太早的，我最初是从 06 年开始做的，知识点差的太多了。建议大家刷 2010 年之后的题。

第四点，虽然我是计算机专业，但是我的 office 功底确实渣的一批，所以排版也是非常糟糕的，只会用标题一，标题二，标题三来区分知识点的从属关系。我以个人的理解，将众多的知识分成了多个章节，数据结构、软件工程、多媒体等。所以可能分的不太科学，只是根据自己的理解而已。但是没关系，考试不会考你某个知识点是属于哪个章节的，你会做就可以。这本笔记的最佳使用方法是打开 word 的导航窗体，根据章节去看。

明天，也很美好

明天，就是 2016 年下半年的软设开考之日。没错，作为这本笔记的书写者，自己都还没通过软设……我程序员考了 2 次才过，软设，这是第 2 次考，希望能过吧。但是讲实话，若是没过，我也不会特别难过，放平心态吧，尽人事听天命。只能说技不如人，不能怨天尤人。拥有一个良好的心态，我觉得比拥有所谓的证书更重要。留个 QQ 吧，我也不知道留了有啥用，出版社也不会来找我，我就是想留一个……948832626.如果学习的过程中有疑问的，还是别+我 QQ 了，我考完就忘了，回答不了你。我留 QQ 纯粹就是觉得应该有一个自己的标识，就是这样……

希望这本笔记，能让你们早日成为软件设计师！

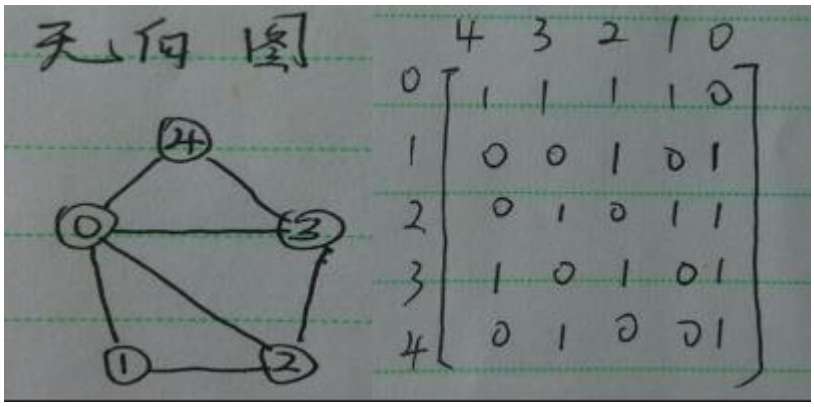
2016 年 11 月 11 日

数据结构

邻接矩阵

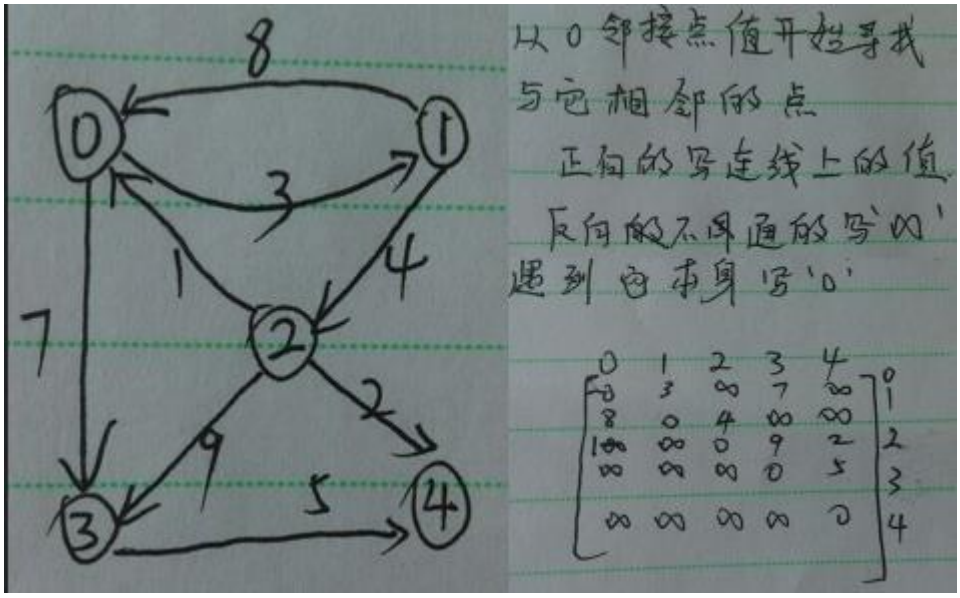
无向图

无向图邻接矩阵：其邻接矩阵第 i 行元素的和即为顶点 i 的度,例如：顶点 4 的度就是第四行的和，即 2。



有向图

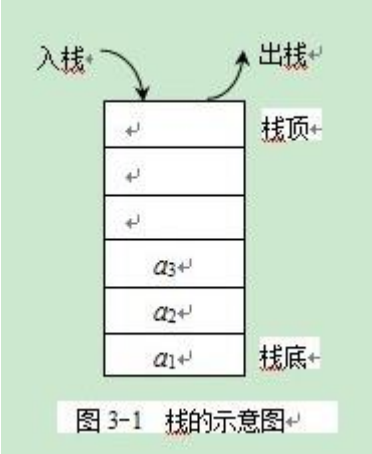
其邻接矩阵的第 i 行元素之和为顶点 i 的出度，而邻接矩阵的第 j 列元素之和为顶点 j 的入度。例如：顶点 3 的出度和入度分别为 5 和 16.



存储结构

顺序存储结构

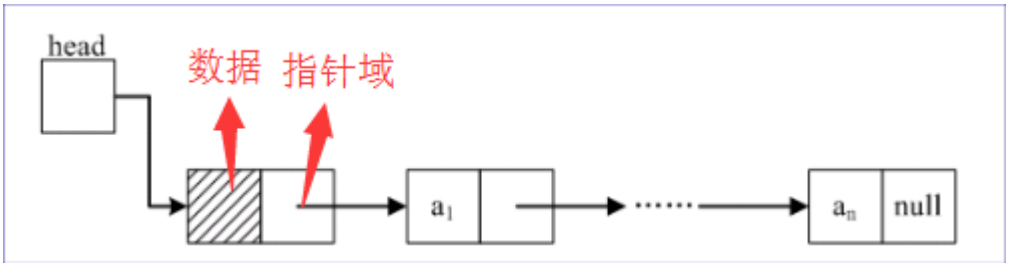
用一组**地址连续的存储单元**依次存储线性表的各个**数据元素**， 适用于**频繁查询**时使用。



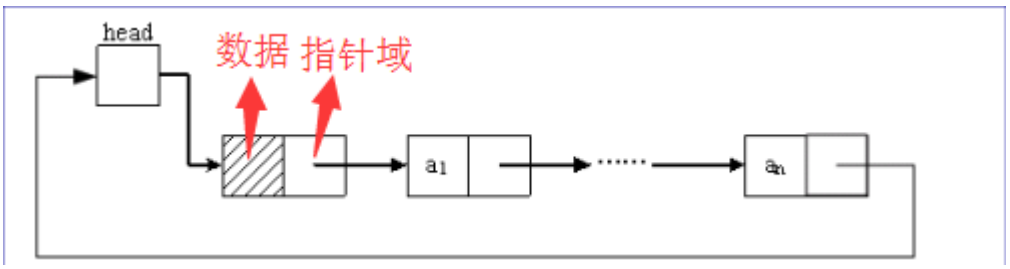
链式存储结构

在计算机中用一组**任意的存储单元**存储线性表的**数据元素**(这组存储单元可以是连续的，也可以是不连续的)，适用于在较频繁地插入、删除、更新元素时使用。

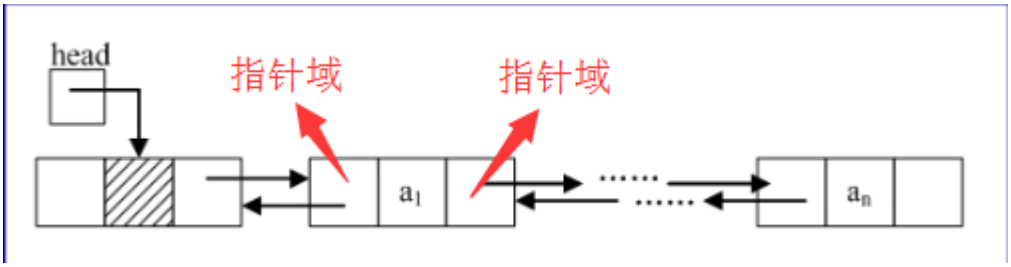
单链表



循环链表



双链表

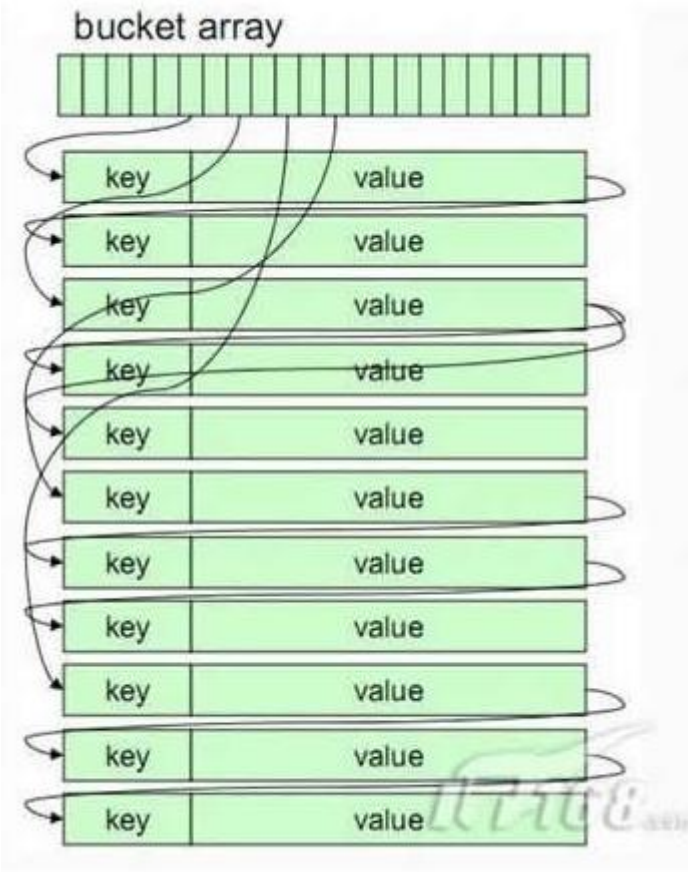


各链表的比较

因为双链表有两个指针域，因此，双链表的**灵活度优于单链表**，但是双链表的**开支要大**一些

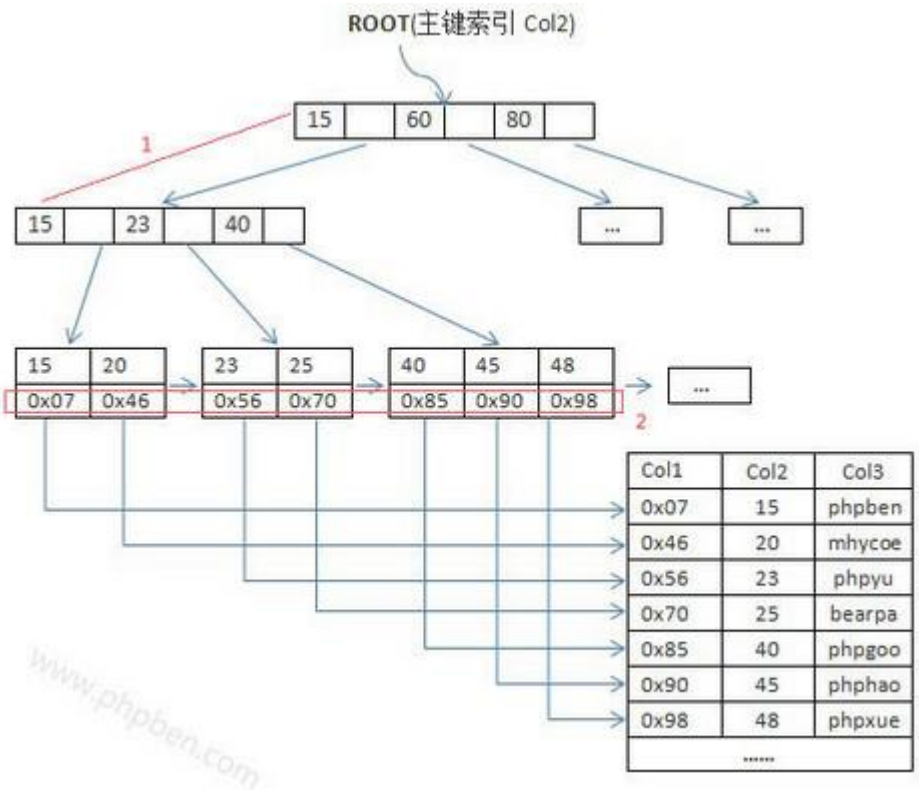
散列存储结构

将**数据元素**的**存储位置**与**关键码**之间建立确定对应关系的查找技术，即**键值对**。



索引存储结构

索引是一个单独的、物理的数据库结构，它是某个表中一列或若干列值的集合和相应的指向表中物理标识这些值的数据页的逻辑指针清单。比如数据库



树

二叉排序树

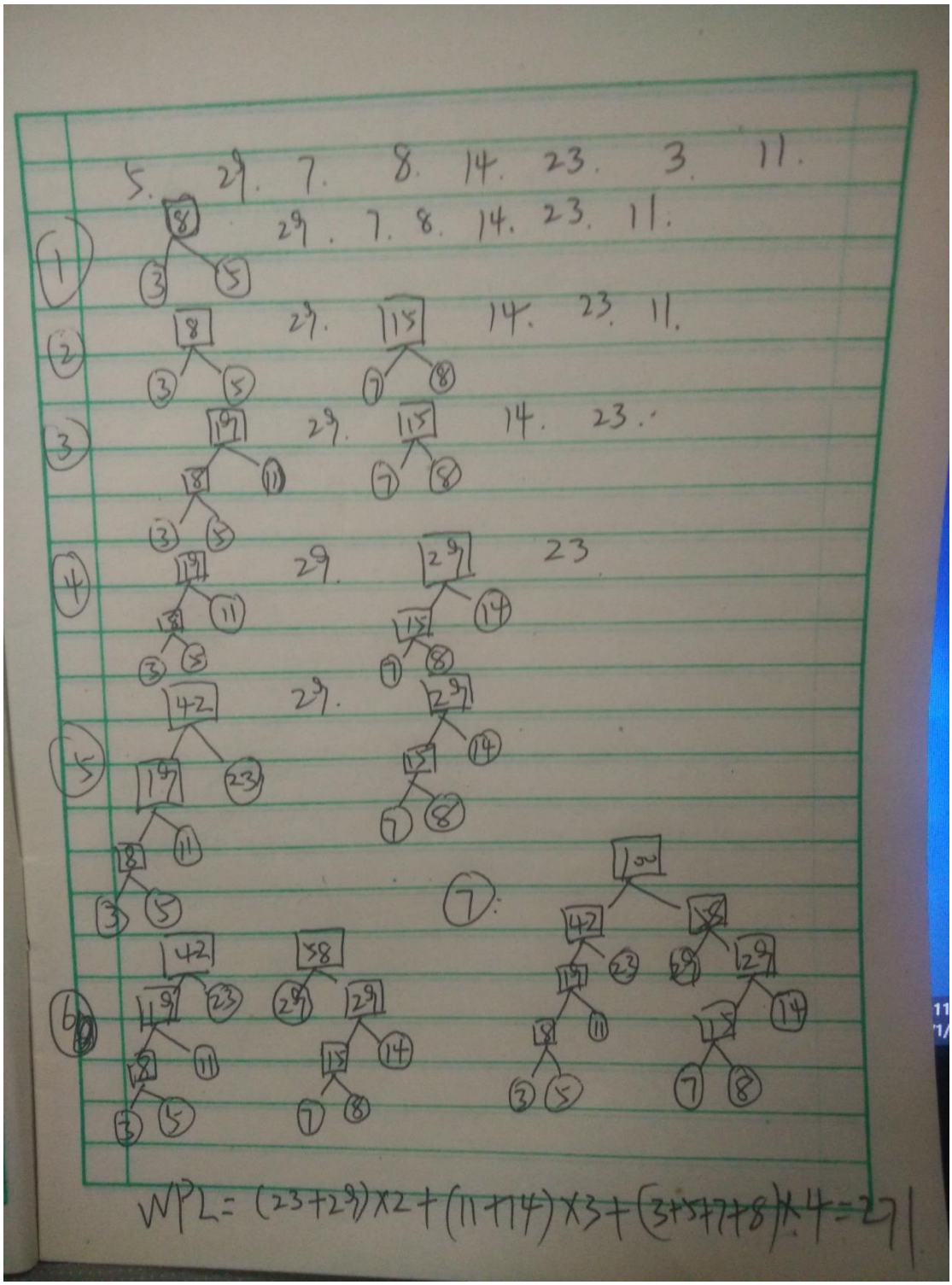
若它的左子树非空，则左子树上所有节点的值均**小于**它的根节点的值
若它的右子树非空，则右子树上所有结点的值均**大于等于**它的根节点的值
它的左、右子树也分别为二叉排序树。查找的时候，**中序遍历**二叉树，得到一个**递增**序列
关键字最大的结点可以有左子树，但**一定没有右子树**

哈夫曼树(最优二叉树)

定义

是**带权路径**(WPL)最短的树，权值越大的叶子节点越靠近根节点。

构造哈夫曼树及 WPL 计算

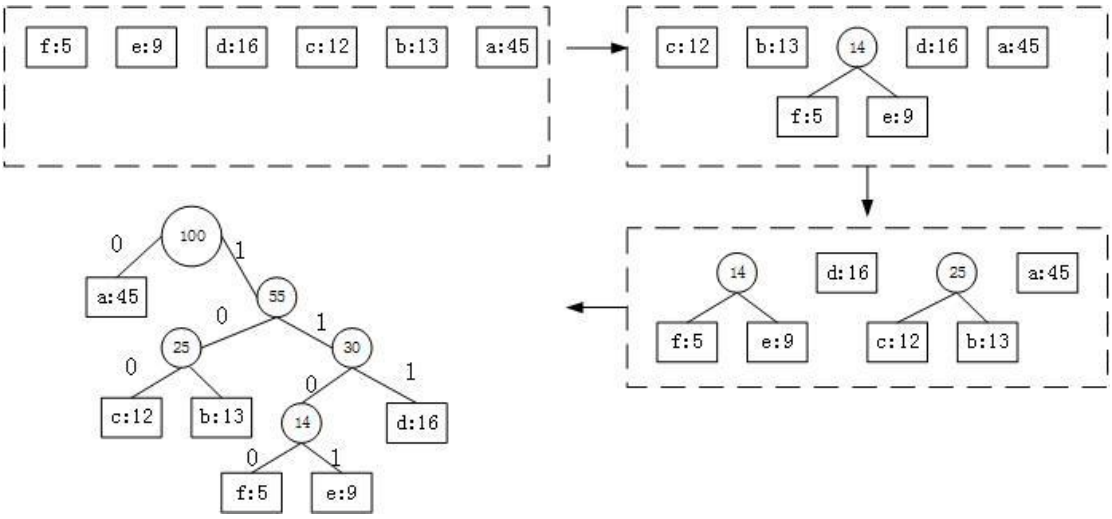


例题

已知一个文件中出现的各字符及其对应的频率如下表所示。若采用定长编码，则该文件中字符的码长应为（）。若采用 Huffman 编码，则字符序列“face”的编码应为（）。

字符	a	b	c	d	e	f
频率（%）	45	13	12	16	9	5

A．2
B．3
C．4
D．5
A．110001001101
B．001110110011
C．101000010100
D．010111101011
解析：所谓定长编码是指用多少位二进制足够表示字符，图中字符是有 6 个的，a、b、c、d、e、f，可用 000 到 101 表示 a 到 f，这样编码字符的码长可以为 3，4 位当然也是可以，但我们是找最合适的，自然 3 位能满足要求。第二问，哈夫曼树的左节点未必要比右节点小，但是通常做题时需要写成左小右大的形式，再左 0 右 1 赋值，所谓“face”编码，是指找到这 4 个字母，从根节点出发，要经历的编码数。如下图所示，所以答案为 BA



平衡二叉树

平衡二叉树（Baland Binary Tree）又被称为 AVL 树（有别于 AVL 算法），且具有以下性质：它是一棵空树或它的左右两个子树的高度差的绝对值不超过 1，并且左右两个子树都是一棵平衡二叉树。

满二叉树

除最后一层无任何子节点外，每一层上的所有结点都有两个子结点或 0 个子节点的二叉树。

查找方法

二分查找法(折半查找法)

适用情况

不经常变动而查找频繁的有序列表

优点

1、比较次数少
2、查找速度快
3、平均性能好

缺点

1、要求待查表为有序表
2、插入删除困难

实现算法

首先，假设表中元素是按升序排列，将表中间位置记录的关键字与查找关键字比较，如果两者相等，则查找成功；否则利用中间位置记录将表分成前、后两个子表，如果中间位置记录的关键字大于查找关键字，则进一步查找前一子表，否则进一步查找后一子表。重复以上过程，直到找到满足条件的记录，使查找成功，或直到子表不存在为止，此时查找不成功。

分块查找

适用情况

节点动态变化的情况

优点

比顺序查找算法（就是一个一个的去比较）快得多

缺点

速度不如折半查找法

实现算法

把一个大的线性表分解成若干块，每块中的节点可以任意存放，但块与块之间必须排序。假设是按关键码值非递减的，那么这种块与块之间必须满足已排序要求，实际上就是对于任意的 i，第 i 块中的所有节点的关键码值都必须小于第 i+1 块中的所有节点的关键码值。此外，还要建立一个索引表，把每块中的最大关键码值作为索引表的关键码值，按块的顺序存放到一个辅助数组中，显然这个辅助数组是按关键码值费递减排序的。查找时，首先在索引表中进行查找，确定要找的节点所在的块。由于索引表是排序的，因此，对索引表的查找可以采用顺序查找或折半查找；然后，在相应的块中采用顺序查找，即可找到对应的节点。

平均查找长度(E(n))

假设某个线性表中共有 n 个节点，分成大小相等的 b 块，每块有 s=n/b,则

$$E(n)= E_b +E_s=\frac{b+1}{2}+\frac{s+1}{2}=\frac{n+s^2}{2s}+1$$

排序

直接插入排序

每一趟将一个待排序的记录，按照其关键字的大小插入到有序队列的合适位置里，直到全部插入完成,比如斗地主抽牌就是这样的规则。

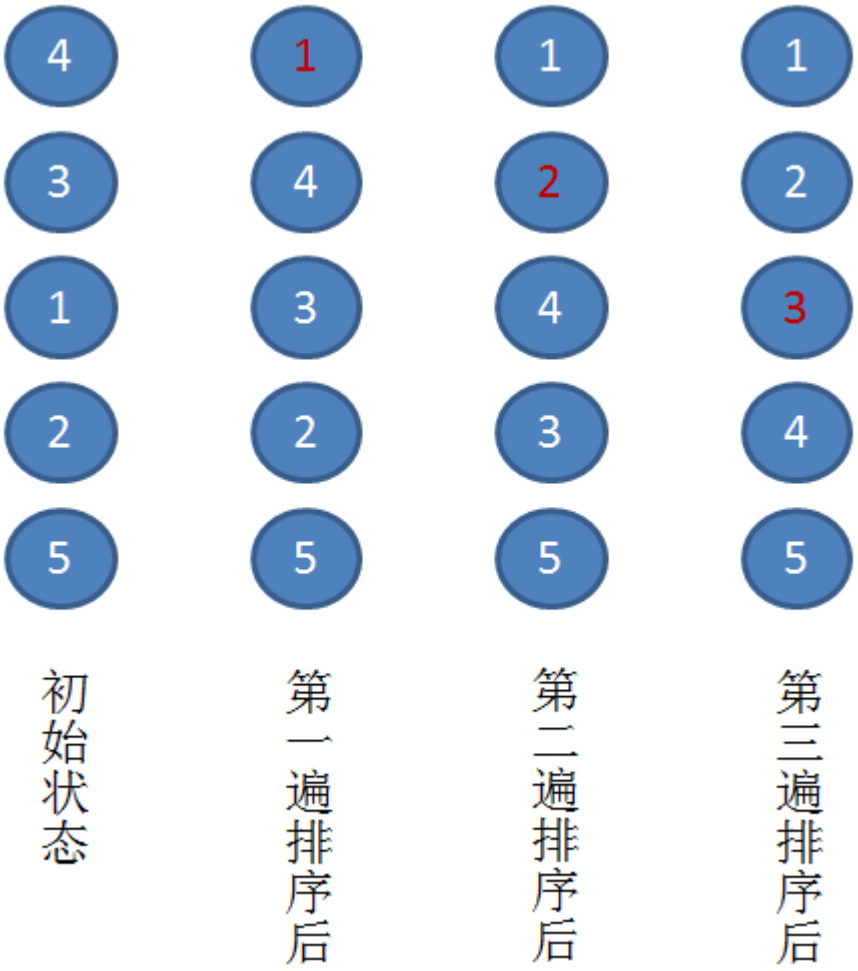
简单选择排序

每趟从待排序的记录中选出关键字最小的记录，顺序放在已排序的记录序列末尾，直到全部排序结束为止。

图-简单选择排序示例图

冒泡排序

两两比较待排序的关键字，并交换不满足次序要求的那对数，直到整个表都满足次序要求为止。



希尔排序

把记录按步长 gap 分组，对每组记录采用直接插入排序方法进行排序。随着步长逐渐减小，所分成的组包含的记录越来越多，当步长的值减小到 1 时，整个数据合成为一组，构成一组有序记录，则完成排序。

图-希尔排序示例图 Victor Zhang

快速排序

通过一趟排序将要排序的数据分割成独立的两部分：分割点左边都是比它小的数，右边都是比它大的数。然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。采用了分治法的算法策略。

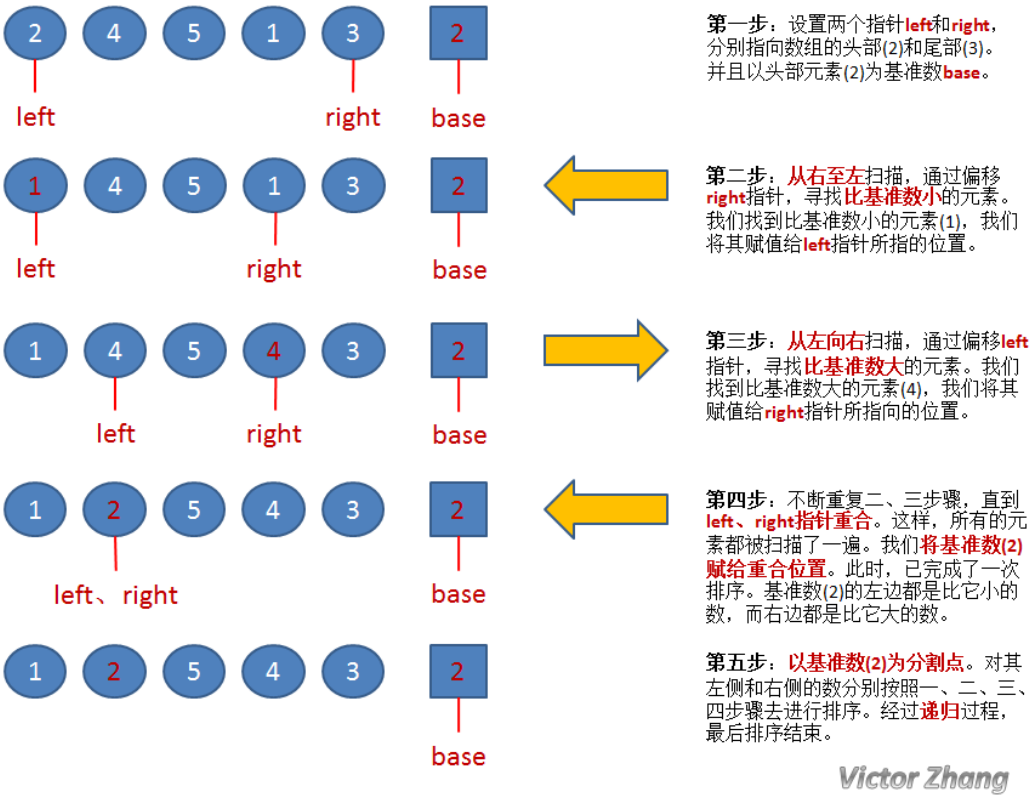


图-快速排序示例图

堆排序

堆排序中堆的定义：n 个元素的序列{k_1，k_2，…，k_n}当且仅当满足下列关系时，称为堆。

$$\begin{cases} k_i \le k_{2i} \\ k_i \le k_{2i+1} \end{cases} 或 \begin{cases} k_i \ge k_{2i} \\ k_i \ge k_{2i+1} \end{cases} \left(i = 1,2,\cdots,\left\lfloor \frac{n}{2} \right\rfloor\right)$$

归并排序

将待排序序列 R[0...n-1]看成是 n 个长度为 1 的有序序列，将相邻的有序表成对归并，得到 n/2 个长度为 2 的有序表；将这些有序序列再次归并，得到 n/4 个长度为 4 的有序序列；如此反复进行下去，最后得到一个长度为 n 的有序序列。

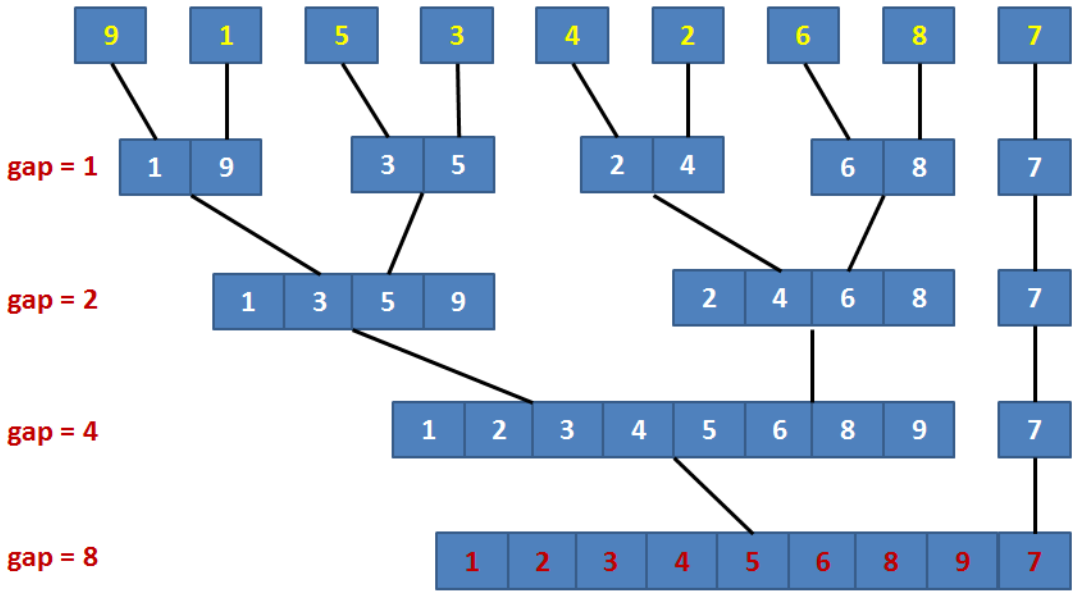


图-归并排序示例图

基数排序

基数排序与本系列前面讲解的七种排序方法都不同，**它不需要比较关键字的大小**。它是根据关键字中各位的值，通过对排序的 N 个元素进行若干趟“分配”与“收集”来实现排序的。
设有一个初始序列为: R {50, 123, 543, 187, 49, 30, 0, 2, 11, 100}。
我们知道，任何一个阿拉伯数，它的各个位数上的基数都是以 0~9 来表示的。所以我们不妨把 0~9 视为 10 个桶。 我们先根据序列的个位数的数字来进行分类，将其分到指定的桶中。例如：R[0] = 50，个位数上是 0，将这个数存入编号为 0 的桶中。

[0]	50	30	0	100
[1]	11			
[2]	2			
[3]	123	543		
[4]				
[5]				
[6]				
[7]	187			
[8]				
[9]	49			

分类后，我们在从各个桶中，将这些数按照从编号 0 到编号 9 的顺序依次将所有数取出来。这时，得到的序列就是个位数上呈递增趋势的序列。 按照个位数排序： {50, 30, 0, 100, 11, 2, 123, 543, 187, 49}。

排序的比较

排序方法	最好时间复杂度	平均时间复杂度	最坏时间复杂度	辅助空间	稳定性
直接插入	O(n)	$O(n^2)$	$O(n^2)$	O(1)	稳定
简单选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)	不稳定
冒泡排序	O(n)	$O(n^2)$	$O(n^2)$	O(1)	稳定
希尔排序	不存在	$O(n^{1.3})$	不存在	O(1)	不稳定
快速排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$	$O(\log_2 n)$	不稳定
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	O(1)	不稳定
归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	O(n)	稳定
基数排序	O(d(n+rd))	O(d(n+rd))	O(d(n+rd))	O(rd)	稳定

例题

堆是一种数据结构，(34)是堆排序

A．(10， 50， 80， 30， 60， 20， 15， 18)
B．(10，18，15，20，50，80，30，60)
C．(10，15，18，50，80，30，60，20)
D．(10，30，60，20，15，18，50，80)

根据定义可知选 B

广义表

广义表的长度是将最外面那层的括号删了以后所剩下的元素(组)个数，深度是括号的层数

例题

L1=((a,(a,b),((a,b),c)))，L2=((1，2，3))，L3=(1，2，3)。求 L1、L2、L3 的长度和深度
答案：

	长度	深度
L1	1	4
L2	1	2
L3	3	1

归并排序的归并路数

归并路数 =|logkm|，其中 m 为元素个数，k 为多路归并趟数

例题

若对 27 个元素只进行三趟多路归并排序，则选取的归并路数为 (37) 。

A. 2 B. 3 C. 4 D. 5
根据公式可得 log 以 3 为底，以 27 为真数的答案为 3。所以选 B

表达式的记法

前缀表达式（前缀记法、波兰式）

从右至左扫描表达式，遇到数字时，将数字压入堆栈，遇到运算符时，弹出栈顶的两个数，用运算符对它们做相应的计算（栈顶元素 op 次顶元素），并将结果入栈；重复上述过程直到表达式最左端，最后运算得出的值即为表达式的结果。
例如前缀表达式“- × + 3 4 5 6”：
(1) 从右至左扫描，将 6、5、4、3 压入堆栈；
(2) 遇到+运算符，因此弹出 3 和 4（3 为栈顶元素，4 为次顶元素，注意与后缀表达式做比较），计算出 3+4 的值，得 7，再将 7 入栈；
(3) 接下来是×运算符，因此弹出 7 和 5，计算出 7×5=35，将 35 入栈；
(4) 最后是-运算符，计算出 35-6 的值，即 29，由此得出最终结果。

中缀表达式

我们平常用的表达式 a+b-c 这样的就是中缀表达式

后缀表达式（后缀记法、逆波兰式）

从左至右扫描表达式，遇到数字时，将数字压入堆栈，遇到运算符时，弹出栈顶的两个数，用运算符对它们做相应的计算（次顶元素 op 栈顶元素），并将结果入栈；重复上述过程直到表达式最右端，最后运算得出的值即为表达式的结果。
例如后缀表达式“3 4 + 5 × 6 -”：
(1) 从左至右扫描，将 3 和 4 压入堆栈；
(2) 遇到+运算符，因此弹出 4 和 3（4 为栈顶元素，3 为次顶元素，注意与前缀表达式做比较），计算出 3+4 的值，得 7，再将 7 入栈；
(3) 将 5 入栈；
(4) 接下来是×运算符，因此弹出 5 和 7，计算出 7×5=35，将 35 入栈；
(5) 将 6 入栈；
(6) 最后是-运算符，计算出 35-6 的值，即 29，由此得出最终结果。

系统基础

符号数

原码

正数的原码等于自身的二进制数，负数的原码第一位为 1(符号位，表示负数)，后面为自身的二进制数

反码

正数的反码等于自身的二进制数，负数的反码符号位不动，其余各位按位取反

补码

正数的补码等于自身的二进制数，负数的补码是在反码的基础上+1

移码(增码)

无论正负数，只要将其**补码**的符号位取反即可

符号数的应用

在计算机中，最适合**数字加减**运算的数字编码是**补码**，最适合表示**浮点数阶码**的数字编码是**移码**。

定点数

所谓定点数，就是小数点的位置固定不变的数。小数点的位置通常有两种约定形式：定点整数（纯整数，小数点在最低有效数值位之后）和定点小数（纯小数，小数点在最高有效数值位之前）。

机器字长为 n 时各种码制表示的带符号数的范围

码制	定点整数	定点小数
原码	$[-(2^{n-1}-1), 2^{n-1}-1]$	$[-(1-2^{-(n-1)}), 1-2^{-(n-1)}]$
反码	$[-(2^{n-1}-1), 2^{n-1}-1]$	$[-(1-2^{-(n-1)}), 1-2^{-(n-1)}]$
补码	$[-2^{n-1}, 2^{n-1}-1]$	$[-1, 1-2^{-(n-1)}]$
移码	$[-2^{n-1}, 2^{n-1}-1]$	$[-1, 1-2^{-(n-1)}]$

记忆技巧

当 $A=2^{n-1}$，$B=1-2^{-(n-1)}$时，则有以下规律

码制	定点整数	定点小数
原码	$[-(A-1), A-1]$	$[-B, B]$
反码	$[-(A-1), A-1]$	$[-B, B]$
补码	$[-A, A-1]$	$[-1, B]$
移码	$[-A, A-1]$	$[-1, B]$

计算机指令系统

立即寻址　　：操作数包含在指令中，获取操作数是最快的
直接寻址　　：操作数的地址在指令中
寄存器寻址　：操作数在寄存器中
寄存器间接寻址：操作数的地址在寄存器中

中央处理器

CPU 的组成

运算器、控制器、寄存器和内部总线，其中控制器不仅要保证程序的正确执行，而且要能够处理异常事件。

存储系统

存储器的分类

按位置分类

内存（主存）、外存（辅存）

按材料分类

磁存储器、半导体存储器、光存储器

按工作方式分类

读写存储器、只读存储器

按访问方式分类

按地址访问的存储器、按内容访问的存储器（相联存储器）

按寻址方式分类

随机存储器、顺序存储器、直接存储器

软件测试

测试阶段划分

单元测试（模块测试）

一般是**在编程阶段完成**，由程序员对自己编写的模块**自行测试**，检查模块是否实现了**详细设计说明书**中规定的**功能**和**算法**，通常使用**白盒测试**。
单元测试计划应该在详细设计阶段制定。
单元测试期间着重从：模块接口、局部数据结构、重要的执行通路、出错处理、边界条件这几个方面对模块进行测试。

集成测试（组装测试）

主要目标是发现模块间的**接口**和**通信**问题。集成测试主要发现**概要设计阶段产生的错误**，通常采用**黑盒测试**。**集成测试计划应该在概要设计阶段制定**。**集成的方式可分为非增殖式和增殖式**。

确认测试

检查软件的功能、性能和其他特征是否与用户的需求一致。它是以**需求规格说明书**作为依据的测试，通常采用**黑盒测试**。软件确认测试首先要进行有效性测试以及软件配置审查，然后进行验收测试。

确认测试一般有以下三个步骤：

（1） 有效性测试

（2） 软件配置审查

（3） 验收测试

α 测试与 β 测试（当一个软件是作为产品被许多客户使用时需要用这种测试）

系统测试

系统测试的任务是把软件放在实际的硬件和网络环境中进行测试，主要测试软件的非功能需求和质量属性是否得到满足。系统测试是根据**系统方案说明书**来设计测试用例，通常采用**黑盒测试**。常见的系统测试有：**恢复测试、安全性测试、强度测试、性能测试、可靠性测试和安装测试**。在已确认的计算机软硬件环境下，通过与系统需求对比，发现系统与用户需求不符或矛盾的地方。

回归测试

在软件发生变更后进行的测试，以发现变更时引起的其他错误

白盒测试

语句覆盖

使被测程序中的每条语句至少执行一次

判定覆盖（分支覆盖）

使被测程序中的每个判定表达式至少获得一次“真”值和“假”值

条件覆盖

使被测程序中的每个逻辑条件的各种可能的值至少满足一次

判定/条件覆盖

使得判定中的每个条件的“真”值和“假”值至少出现一次，并使本身判定结果的“真”值和“假”值至少出现一次

条件组合覆盖

使得每个判定中条件的各种可能值的组合都至少出现一次。满足条件组合覆盖的测试用例是一定满足判定覆盖、条件覆盖和判定/条件覆盖的

路径覆盖

覆盖被测试程序中所有可能的路径

面向对象测试

以下四个层次由低到高的顺序排列
(1)测试与对象关联的单个操作，即算法层。
(2)测试单个对象类，类层。
(3)测试对象集群，模板层
(4)测试面向对象系统，系统层。

编译原理

校验码

奇偶校验

通常用于对少量数据的校验

奇校验

将信息数据的各位进行模二加法并作为校验码的称为奇校验。

偶校验

将信息数据的各位进行模二加法并**取反**作为校验码的称为偶校验。

海明码

采用多位校验码的方式，**可以发现、纠正错误**。数据位和校验位必须满足关系式：$2^{校验位}-1\geq$数据位+校验位。码距至少是 **3**。

循环冗余校验码

检错能力非常强，但是**不能纠错**。编码长度(CRC 字长)为数据位+校验位

文法

终结符和非终结符

文法格式通常为：α→β，若字符为大写字母，则是非终结符，若字符为小写字母，则是终结符

文法的分类

0 型文法(短语文法)

设 G=（VN，VT，P，S），如果它的每个产生式 α→β 是这样一种结构： α∈(VN∪VT)*且**至少含有一个非终结符**，而 β∈(VN∪VT)*，则 G 是 一个 0 型文法。一个非常重要的理论结果是:0 型文法的能力相当于图灵机(Turing)。或者说，**任何 0 型文语言都是递归可枚举的，反之，递归可枚举集必定是一个 0 型语言**。0 型文法是这几类文法中，限制最少的一个，所以我们在试题中见到的,至少是 0 型文法。

1 型文法(上下文有关文法)

此文法对应于**线性有界自动机**。它是在 0 型文法的基础上每一个 α→β,都有|β|>=|α|。这里的|β|表示的是 β 的长度。

注意：虽然要求|β|>=|α|，但有一特例： α→ε 也满足 1 型文法。

如有 A->Ba 则|β|=2,|α|=1 符合 1 型文法要求。反之,如 aA->a，则不符合 1 型文法。

2 型文法(上下文无关文法)

此文法对应于**下推自动机**。2 型文法是在 1 型文法的基础上,再满足:**每一个 α→β 都有 α 是非终结符**。如 A->Ba,符合 2 型文法要求。**大多数程序设计语言的语法规则可以用上下文无关文法描述**

如 Ab->Bab 虽然符合 1 型文法要求,但不符合 2 型文法要求,因为其 α=Ab,而 Ab 不是一个非终结符。

3 型文法(正规文法)

此文法对应于**有限状态自动机**。它是在 2 型文法的基础上满足:A→α|αB(右线性)或 A→α|Bα(左线性)。

如有:A->a,A->aB,B->a,B->cB,则符合 3 型文法的要求。但如果推导为:A->ab,A->aB,B->a,B->cB 或推导为:A->a,A->Ba,B->a,B->cB 则不符合 3 型方法的要求了。具体的说,例子 A->ab,A->aB,B->a,B->cB 中的 A->ab 不符合 3 型文法的定义,如果把后面的 ab,改成“一个非终结符+一个终结符”的形式(即为 aB)就对了。例子 A->a,A->Ba,B->a,B->cB 中如果把 B->cB 改为 B->Bc 的形式就对了,**因为 A→α|αB(右线性)和 A→α|Bα(左线性)两套规则不能同时出现在一个语法中**,只能完全满足其中的一个,才能算 3 型文法。

数据流图(Data Flow Diagram,DFD)

在面向数据流的设计方法中,一般把数据流图中的数据流划分为两种类型,一种是**变换流**,一种是**事务流**。DFD 由**数据流、加工、数据存储和外部实体** 4 个要素构成。

编译过程

词法分析

从左到右逐个字符地扫描,从中识别出一个个“单词”符号。“单词”符号是程序设计语言的基本语法单位,如**关键字、标识符、常数、运算符和分隔符**等。

语法分析

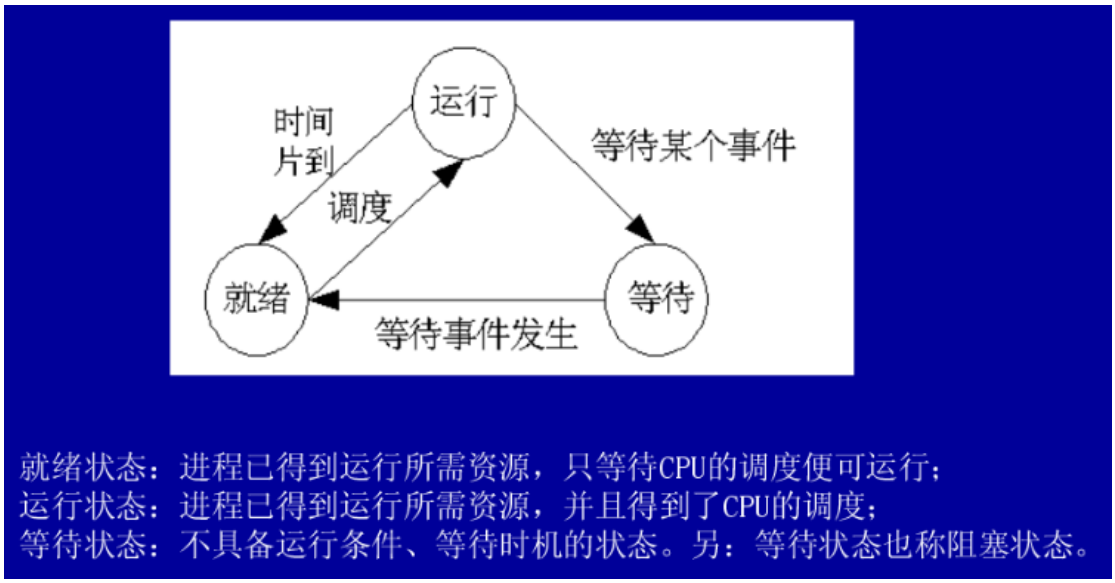
根据语言的语法规则将单词符号序列分解成各类语法单位,比如**表达式、语句和程序**等。语法规则就是各类语法单位的构成规则。通过语法分析确定整个输入串是否构成一个语法上正确的程序。

语义分析

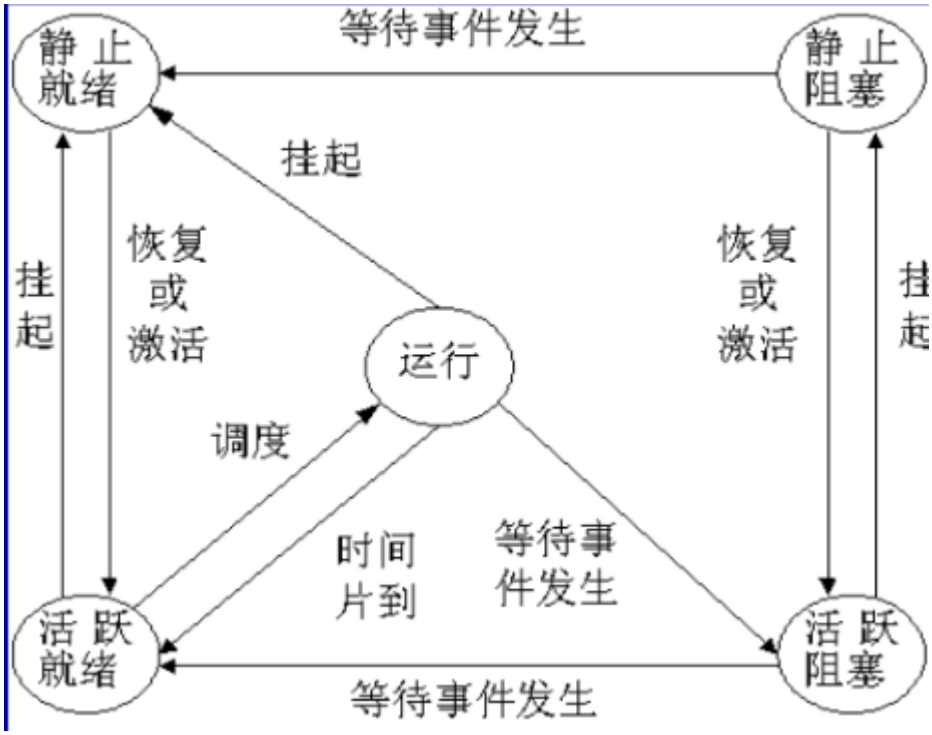
检查源程序是否包含静态语义错误，并收集类型信息供后面的代码生成阶段使用。只有语法和语义都正确的源程序才能被翻译成正确的目标代码。
语义分析的一个主要工作是进行类型分析和检查。程序语言中的一个数据类型一般包含两个方面的内容：类型的载体及其上的运算。例如：整除取余运算只能对整型数据进行运算，若其运算对象中有浮点数就认为是类型不匹配的错误。**静态的语义错误是指编译程序可以发现，动态的语义错误是指源程序虽然能够被编译和执行，但是结果不对，一般是逻辑上的错误。**

进程

进程的三态图



进程的五态图



PV 操作

进入临界区时执行 P 操作（申请），退出临界区时执行 V 操作（释放）

操作系统

内存编址

已知主存容量为 16MB,且按①,求该主存地址至少需要多少位

当①的内容为“字节编址”时

字节编址就是 8 位的意思,所以 16MB=(16*1024)KB=(16*1024*1024)byte=$2^4*2^{10}*2^{10}=2^{24}$ byte.
所以需要 24 位

当①的内容为“4 位编址”时

先将编址方式凑成 8 位，即 2*4 位，相应的主存容量也扩充 2 倍为 32MB，所以

32MB=(32*1024)KB=(32*1024*1024)byte=2^5*2^{10}*2^{10}=2^{25}byte.所以需要 25 位

例题

若内存按字节编址，用存储容量为 32K×8 比特的存储器芯片构成地址编号 A0000H～DFFFFH 的内存空间，则至少需要_____片。

1、A. 4　　B. 6　　C. 8　　D. 10
空间大小为 DFFFF-A0000+1=262144byte=256kb。
组成内存储器的芯片数量=内存储器的容量/单个芯片的容量=(256kb*8b)/(32k*8b)=8，所以选 C

指令运行参数

设定变量 T 为指令运行总时间，t 为所需时间最长部分指令的时间（周期），n 为指令条数

指令相关公式

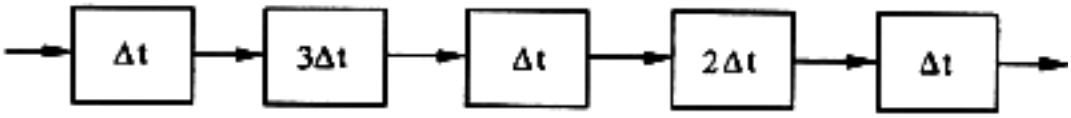
顺序方式运行指令所需时间：T*n
流水方式运行指令所需时间：T+(n-1)t
重叠方式运行指令所需时间：（n+2）*t
吞吐率：n/流水方式运行指令所需时间
效率：效率=吞吐率*t
加速比：加速比=效率*n

例题

若指令流水线把一条指令分为取指、分析和执行三部分，且三部分的时间分别是 2ns,2ns,1ns，则 100 条指令全部执行完毕所需的时间是多少 ns
答：因为指令运行时间=T+(n-1)t，所以(2+2+1)+(100-1)*2=203。注意，若选择题中没有相应的选择，则将每段指令所需时间均设置为最长时间，就本题而言，应该是 2ns,2ns,2ns。(因为目前对于指令运行时间的算法没有统一)

若每一条指令都可以分解为取指、分析和执行三步。已知取指时间 t 取指=5Δt，分析时间 t 分析=2Δt，执行时间 t 执行=5Δt。如果按顺序方式从头到尾执行完 500 条指令需_____Δt。如果按照[执行] k、分析 k+1、[取指] k+2 重叠的流水线方式执行指令，从头到尾执行完 500 条指令需_____Δt。
4、A. 5590　　B. 5595　　C. 6000　　D. 6007
5、A. 2492　　B. 2500　　C. 2510　　D. 2515
第一个空符合顺序方式，则时间为 T*n=12*500=6000，第二个空符合重叠的流水线方式，则时间为（n+2）*t=(500+2)*5=2510，所以选 C，C

某指令流水线由 5 段组成，各段所需要的时间如下图所示。连续输入 10 条指令时的吞吐率为_____　。



6、A. 10/70Δt　　B. 10/49Δt　　C. 10/35Δt　　D. 10/30Δt

吞吐率= n/ T+(n-1)t=10/((1+3+1+2+1)+9*3)= 10/35Δt，所以选 C

内存管理

分配方案的比较

分配办法	单一连续分配	固定分区分配	可变分区分配
分配类型	静态分配法	静态分配法	动态分配法
分配特点	不分区，所有用户空间给某个进程或作业	分成大小不等的区域，区域分完后固定不变	分成大小不等的区域，根据用户要求动态分配

可变分区分配算法

首次适应法

从主存低地址开始，寻找第一个可用（即大于等于作业需求的内存）的自由区，这种方法可实现快速分配，缩短查找时间。

循环适应法

是首次适应法的一个变种，也就是不再是每次都从头开始匹配，而是连续向下匹配。

最佳适应法

选择最接近作业需求的内存自由区进行分配。这种方法可以减少碎片，但同时也可能带来更多小得无法再用的碎片。

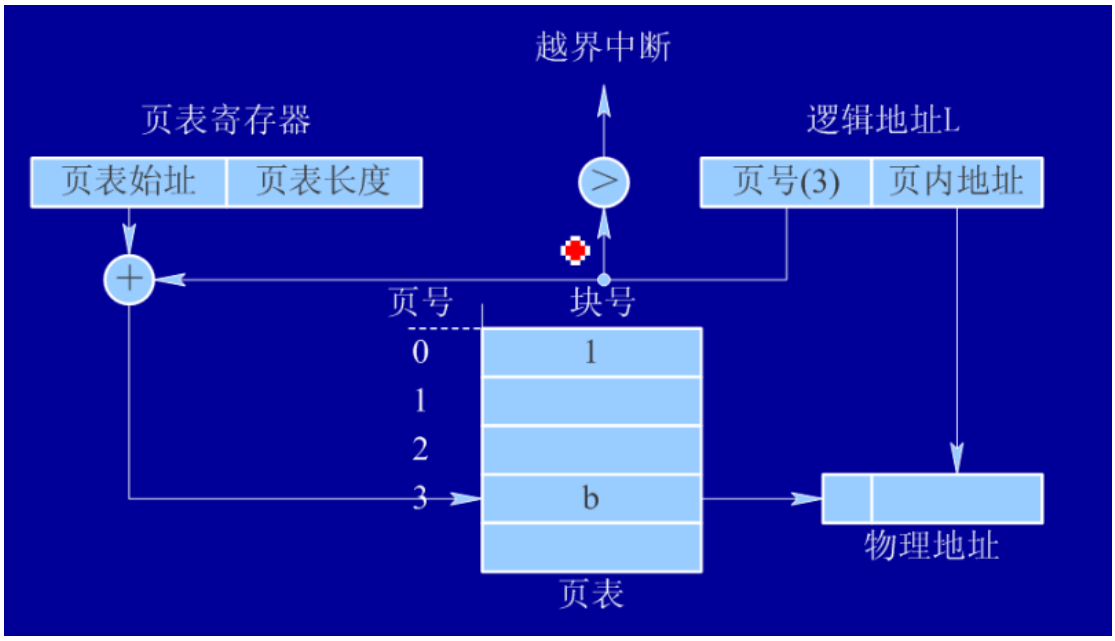
最差适应法

选择整个主存中最大的内存自由区。

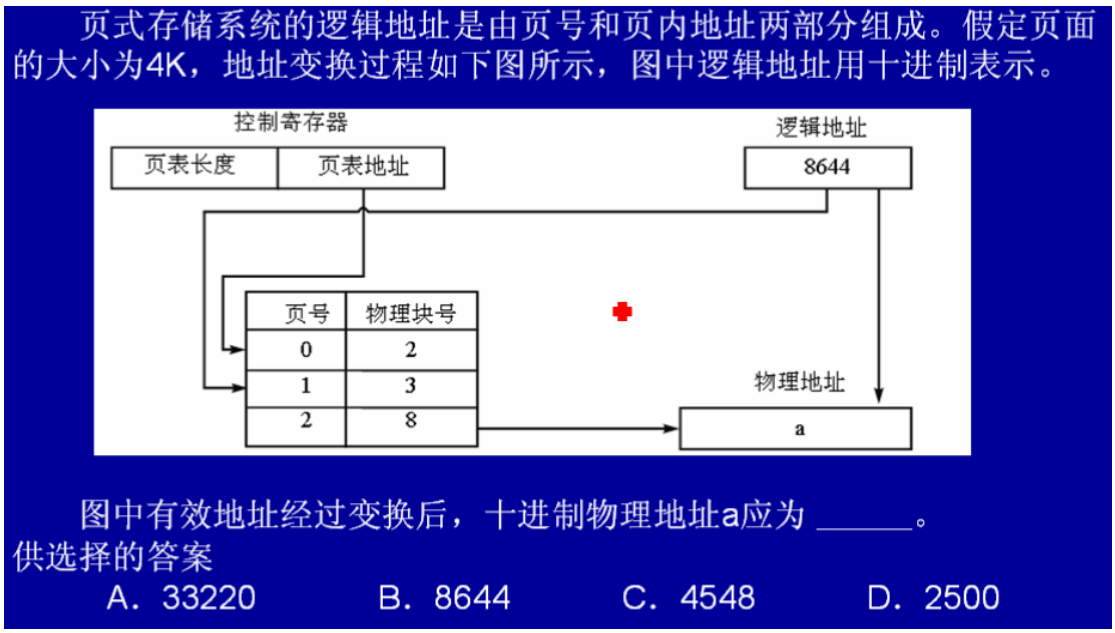
虚存管理

页式存储

题目往往是给出下图求物理地址，首先，我们知道所谓的逻辑地址是页号+页内地址，页内地址=页面大小位数。具体的做法是：将逻辑地址转换为二进制，从右边开始数页内地址位数，剩下的左边二进制即为页号。根据图片找到对应的块号，则物理地址为块号(二进制)+页内地址。

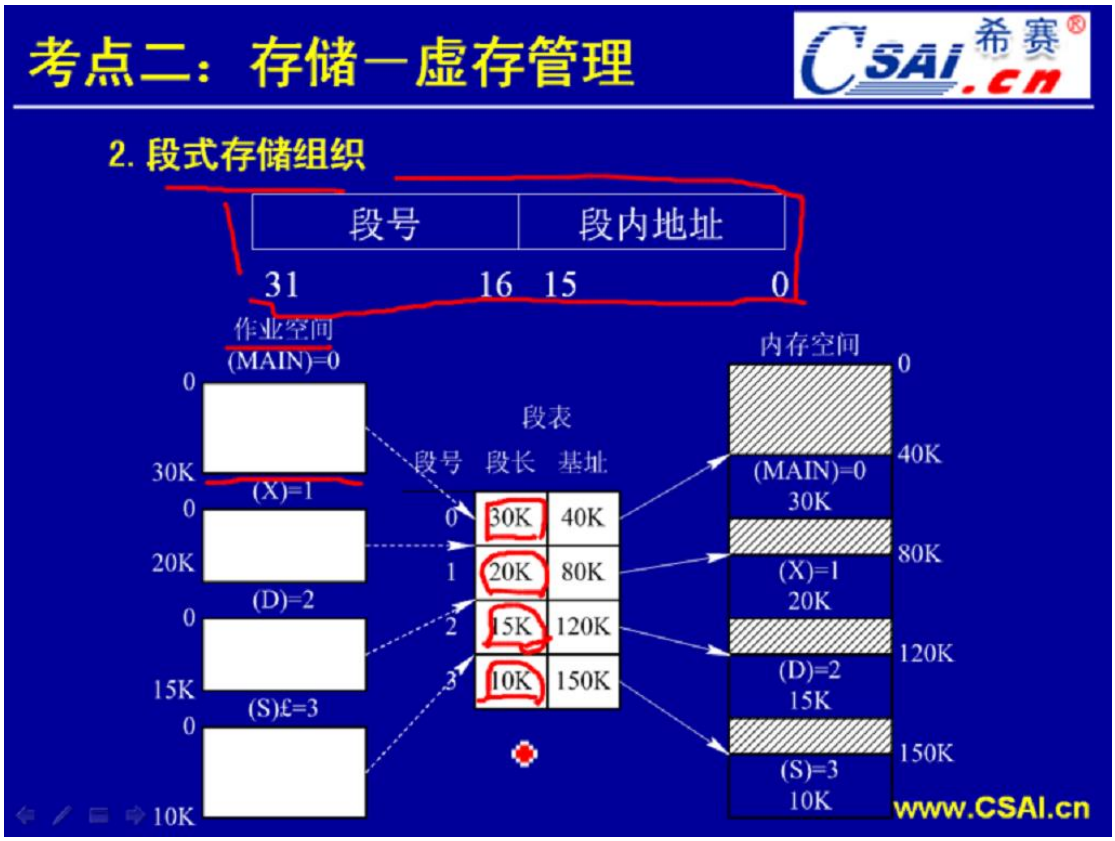


例题



此题考查的是虚拟存储中的页式存储，题目已知页面大小为4K，因为4K=2^{12}，所以页内地址有12位。现在把逻辑地址8644转成二进制，得10 0001 1100 0100，其中的低12位为页内偏移量，最高两位则为页号，所以逻辑地址8644的页号为10，即十进制的2，所以物理块号为8，化为二进制得1000。把物理块号和页内偏移地址拼合得1000 0001 1100 0100，化为十进制得33220。所以正确答案是A。

段式存储



不同管理方式之间的优缺点

	优点	缺点
段式存储	便于多道程序共享内存，便于对存储器的保护，各段程序修改互不影响	内存利用率低，内存碎片浪费大
页式存储	利用率高，产生的内存碎片小，内存间分配及管理简单	要有相应的硬件支持，增加了系统开销，请求调页的算法如选择不当，有可能产生抖动现象
段页式存储	空间浪费小，存储共享容易、	由于管理软件的增加，复杂

	存储保护容易、能动态连接	性和开销也随之增加，需要的硬件以及占用的内容也有所增加，使得执行速度大大下降

页面置换算法

最优置换算法

后续页面最先被访问的页面保留，淘汰那些很久才会被使用的页面

先进先出算法

优先淘汰最先进入的页面

最近最少使用算法

优先淘汰存在最久的页面

局部性原理

时间局部性

一个页面被访问后，很有可能再次被访问

空间局部性

一个页面被访问后，很有可能访问其周围的页面

作业管理

作业时间

作业周转时间

T=作业完成时间-作业提交时间
T=作业等待时间+作业运行时间

作业平均周转时间

T 平均=（T1+T2+……Tn）/n

带权作业周转时间

单个作业带权周转时间

W=作业周转时间/作业实际运行时间

作业平均带权周转时间

W=（W1+W2+……+Wn）/n

作业调度算法

先来先服务

按作业序号依次运行

最短作业优先

优先运行所需运行时间最短的作业

优先数

优先级高优先运行

定时轮转

规定一个时间片大小，作业按序号依次运行，每次运行时间与时间片大小等同，到期后继续运行下一个时间片长度的任务，重复步骤。

Cache（高速缓存）

Cache 的读写过程

写直达

当要写 Cache 时，数据同时写回主存

写回

当相应数据从 Cache 中被淘汰时写回主存

标识法

取数据时添加一个有效位为 1，当修改主存数据后设置有效位为 0.

地址映像

直接映像

主存与 Cache 的映像

主存的每一页与 Cache 的每一页相同大小，因为主存比 Cache 大得多，所以尽管每一页相等，主存的总页数远大于 Cache 的总页数。因此，根据 Cache 的页数大小，主存将相同分为多个组，每个组的页数与 Cache 总页数相等，其中的每一页正好配对。编号不一致的页是不能相互映像的。

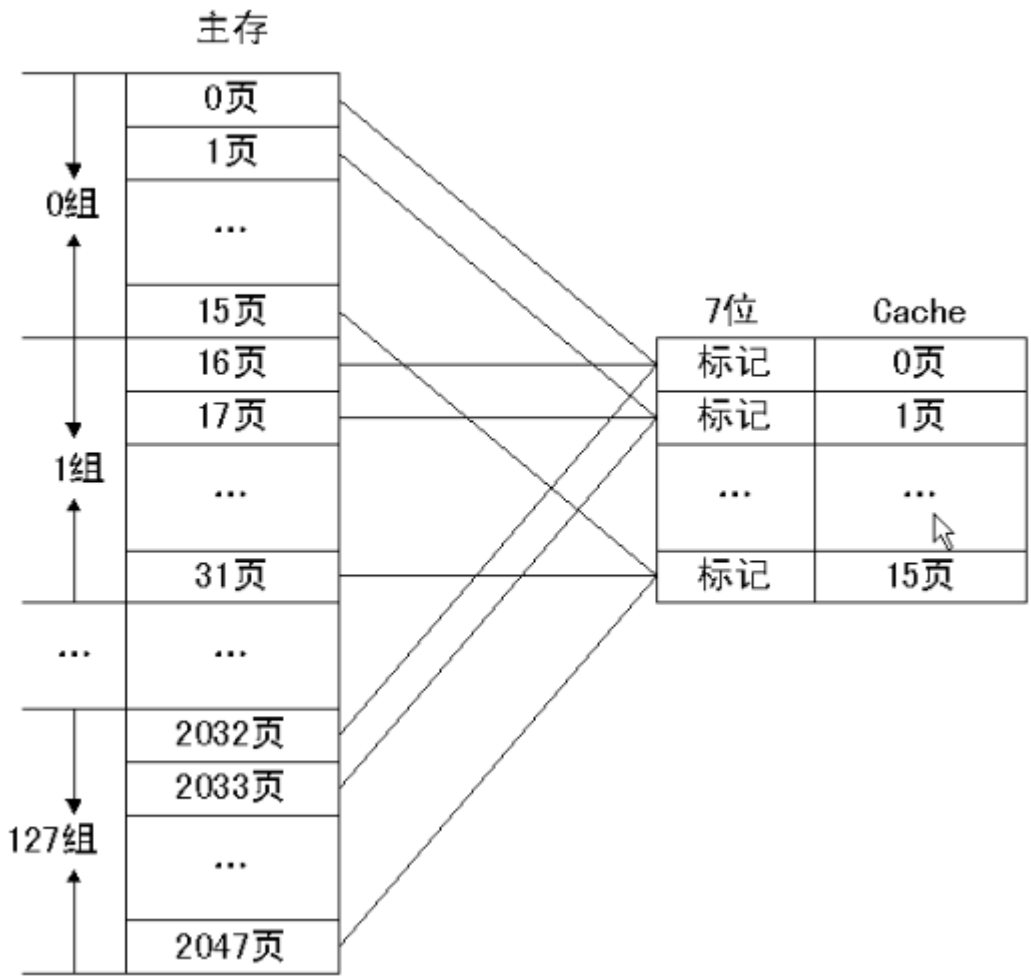
优缺点

优点：地址变换简单。
缺点：每块相互对应，不够灵活。

主存与 Cache 的地址组成

主存：区号+块号+块内地址
Cache：块号+块内地址

示意图



全相联映像

主存与 Cache 的映像

将主存与 Cache 划分成若干个大小相等的块，主存中每一块都可以调到 Cache 中的每一块。

特点

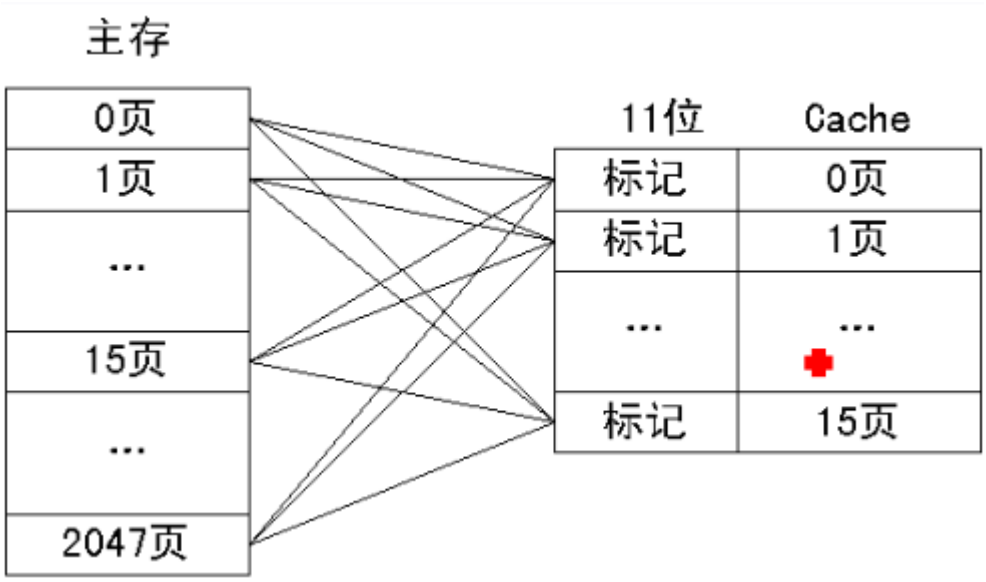
优点：访问灵活，冲突率低，只有 Cache 满时才会出现在冲突。缺点：地址变换比较复杂，速度相对慢。

主存与 Cache 的地址组成

主存：块号+块内地址

Cache：块号+块内地址

示意图



组相联映像

主存与 Cache 的划分

主存：主存根据 Cache 大小划分成若干个区，每个区内划分成若干个组，每个组再划分成若干个块。

Cache：划分成若干个组，每个组划分成若干个块。

主存与 Cache 的映像

主存的每个分区与 Cache 采用直接映像，主存的每个组之内采用全相联映像。

特点

融合了直接映像与全相联映像两种映像方式，结合了两者的优据点。具体实现容易，命中率

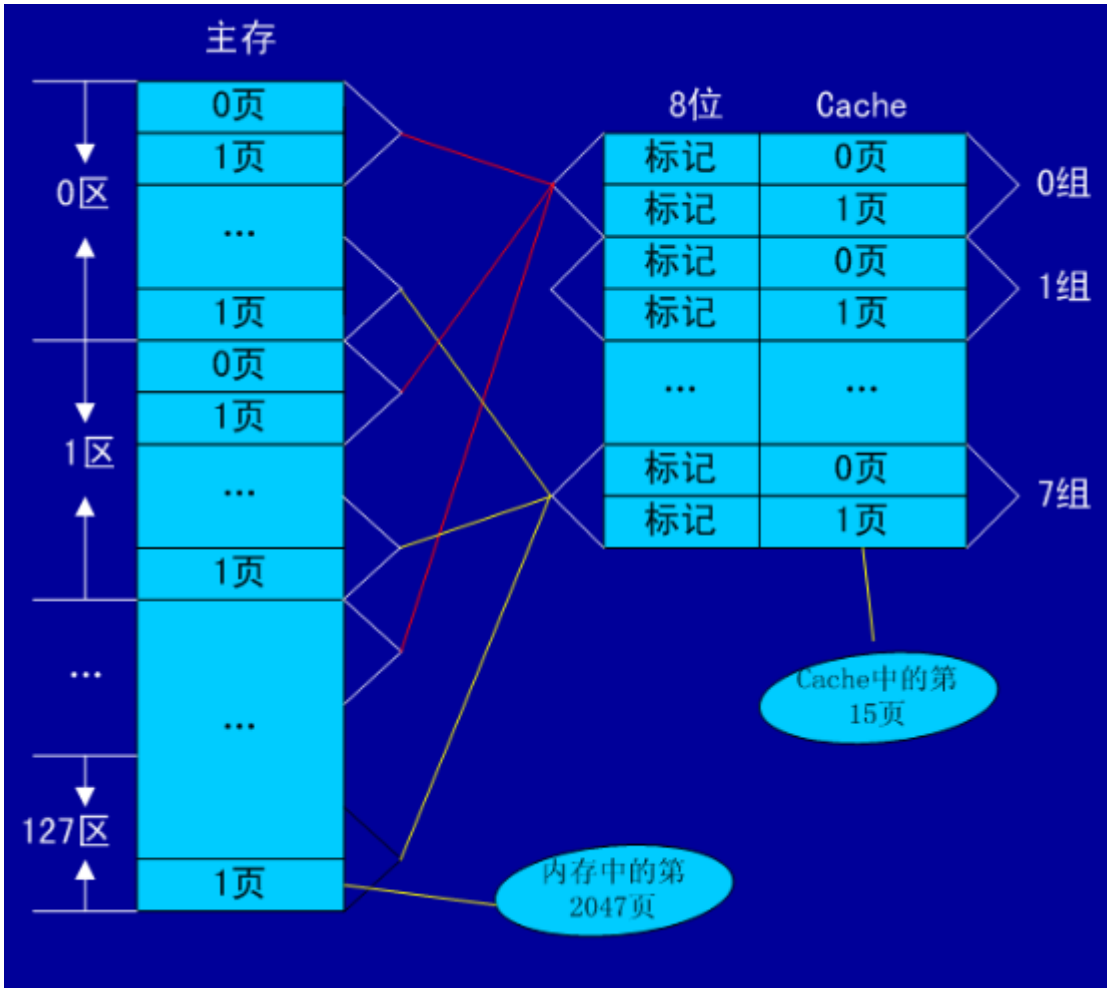
与全相联映像接近。

主存与 Cache 的地址组成

主存：区号+组号+块号+块内地址

Cache：组号+块号+块内地址

示意图



块冲突次数比较

发生块冲突次数从大到小排序为：直接映像>组相联映像>全相联映像

例题

某 32 位计算机的 cache 容量为 16KB，cache 块的大小为 16B，若主存与 cache 的地址映射采用直接映射方式，则主存地址为 1234E8F8（十六进制）的单元装入的 cache 地址为____。

A. 00 0100 0100 1101 (二进制)
B. 01 0010 0011 0100 (二进制)
C. 10 1000 1111 1000 (二进制)
D. 11 0100 1110 1000 (二进制)

cache 的容量为 16KB，块大小为 16B，因此一共有 16KB/16B=1024 个页，所以需要 10 位来存储（2^{10}=1024）。块所需要的存储地址位数为 4 位（2^4=16）。因为采用的是直接映射方式，所以主存与 cache 的地址关系为：

主存：区号+块号+块内地址

Cache：块号+块内地址

所以，cache 地址一共是 14 位，将主存地址转换为二进制后最后 14 位就是 Cache 地址了，选 C

“Cache+主存储器”系统的平均周期

若 t1 为 Cache 的周期时间，t2 为主存储器周期时间，h 为 Cache 的访问命中率，则系统的平均周期为 h*t1+(1-h)*t2

命中率

Cache 由两部分组成：控制部分和 Cache 存储器部分。控制部分的功能是判断 CPU 要访问的信息是否在 Cache 存储器中，若在即为命中，若不在则没有命中。命中时直接对 Cache 存储器寻址；未命中时，要按照替换原则决定主存的一块信息放到 Cache 存储器的哪一块里。

Cache 命中率=（平均存取时间-主存存取时间）/（高速缓存存取时间-主存存取时间）

例题

高速缓存 cache 与主存间采用全相联地址映像方式，高速缓存的容量为 4MB，分为 4 块，每块 1MB，主存容量为 256MB。 若主存读写时间为 30ns，高速缓存的读写时间为 3ns，平均读写时间为 3.27ns，则该高速缓存的命中率为_____

由公式可得命中率为 99%

总线系统的数据传送速率

计算公式

公式 Q=W*F/N，其中 Q 为总线数据传输率，W 为总线数据宽度(总线位宽/8)，F 为总线工作频率，N 为完成一次数据传送所需的总线时钟周期个数。

例题

若总线位宽为 16 位，总线工作频率为 8MHZ，完成一次数据传送需 2 个总线时钟周期，则总线的数据传输速率为多少？
Q 的计算公式为：((16/8)*8M)/2=8MB/s

中断响应时间

从发出中断请求到进入中断处理所用的时间

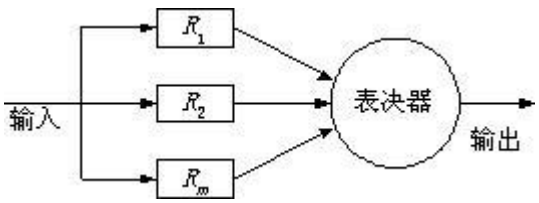
系统的可靠性

设 p1,p2 为 2 个部件的可靠度

串联与并联系统

若为并联，则可靠度=1-(1-p1)(1-p2)
若为串联，则可靠度=p1p2

N 模冗余系统



N 模冗余系统是当有一定量的部件做出相同的处理结果时，则输出该结果，概念太复杂，懒得写了，直接举例子吧，如上图所示，若该 3 个部件中有 2 个部件正常运行，则输出结果，因此要计算 2 个部件的情况和 3 个部件的情况，计算公式如下，其中 N 为总部件数，i 为当前计算的正常运行的部件数，R 为可靠性。

$$C_N^i \times R_0^i \left(1-R_0\right)^{N-i}$$

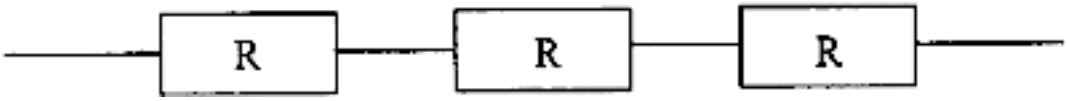
$$C_m^n = \frac{m!}{n!(m-n)!}$$

平均无故障时间（MTBF）

MTBF=1/λ， λ为失效率

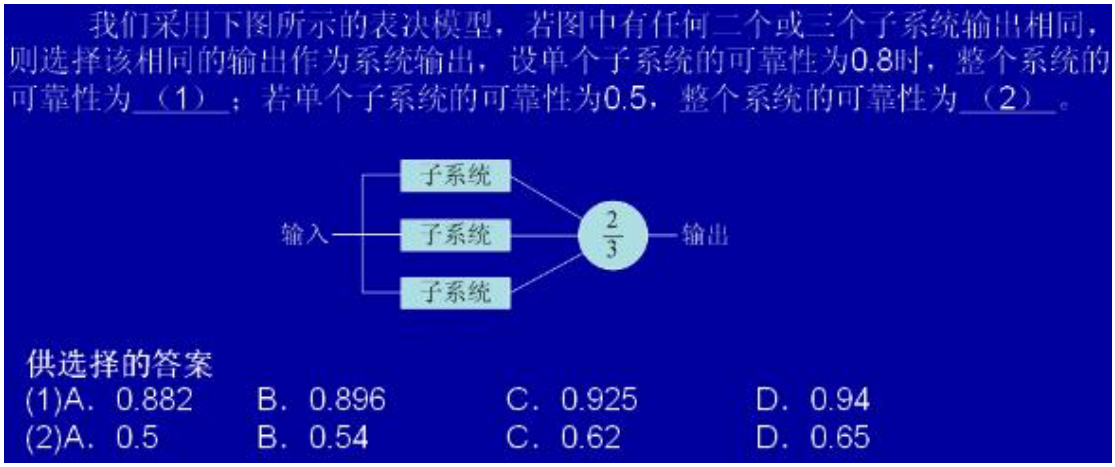
例题

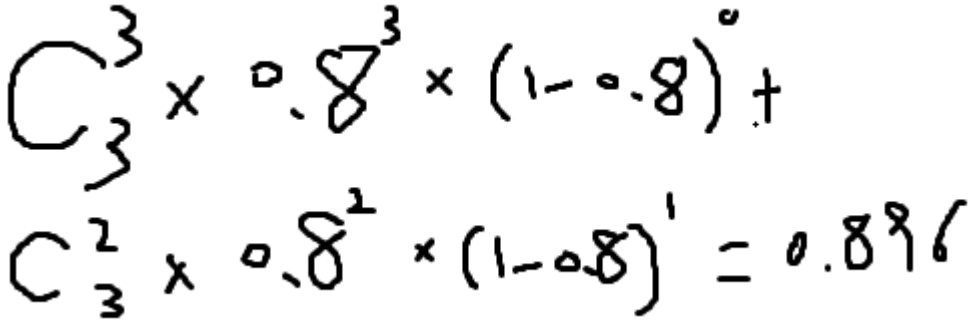
三个可靠度 R 均为 0.8 的部件串联构成一个系统，如下图所示：



则该系统的可靠度为_____

由串联公式可得：0.8×0.8×0.8=0.512





同理，当 R 为 0.5 时可计算出可靠性为 0.5，所以选择 BA

若某计算机系统是由 500 个元器件构成的串联系统，且每个元器件的失效率均为 10-7/H,在不考虑其他因素对可靠性的影响时,该计算机系统的平均故障间隔时间为_____小时。
根据公式可得 MTBF=1/5*10^{-5}=2*10^{4}

缓冲区

单缓冲区

花费的时间=(读缓冲时间+传送用户时间)*磁盘块+每个磁盘处理时间

双缓冲区

花费的时间=读缓冲时间*磁盘块+传送用户时间+每个磁盘处理时间

软件工程

软件生命周期

问题定义

要求系统分析员与用户进行交流，弄清“用户需要计算机解决什么问题”然后提出关于“系统目标与范围的说明”，提交用户审查和确认

可行性研究

一方面在于把待开发的系统的目标以明确的语言描述出来
另一方面从经济、技术、法律等多方面进行可行性分析。

需求分析

确定软件系统的功能需求和非功能需求；
分析软件系统的数据要求：
导出系统的逻辑模型；
修正项目开发计划；
如有必要，可以开发一个原型系统。

开发阶段

1，设计
2，实现：根据选定的程序设计语言完成源程序的编码。
3，测试

维护

1，改正性维护：在软件交付使用后，由于开发测试时的不彻底、不完全、必然会有一部分隐藏的错误被带到运行阶段，这些隐藏的错误在某些特定的使用环境下就会暴露。
2，适应性维护：是为适应环境的变化而修改软件的活动。
3，完善性维护：是根据用户在使用过程中提出的一些建设性意见而进行的维护活动。
4，预防性维护：是为了进一步改善软件系统的可维护性和可靠性，并为以后的改进奠定基础。

统一过程(UP)

简介

每个阶段达到某个里程碑时结束，以下列出了各个阶段的里程碑

初启阶段

生命周期目标

精化阶段

生命周期架构

构建阶段

初始运作功能

移交阶段

产品发布

CMM（Capability Maturity Model，软件能力成熟度模型）

初始级

软件工程管理制度缺乏，过程缺乏定义、混乱无序

可重复级

建立了基本的项目管理过程和实践来追踪项目费用、进度和功能特性

已定义级

所有项目都采用根据实际情况修改后得到的标准软件过程来开发和维护软件

已管理级

收集对软件过程和产品质量的详细度量，对软件过程和产品都有定量的理解与控制

优化级

过程的量化反馈和先进的新思想，新技术促使过程不断改进

CMMI（Capability Maturity Model Integration，软件能力成熟度集成模型）

未完成级

表明过程域的一个或多个特定目标没有被满足

已执行级

过程通过转化可识别的输入工作产品，产生可识别的输出工作产品，关注于过程域的特定目标的完成

已管理级

过程作为已管理的过程制度化，针对单个过程实例的能力

已定义级

过程作为已定义的过程制度化，关注过程的组织级标准化和部署

量化管理级

过程作为定量管理的过程制度化

优化级

过程作为优化的过程制度化，表明过程得到很好地执行且持续得到改进

COCOMO

基本 COCOMO 模型

静态单变量模型，用于对整个软件系统进行估算。

中级 COCOMO 模型

静态多变量模型，将软件系统模型分为系统和部件两个层次

详细 COMO 模型

将软件系统模型分为系统、子系统和模块三个层次，除包括中级模型所考虑的因素外，还考虑了在需求分析、软件设计等每一步的成本驱动属性的影响。

COCOMOII

应用组装模型

在软件工程的前期阶段使用，这时用户界面的原型开发、对软件和系统交互的考虑、性能的评估以及技术成熟度的评价是最重要的

早期设计阶段模型

在需求已经稳定并且基本的软件体系结构已经建立时使用

体系结构阶段模型

在软件的构造过程中使用

Putnam 估算模型

动态多变量，它是假设在软件开发的整个生存周期中工作量有特定的分布。

ISO/IEC 9126（软件质量模型）

简介

由 3 个层次组成：第一层是质量特性，第二层是质量子特性

特性含义

其中各六个质量特性与二十七个质量子特性的关系如下表：

质量特性	功能性	可靠性	易使用性	效率	可维护性	可移植性
质量子特性	适合性	成熟性	易理解性	时间特性	易分析性	适应性
	准确性	容错性	易学性	资源特性	易改变性	易安装性
	互用性	易恢复性	易操作性		稳定性	一致性
	依从性				易测试性	易替换性

UML（Unified Modeling Language，统一建模语言）

构件图

用来以图形化的方式描述**系统的物理结构**，它可以用来显示程序代码如何分解成模块

部署图

描述系统中**硬件**和**软件**的**物理架构**，它描述构成系统架构的**软件构件、处理器和设备**

用例图

以图形化的方式描述**系统与外部系统及用户的交互**。对系统的使用方式进行分类。

协作图

强调收发消息的对象之间的**结构组织**

序列图

描述了在一个用例或操作的执行过程中以**时间顺序**组织的对象之间的交互活动，关注系统的**动态视图**

对象图

展现了**一组对象**以及它们之间的关系，描述了在类图中所建立的事物的实例的**静态快照**

类图

展现了**一组对象、接口、协作**和它们之间的关系，给出系统的**静态视图**，对系统的**静态设计视图建模**（对系统的词汇建模、对简单协作建模、对逻辑数据库模式建模）

状态图

用于**类、接口、协作的行为建模**，强调对象行为的**事件顺序**，关注系统的**动态视图**

活动图

是一种特殊的状态图，展现了在系统内从一个活动到另一个活动的流程。活动图专注于系统的**动态视图**，它对于系统的功能建模特别重要，并强调**对象间的控制流程**。

顺序图

强调的是对象间**发送消息的顺序**。

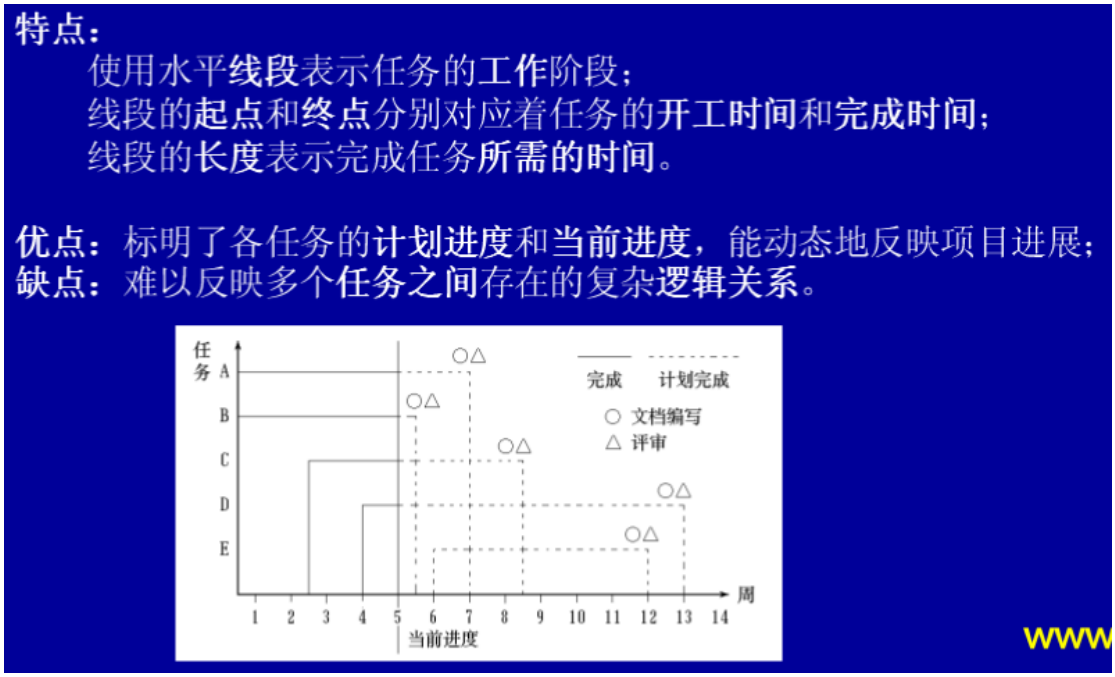
类之间的相互联系

在类与类之间的 5 种关系中，从弱到强依次为：**依赖，关联，聚合，组合和继承**

项目管理

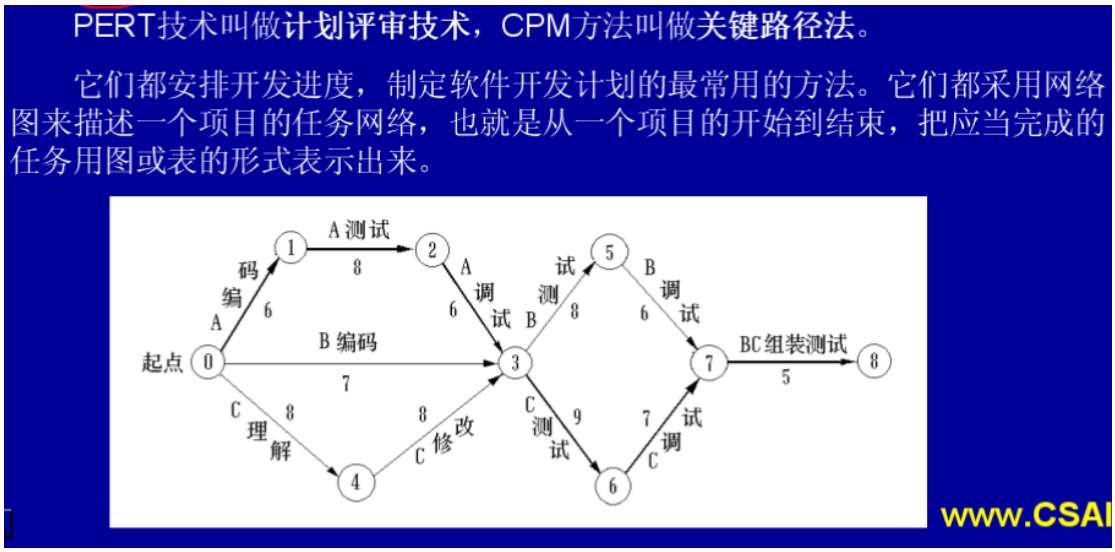
Gantt

无法描述任务之间的依赖关系，也无法确定影响进度的关键任务



PERT 和 CPM

pert 无法描述任务之间的并行关系



耦合

无直接耦合

两个模块之间没有直接的关系，它们分别从属于不同模块的控制与调用，它们之间不传递任何信息。因此，模块间耦合性最弱，模块独立性最高

数据耦合

两个模块之间有调用关系，传递的是简单的**数据值**

标记耦合

两个模块之间传递的是数据结构，如高级语言的数组名，记录名，文件名等这些名字即为标记，其实传递的是这个**数据结构的地址**.

控制耦合

模块间传递的信息不但有数据，还包括**控制信息**。例如：一个模块通过传递开关、标志对某一模块的多种功能进行选择，则这两个模块之间的耦合方式是控制耦合。

内容耦合

当一个模块直接使用另一个模块的内部数据，或通过非正常入口而转入另一个模块内部

外部耦合

模块间通过软件之外的环境联结(如 I/O 将模块耦合到特定的设备、格式、通信协议上)

公共耦合

通过一个公共数据环境相互作用的那些模块间的耦合

内聚

功能内聚

完成一个**单一功能**，各个部分协同工作，**缺一不可**。

顺序内聚

处理元素相关，而且必须**顺序执行**

通信内聚

所有处理元素集中在一个**数据结构的区域**上

过程内聚

处理元素相关，而且必须按**特定的次序**执行

瞬时内聚

所包含的任务必须在**同一时间间隔执行**，如初始化模块

逻辑内聚

完成**逻辑上相关的一组任务**

偶然内聚

完成一组**没有关系或松散关系的任务**

软件开发模型

瀑布模型

给出了**软件生存周期**中制定开发计划、需求分析、软件设计、编码、测试和维护等阶段以及各阶段的固定顺序，上一阶段完成后才能进入到下一阶段，整个过程如同瀑布流水。该模型为软件的开发和维护提供了一种有效的管理模式，但在大量的实践中暴露出其缺点，其中最为突出的是缺乏灵活性，特别是**无法解决软件需求不明确或不准确的问题**。这些问题有可能造成开发出的软件并不是用户真正需要的，并且这一点只有在开发过程完成后才能发现。所以瀑布模型**适用于需求明确，且很少发生较大变化的项目**。

演化模型

允许在获取了一组基本需求后，通过快速分析构造出软件的一个初始可运行版本(称作原型)，然后根据用户在适用原型的过程中提出的意见对原型进行改进，从而获得原型的新版本。这一过程重复进行，直到得到令用户满意的软件。该模型主要用于**对软件需求缺乏准确认识的情况。**

螺旋模型

将瀑布模型和演化模型进行结合，在保持二者优点的同时，**增加了风险分析**，从而弥补了二者的不足。该模型沿着螺线旋转，并通过笛卡尔坐标的四个象限分别表示四个方面的活动：制定计划、风险分析、实施工程和客户评估。螺旋模型为项目管理人员及时调整管理决策提供了方便，进而可降低开发风险。

喷泉模型

以**面向对象**的软件开发方法为基础，以用户需求为动力，以对象来驱动的模型。该模型主要用于描述面向对象的开发过程，体现了面向对象开发过程的迭代和无间隙特性。迭代指模型中的活动通常需要重复多次，相关功能在每次迭代中被加入新的系统。无间隙是指在各开发活动(如分析、设计、编码)之间没有明显边界。

软件开发方法

结构化方法

面向数据流、自顶向下、适合数据处理领域的问题、不适合大规模、复杂的项目、难以适应需求的变化

Jackson 方法

面向数据结构、适合小规模的项目、当输入数据结构与输出数据结构之间没有对应关系时，难以应用此方法

原型化方法

适合需求不清、业务理论不确定、需求经常变化的情况。

冗余附加技术

屏蔽硬件错误的容错技术

冗余附加技术包括：关键程序和数据的冗余及调用；检测、表决、切换、重构和复算的实现。

屏蔽软件错误的容错技术

冗余附加技术包括：冗余备份程序的存储及调用；实现错误检测和错误恢复的程序；实现容错软件所需的固化程序。

风险管理

软件风险

项目风险

项目风险威胁到项目计划，若发生项目风险，就有可能拖延项目的进度和增加项目的成本。项目风险是指预算、进度、人员、资源、利益相关者、需求等方面的潜在问题以及他们对软件项目的影响。项目复杂度、规模及结构不确定性也属于项目风险因素

技术风险

技术风险威胁到要开发软件的质量及交付时间。若发生技术风险，开发工作就可能变得很困难或根本不可能。技术风险是指设计、实现、接口、验证和维护等方面的潜在问题。此外，规格说明的歧义性、技术的不确定性、技术陈旧以及前沿技术也是技术风险因素。

商业风险

市场风险

开发了一个没有人真正需要的优良产品或系统

策略风险

开发的产品不再符合公司的整体商业策略

销售风险

开发了一个销售部们不知道如何去销售的产品

管理风险

由于重点的转移或人员的变动而失去了高级管理层的支持

预算风险

没有得到预算或人员的保证

风险识别

风险识别的方法

建立风险条目检查表

风险识别的类型

产品规模

与要开发或要修改的软件的总体规模相关的风险

商业影响

与管理者或市场所施加的约束相关的风险

客户特性

与客户的素质以及开发者的客户定期沟通的能力相关的风险

过程定义

与软件过程定义的程度以及该过程被开发组织遵守的程度相关的风险

开发环境

与用来开发产品的工具的可得性及质量相关的风险

开发技术

与待开发软件的复杂性及系统所包含技术的“新奇性”相关的风险

人员才干及经验

与软件工程师的总体技术水平及项目经验相关的风险

网络安全

主动攻击和被动攻击

主动攻击

包括篡改数据流或伪造数据流，这种攻击试图改变系统资源或影响系统运行。

被动攻击

对信息的保密性进行攻击，即通过窃听网络上传输的信息并加以分析从而获得有价值的情

报，但它并不修改信息的内容。它的目标是**获得正在传送的信息**，其特点是偷听或监视信息的传递。被动攻击只对信息进行监听，不对其进行修改。被动攻击包括信息内容泄露和业务流分析 2 大类

数据安全与保密

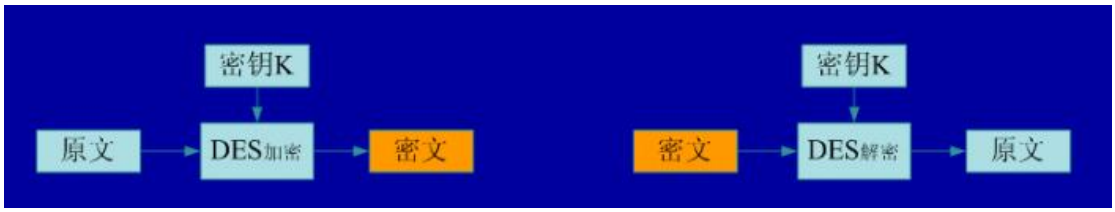
对称加密技术

加密系统的加密密钥和解密密钥相同，或者虽然不同，但从其中的任意一个可以很容易地推导出另一个。

示例

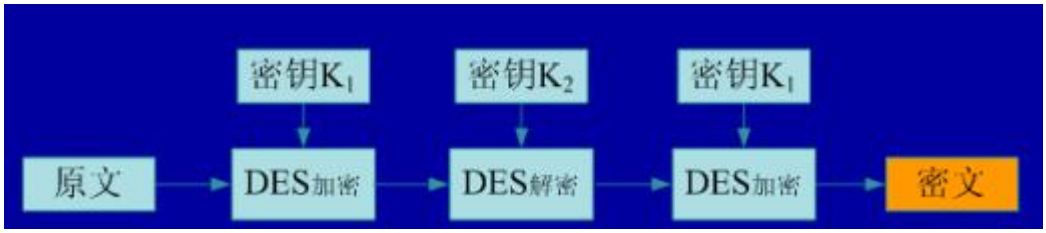
DES（数据加密标准算法）

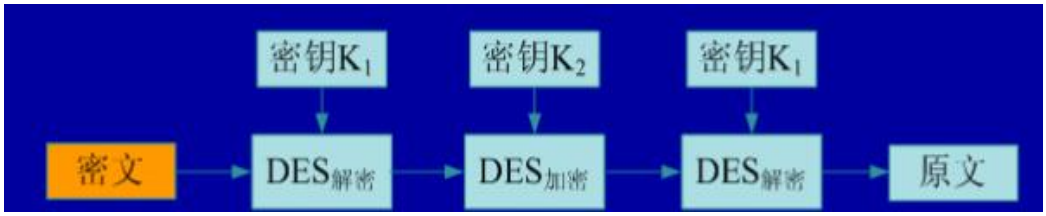
DES 主要采用**替换和移位**的方法加密。它用 **56 位**（64 位密钥只有 56 位有效密钥）密钥对 **64 位**二进制数据块进行加密，每次加密可对 64 位的输入数据进行 16 轮编码，经一系列替换和移位后，输入的 64 位原始数据转换成完全不同的 64 位输出数据。其优点是加密速度快，密钥产生容易。



3DES（TDES）

在 DES 的基础上采用三重 DES，即用两个 56 位的密钥 K1、K2，发送方用 K1 加密，k2 解密，再使用 k1 加密。接受方则使用 k1 解密，k2 加密，再使用 k1 解密，其效果相当于将密钥长度加倍。





IDEA

其明文和密文都是 64 位，密钥长度为 128 位。它比 DES 的加密性好，而且对计算机功能要求也没有那么高。

RC-5

1994 年开发出来的，知道是对称算法即可。

优点

1、 加/解密速度快，适合对大量数据加密
2、 适宜一对一的信息加密传输

缺点

1、 密钥难于传输
2、 密钥的数目难于管理
3、 无法提供信息完整性的鉴别
4、 对称密钥的管理和分发工作是一件具有潜在危险和繁琐的过程

非对称加密技术

公钥和私钥是不一样的，公钥对外开放，私钥仅限自己知道

示例

RSA

知道是非对称加密技术即可

ECC

知道是非对称加密技术即可

优点

1、 安全性好

缺点

1、加/解密速度慢，只适合对少量数据进行加密

消息摘要

消息摘要算法实际上就是一个单向**散列函数**。数据块经过单向散列函数得到一个固定长度的散列值，攻击者不可能通过散列值而编造数据块，使得编造的数据块的散列值和原数据块的散列值相同。

例子

MD5

散列值为 128 位

SHA

散列值为 160 位

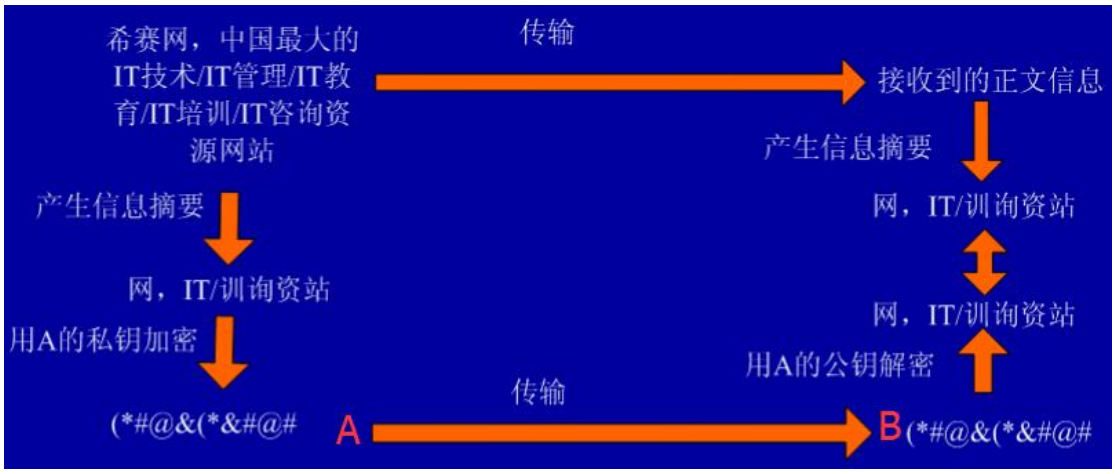
安全性比较

由于 SHA 通常采用的密钥长度较长，因此安全性高于 MD5。

数字签名

实现原理

以下题为例，A 想要传送一段文本给 B，则 A 先利用信息摘要技术产生一段字符串，再利用自身的私钥对字符串进行加密生成密码串。将密码串与明文一同发送给 B。B 接受后，利用 A 的公钥对密码串进行解密，产生字符串，再对明文进行信息摘要产生字符串，将 2 个字符串相比较，如果一致，则证明该明文的确来自 A 且未被修改。

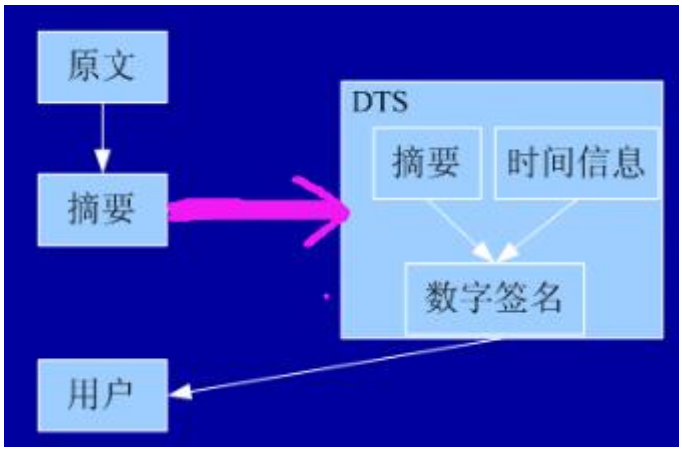


优点

1、 可以证明消息来源是否为本人
2、 保证明文没有被人篡改过

数字时间戳

属于数字签名技术的变种应用。数字时间戳服务（Digtal Time Stamp Service,DTS）是网上电子商务安全服务项目之一，能提供电子文件的日期和时间信息的安全保护。
时间戳是一个经加密后形成的凭证文档，它包括 3 个部分：
1、 需加时间戳的文件的摘要
2、 DTS 收到文件的日期和时间
3、 DTS 的数字签名



病毒

文件型

感染可执行文件，包括 EXE 和 COM 文件

引导型

影响软盘或硬盘的引导扇区

目录型

修改硬盘上存储的所有文件的地址

宏病毒

前缀为 Macro，感染 word 或者 excel 文件

数据通信与网络基础

OSI 七层模型及相关考点

记忆技巧：All people seem to need data processing.

层次	名称	主要功能	主要设备及协议
7	应用层	实现具体的应用功能	POP3、FTP、HTTP、Telnet、SMTP DHCP、TFTP、SNMP、DNS
6	表示层	数据的格式与表达、加密、压缩	
5	会话层	建立、管理和终止会话	
4	传输层	端到端的连接	TCP、UDP
3	网络层	分组传输和路由选择	三层交换机、路由器 ARP、RARP、IP、ICMP、IGMP
2	数据链路层	传送以帧为单位的信息	**网桥**、交换机、网卡 PPTP、L2TP、SLIP、PPP
1	物理层	二进制传输	中继器、**集线器**

分层模型	OSI 模型
进程/应用层	应用层
	表示层
	会话层
主机-主机层	传输层
网络互联层	网络层
网络接口层	数据链路层
	物理层

已知 IP 地址与掩码求网络号与主机号

IP 地址 A.B.C.D 一共 4 个字段，各个字段各占 1 字节，若转化为二进制，则一共 32 位，每个字段占 8 位

A 类地址最高位是 0，网络地址占 1 字节，主机地址占 3 字节

B 类地址最高位是 10，网络地址占 2 字节，主机地址占 2 字节

C 类地址最高位是 110，网络地址占 3 字节，主机地址占 1 字节

D 类地址最高位是 1110，用于组播，无网络地址与主机地址

E 类地址最高位是 1111，实验保留，无网络地址与主机地址

表 5-2　带点十进制符号表示的默认子网掩码

地　址　类	子网掩码位	子 网 掩 码
A 类	11111111 00000000 00000000 00000000	255.0.0.0
B 类	11111111 11111111 00000000 00000000	255.255.0.0
C 类	11111111 11111111 11111111 00000000	255.255.255.0

子网掩码的 1 代表网络号，0 代表主机号

网络号求法：将 IP 地址与子网掩码转化为二进制后取与运算后，根据子网掩码 1 的位数取字节

以下数据为例：

IP 地址：202.197.119.110

掩码 ：255.255.255.0

IP 地址二进制 ：1100 1010.1100 0101.0111 0111.0110 1110

子网掩码二进制：1111 1111.1111 1111.1111 1111.0000 0000

求得网络号 ：1100 1010.1100 0101.0111 0111(因为子网掩码有 24 个 1，所以取 3 个字节)

主机号求法：先将子网掩码按位取反再与 IP 地址进行取与运算，根据子网掩码 0 的位数取字节

IP 地址二进制 ：1100 1010.1100 0101.0111 0111.0110 1110

子网掩码取反 ：0000 0000.0000 0000.0000 0000.1111 1111

主机号 ： 0110 1110(因为子网掩码有 8 个 0，所以取 1 个字节)

判断 2 个 IP 地址是否在一个网段

分别求出 2 个 IP 地址的网络号，如果相同则说明是同一个网段的 IP 地址，否则不在同一个网段

可连接的主机数

2^n-2，n 为主机号位数

例题

一个局域网中某台主机的 IP 地址为 176.68.160.12，使用 22 位作为网络地址，那么该局域网的子网掩码为_____，最多可以连接的主机数为_____

子网掩码的 1 是网络位，0 是主机位，22 位网络地址代表着子网掩码有 22 个 1，所以子网掩码为 11111111.11111111.11111100.00000000，转化为十进制则是 255.255.252.0。既然网络位为 22 位，则主机位为 32-22=10 位，可以连接的主机数为 $2^{10}-2=1022$

端口

默认端口

FTP

从服务器端向客户端发起连接，称之为主动模式，默认数据端口是 20

从客户端向服务器端发起连接，称之为被动模式，默认数据端口是 1025-65535

默认控制端口是 21

HTTP

默认端口号是 80

SMTP

默认端口是 25

POP3

默认端口是 110

NNTP news

新闻组传输协议默认端口是 119

Telnet

默认内部端口与外部端口均为 23

端口范围

公共服务保留端口

0~1023（有时是不算 0 号端口的）

注册端口

1024~49151

动态或私有端口

49152~65535

多媒体

计算公式

数据传输率

采样频率(Hz)×量化位数(bit)×声道数，单位为 b/s

声音信号数据量

数据传输率×持续时间/8

音频容量的计算公式

存储量=采样时间(s) *采样频率(Hz) *量化位数(位) *声道数/8/1024(kb)

图片容量的计算公式

存储量=水平像素*垂直像素*颜色位数/8/1024(kb)
若提示为 X 位或 X 位色，则颜色位数就是 X，若提示为 X 色，那么颜色位数为$\log_2 X$。

视频容量的计算公式

存储量=每帧图像容量*图像帧数
=水平像素*垂直像素*颜色位数*帧频*时间(秒)/8/1024(kb)
PAL 制：25 帧/秒
NTSC 制：30 帧/秒

例题

CD 上声音的采样频率为 44.1kHz，样本精度为 16b/s，双声道立体声，那么其未经压缩的数据传输率为_____

14、A. 88.2kb/s B. 705.6kb/s C. 1411.2kb/s D. 1536.0kb/s
数据传输率=采样频率(Hz)×量化位数(bit)×声道数=44100*16*2=1411200b/s，选 C

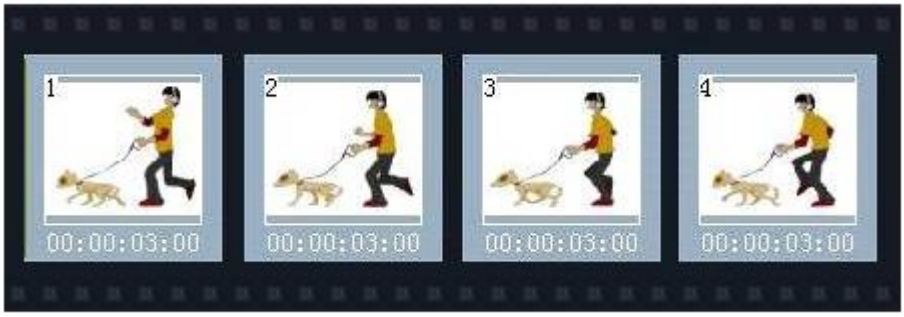
冗余

空间冗余(几何冗余)

多个像素点都是重复的，例如，一幅蔚蓝的天空中漂浮着白云的图像，其蔚蓝的天空及白云本身都具有较强的相关性，这种相关性的图像部分，在数据中就表现为冗余。

时间冗余

对于电视动画类的图像，在其序列的前后相邻的 2 幅图像中，其图像呈现较强的相关性，这就反映为时间冗余。如某一帧图像经过 T 时间后，在某下一帧图像中带有较强的相关性(即画面像素相似)。



知觉冗余

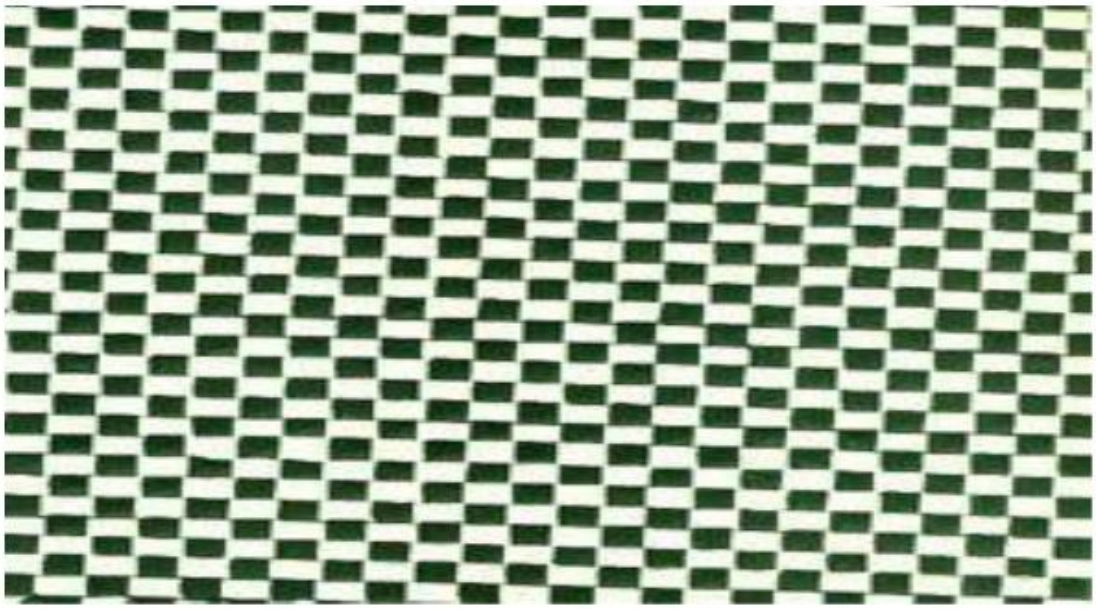
知觉冗余指那些处于人们听觉和视觉分辨力以下的视频音频信号，若在编码时舍去这种在感知界限以下的信号，虽然会使恢复原信号产生一定的失真，但并不能为人们所感知，为此，此种超出人们感知能力部分的编码就称为知觉冗余。例如，一般的视频图像采用 28 的灰度等级，而人们的视觉分辨力仅达 26 的等级，此差额即为知觉冗余。

信息熵冗余

信息熵是指一组数据所携带的信息量，信息的码符号的信息量大于该信息本身的信息量即是信息熵冗余。

结构冗余

有些图像从大的区域上看存在着非常强的**纹理结构**，例如，布纹图像和草席图像在结构上存在冗余。



知识冗余

有许多图像的理解与某些**基础知识**有相当大的相关性。例如，人脸的图像有固定的结构，嘴的上方有鼻子，鼻子的上方有眼睛，鼻子位于正面图像的中上方等。这类规律性的结构可由先验知识和背景知识得到，称此类冗余为知识冗余。

MPEG 标准

MPEG-1

数字电视标准，MP3。

MPEG-2

数字电视标准。

MPEG-4

多媒体应用标准。

MPEG-7

多媒体内容描述接口标准。

MPEG-21

多媒体框架结构标准。

媒体

感觉媒体

感觉媒体指的是能直接作用于人们的感觉器官，从而能使人产生直接感觉的媒体。如文字、数据、声音、图形、图像等。
在多媒体计算机技术中，我们所说的媒体一般指的是感觉媒体。

表示媒体

表示媒体指的是为了传输感觉媒体而人为研究出来的媒体，借助于此种媒体，能有效地存储感觉媒体或将感觉媒体从一个地方传送到另一个地方。如**语言编码、电报码、条形码**等。

表现媒体

表现媒体指的是用于通信中使电信号和感觉媒体之间产生转换用的媒体。如**输入、输出设备，包括键盘、鼠标器、显示器、打印机**等。

存储媒体

存储媒体指的是用于存放表示媒体的媒体。如**纸张、磁带、磁盘、光盘**等。

传输媒体

传输媒体指的用于传输某种媒体的物理媒体。如**双绞线、电缆、光纤**等。

算法设计

迭代法

用于求方程的近似根。
1、若方程无解，则算法求出的近似根序列就不会收敛，迭代过程会变成死循环，因此在使用迭代算法前应先考查方程是否有解，并在程序中对迭代的次数给予限制。
2、方程虽有解，但迭代公式选择不当，或迭代的初始近似根选择不合理，也会导致迭代失败。

穷举搜索法

对可能是解的众多候选解按某种顺序进行逐一枚举和检查，并从中找出符合要求的候选解作为问题的解

递推法

利用问题本身所具有的一种递推关系求问题解的一种方法

递归法

执行过程分递推和回归两个阶段：在递推阶段，把较复杂的问题的求解分解成比原问题简单一些的问题的求解。在回归阶段，从获得的最简单情况的解，逐级返回，依次获得稍复杂问题的解。（递归法就是把问题转化为规模缩小了的同类问题的子问题）

分治法

把大问题分解成一些较小的问题，然后由小问题的解方便地构造出大问题的解，每个小问题都是相互独立的。

示例

二分查找法、汉诺塔问题、斐波那契、归并排序

动态规划法

基本思想也是将大问题分解为多个小问题，但是与分治法不同的是，动态规划法的子问题往往不是独立的。因此，动态规划法可以避免大量重复的计算。以自底向上的方式计算出最优

值。

示例

最大子段问题

贪心法（贪婪法）

不追求最优解，只希望得到较为满意解的方法。可以快速得到满意的解，不考虑整体情况，所以贪心法不要回溯。

示例

哈夫曼编码

回溯法（试探法、通用解题法）

该方法首先暂时放弃关于问题规模大小的限制，并将问题的候选解按某种顺序逐一枚举和检验。当发现当前候选解不可能是解时，就选择下一个候选解；倘若当前候选键除了不满足问题规模要求外，满足所有其他要求，继续扩大当前候选解的规模，并继续试探。如果当前候选解满足包括问题规模在内的所有要求，该候选键就是问题的一个解。在回溯法中，放弃当前候选解，寻找下一个候选解的过程被称为回溯；扩大当前候选解的规模，以继续试探的过程称为向前试探，回溯法以**深度优先**的方式搜索解空间树。

分支限界法

类似于回溯法，也是在问题的解空间树上搜索问题解的方法。但在一般情况下，二者的求解目标不同。回溯法是找出解空间树中满足约束条件的所有解，而分支限界法的求解目标则是找出满足约束条件的一个解，或是在满足约束条件的解中找出使某一目标函数值达到极大或极小的解，即在某种意义下的最优解。分支限界法以**广度优先或以最小耗费优先**的方式搜索解空间树。

示例

单源最短路径问题

算法复杂度

●已知算法 A 的运行时间函数为 $T(n)=8T(n/2)+n^2$，其中 n 表示问题的规模，则该算法的时间复杂度为（ 62）另已知算法 B 的运行时间函数为 $T(n)=XT(n/4)+n^2$，其中 n 表示问题的规模。对充分大的 n，若要算法 B 比算法 A 快，则 X 的最大值为（ 63）。

(62) A. O(n)　　B. O(nlgn)　　C. $O(n^2)$　　D. $O(n^3)$

(63) A．15　　B．17　　C．63　　D．65

一般地，当递归方程为 T(n) = aT(n/c) + O(n), T(n)的解为：

O(n)　　(a<c && c>1)

$O(n^{\log_2 n})$　　(a=c && c>1)

$O(n^{\log_c a})$　　(a>c && c>1)

第一问套用第三个公式即可解决，选 D。

第二问，算法 B 比 A 快，意味着 B 的时间复杂度比 A 小。那么选项中均符合第三个公式的情况，那么 B 的时间复杂度为$O(n^{\log_4 X})$，可求得当 X=64 时，与 A 的算法时间复杂度相等，根据题意选 C

概率算法

数值概率算法

适用于数值问题的求解，这类算法得到的往往是**近似解**，且近似解的精度随时间的增加不断提高。在多数情况下，要计算出问题的精确解是不可能的或是没有必要的，因此数值概率算法可得到相当令人满意的解。

蒙特卡罗算法

适用于求问题的**精确解**，该算法能求得一个解，但该解未必是正确的，正确的概率依赖算法所用的时间，时间约久，正确率越高。因此，该算法的缺点就是无法有效地判断所得到的解是否正确。

拉斯维加斯算法

如果该算法找到一个解，那一定是正确的解，得到正确解的概率依赖算法所用的时间。

舍伍德算法

一定能找到一个正确的解，该算法**设法消除最坏情形与特定实例之间的关联性**。

类之间的关系

名称 概念 实(虚)线 三角形(菱形) 实(空)心

名称	概念	实(虚)线	三角形(菱形)	实(空)心
依赖	有 2 个元素 X，Y，若修改 X 的定义可能会引起对另一个元素 Y 的定义的修改，则称 Y 依赖于 X	虚线	三角形	实心
泛化	父类是子类的泛化	实线	三角形	空心
关联(聚合)	表示两个对象之间是整体和部分的弱关系，部分的生命周期可以超越整体。如电脑和鼠标。	实线	菱形	空心
关联(组合)	表示两个对象之间是整体和部分的强关系，部分的生命周期不能超越整体，或者说不能脱离整体而存在。	实线	菱形	实心
实现	接口和实现借口的类	虚线	三角形	空心

面向对象设计模式

创建型模式

简单工厂模式 (Simple Factory Pattern)
工厂方法模式 (Factory Method Pattern)
抽象工厂模式 (Abstract Factory Pattern)
单例模式 (Singleton Pattern)
建造者模式 (Builder Pattern)
原型模式 (Prototype Pattern)

结构型模式

桥接模式 (Bridge Pattern)
组合模式 (Composite Pattern)
装饰者模式 (Decorator Pattern)
外观模式 (Facade Pattern)
代理模式 (Proxy Pattern)
享元模式 (Flyweight Pattern)
适配器模式 (Adapter Pattern)

行为型模式

模板方法模式 (Template Method Pattern)
命令模式 (Command Pattern)
迭代器模式 (Iterator Pattern)
观察者模式 (Observer Pattern)
解释器模式 (Interpreter Pattern)
中介者模式 (Mediator Pattern)
职责链模式 (Chain of Responsibility Pattern)
备忘录模式 (Memento Pattern)
策略模式 (Strategy Pattern)
访问者模式 (Visitor Pattern)
状态模式 (State Pattern)

泛化(概化)

表示把几类对象类的公共属性和行为抽象成超类，然后其属性和方法被那些子类继承

聚合

表示一个较大的“整体”类包含一个或多个较小的“部分”类

合成

表示关系中“整体”负责其“部分”的创建和销毁，如果“整体”不存在了，“部分”也将不存在。

单例

保证一个类仅能够生成一个对象

组合

表示“部分-整体”的层次结构，并且对部分和整体的使用具有一致性

装饰

动态地给一个对象增加一些额外的职责，无须改变类的设计和实现