合肥工業大學
HEFEI UNIVERSITY OF TECHNOLOGY

数据结构课程设计习题集

（2017 版）

单　　位：合肥工业大学

授课时间：2017 - 2018 学年 第一学期

命 题 人：周 波

授课年级：2016 级本科生

总 课 时：

课程设计评分思想：

（1）需求分析清晰，能清楚地领会题目要求；

（2）思路清晰，数据结构设计合理，算法编写步骤清晰；

（3）对于有交互需求的题目，需要设计界面，要求界面友好，可操作性强；

（4）鼓励采用多种算法解决问题，并要能清晰地阐述多种算法的思想，采用多种算法解决问题的需要采用科学合理的方法对算法进行测试、比较，并对测试的结果进行总结、分析、说明；

（5）文档要按照指定格式完成，要求文档格式规范，文字简洁易懂，对于有图表的，需要对图表进行标号，图表的绘制要符合图表制定的要求，对于文档没有按照教师意见进行修改的，课程设计得分不能超过 60 分；

（6）答辩要脱稿完成，对于不能够完整、清晰地陈述设计思路、思想的，其打分不能超过 60 分，对于优秀的课程设计（>=90 分），需要采取公开答辩或找两位课程设计老师答辩的方式完成；

（7）文档、代码打印部分需在指定时间内上交，对于在指定时间内没有上交的同学，其课程设计得分不得超过 60 分；

（8）课程设计题目分数为完成了题目需求和文档的分数，原则上最高得分不能超过该分数，但对于有同学完成了题目要求之外的功能的（创新性功能的），可给予加分，但加分不能超过 10 分。

周　波

2017 年

基本习题部分

1 字符串距离

目的：字符串是一种基础且广泛使用的数据结构，与字符串相关的题目既可以考察基本程序设计能力和技巧，也可以考查算法设计能力。

题目：求字符串之间距离

要求：设有字符串 X，称在 X 的头尾及中间插入任意多个空格后构成的新字符串为 X 的扩展串，如字符串 X 为“abcbcd”，则字符串“abcb□cd”，“□a□bcbcd□”和“abcb□cd□”都是 X 的扩展串，这里“□”代表空格字符。如果 A1 是字符串 A 的扩展串，B1 是字符串 B 的扩展串，A1 与 B1 具有相同的长度，那么定义字符串 A1 与 B1 的距离为相应位置上的字符的距离总和，而两个非空格字符的距离定义为它们的 ASCII 码的差的绝对值，而空格字符与其它任意字符之间的距离为已知的定值 K，空格字符与空格字符的距离为 0。在字符串 A、B 的所有扩展串中，必定存在两个等长的扩展串 A1、B1，使得 A1 与 B1 之间的距离达到最小，将这一距离定义为字符串 A、B 的距离。请编写程序，求出字符串 A、B 的距离。

2 求最长公共子串

求解 2 个字符串的最长公共子串。输入的 2 个字符串可以从键盘读入，也可以从两个文本文件中读入。

实现提示：对于采用动态规划法和后缀树算法的，要对算法设计思想能够进行详细阐述，并分析算法的时间复杂和空间复杂度。能够详细阐述多种算法并设计完成的，总分可达到 80 分。

3 集合运算

问题描述：

设有两个用单链表表示的集合 A、B，其元素类型是 int 且以递增方式存储，其头结点分别为 a、b。要求下面各问题中的结果集合同样以递增方式存储，结果集合不影响原集合。

实现要求：

（1）编写集合元素测试函数 IN_SET，如果元素已经在集合中返回 0，否则返回 1；

（2）编写集合元素输入并插入到单链表中的函数 INSERT_SET，保证所输入的集合中的元素是唯一且以递增方式存储在单链表中；

（3）编写集合元素输出函数，对建立的集合链表按递减方式输出；

（4）编写求集合 A、B 的交 C=A∩B 的函数，并输出集合 C 的元素；

（5）编写求集合 A、B 的并 D=A∪B 的函数，并输出集合 D 的元素；

（6）求集合 A 与 B 的对称差 E=（A-B）∪（B-A） 的函数，并输出集合 D 的元素；

（7）设计一个菜单，具有输入集合元素、求集合 A、B 的交 C、求集合 A、B 的并 D、求集合 A 与 B 的对称差 E、退出等基本的功能。

测试数据：由读者自定，但集合 A、B 的元素个数不得少于 16 个。

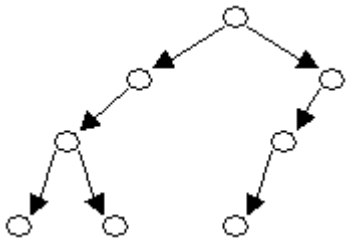
4 二叉树结点染色问题

目的：二叉树是常用的重要非线性数据结构，在客观世界中有着广泛的应用。通过本题可以加深对于二叉树这一数据结构的理解。掌握二叉树的存储结构及各种操作。

要求：一棵二叉树可以按照如下规则表示成一个由 0、1、2 组成的字符序列，我们称之为“二叉树序列 S”：

$$S=\begin{cases}0 & \text{表示该树没有子节点}\\ 1S_1 & \text{表示该树有一个子节点，}S_1\text{为其子树的二叉树序列}\\ 2S_1S_2 & \text{表示该树有两个子节点，}S_1\text{和}S_2\text{分别表示其两个子树的二叉树序列}\end{cases}$$

例如，下图所表示的二叉树可以用二叉树序列 S=21200110 来表示。



任务是要对一棵二叉树的节点进行染色。每个节点可以被染成红色、绿色或蓝色。并且，一个节点与其子节点的颜色必须不同，如果该节点有两个子节点，那么这两个子节点的颜色也必须不相同。给定一棵二叉树的二叉树序列，请求出这棵树中最多和最少有多少个点能够被染成绿色。

5 散列表的设计与实现

任务：设计散列表实现电话号码查找系统。

要求：

①设每个记录有下列数据项：用户名、电话号码、地址；

②从键盘输入各记录，以用户名（汉语拼音形式）为关键字建立散列表；

③采用一定的方法解决冲突；

④查找并显示给定电话号码的记录；

可以选作内容：

①系统功能的完善；

②设计不同的散列函数，比较冲突率；

③在散列函数确定的前提下，尝试各种不同类型处理冲突的方法，考察平均查找长度的变化。

6 求有向图简单路径

简单路径：如果一条路径上的顶点除了起点和终点可以相同外，其它顶点均不相同，则称此路径为一条简单路径；起点和终点相同的简单路径称为回路（或环）。

要求：

①求出有向图中从起点到终点的所有简单路径。其中起点和终点可以由用户自行设定。

②求出有向图中从起点到终点的指定长度（如 K）的所有简单路径。其中起点和终点可以由用户自行设定。

7 关键路径问题

问题描述：设计一个程序求出完成整项工程至少需要多少时间以及整项工程中的关键活动。

基本要求：（1）对一个描述工程的 AOE 网，应判断其是否能够顺利进行。

（2）若该工程能顺利进行，输出完成整项工程至少需要多少时间，以及每一个关键活动所依附的两个顶点、最早发生时间、最迟发生时间。

8 布尔表达式真值问题

目的：本课程设计是求中缀算术表达式真值问题。求中缀算术表达式值的问题是数据结构中栈的一个典型应用。通过本题，学生应掌握中缀表达式和后缀表达式的转换方法和后缀表达式求值问题。

要求：已知某种类型的布尔表达式由“T”、“F”、“！”、“&”和“|”组成，其中，“T”代表真值 True，“F”代表真值 False，“！”代表逻辑非运算，“&”代表逻辑与运算，“|” 代表逻辑或运算。并且，运算符“！”、“&”和“|”的优先级为：“！”最高，“|”最低，“&”介于“！”和“|”之间。你的任务是，计算给定布尔表达式的真值。

例如，布尔表达式“（T|T）&F&（F|T）”的真值为“F”.

9 字符串模式匹配算法比较

用三种以上方法实现字符串模式匹配，比如 BF、KMP 和 BM 方法等，并利用科学合理的方法针对这几种方法进行性能比较。

10 排序算法及性能对比

编程实现快速排序、堆排序、希尔排序、冒泡排序、归并排序算法，学生可增加其它排序算法，并完成不同排序算法的性能对比分析。

要求：

①时间性能包括平均时间性能、最好情况下的时间性能、最差情况下的时间性能等。

②实验数据应具有说服力，包括：

规模范围要大（如从 100 到 10000）

数据的初始特性类型要多，因而需要具有随机性；

实验数据的组数要多，即同一规模的数组要多选几种不同类型的数据来实验。

③算法所用时间必须是机器时间，也可以包括比较和交换元素的次数。

④实验分析及其结果要能以清晰的方式来描述，如数学公式或图表等。

⑤要给出实验的方案及其分析。

11 广义表实现

选择合适的存储结构表示广义表，并能实现下列运算要求：

①用大写字母表示广义表，用小写字母表示原子，并提供设置广义表的值的

功能。

②取广义表 L 的表头和表尾的函数 head（L）和 tail（L）。

③能用这两个函数的复合形式求出广义表中的指定元素。

④由广义表的字符串形式到广义表的转换函数 Lists Str_ToLists_（S）；

例如

Str_ToLists_（“ （a,（a,b）,c）”）的值为一个广义表。

⑤由广义表到广义表的字符串形式的转换函数 char * Lists_To_Str（L）。

⑥最好能设置多个广义表。

算法编写

12 后缀表达式计算器

目的：后缀表达式不包含括号，运算符放在两个运算对象的后面，所有的计算按运算符出现的顺序，严格从左向右进行（不再考虑运算符的优先规则，如：

2 1 + 3 *,即（2 + 1） * 3

通过本课程设计，应使学生掌握后缀表达式的特点、栈的基本方法和基本原理，培养学生运用语言编程及调试的能力，运用数据结构解决简单的实际问题的能力，为后续计算机专业课程的学习打下坚实的基础。

要求：实现一个简单的后缀表达式计算器。假定表达式里的基本数值为实数，可用的运算符包括+,-,*,/,^，其中的 ^ 表示求幂运算。

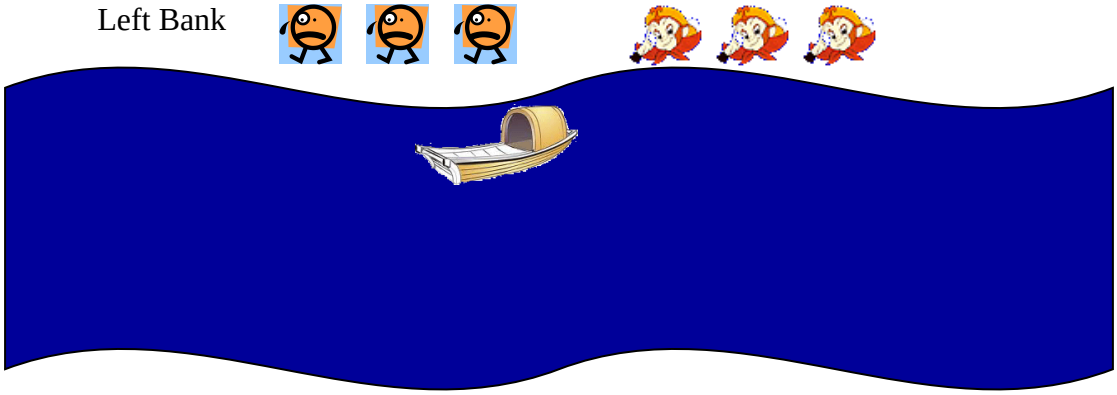
①假定输入表达式里的数和运算符之间都有空格，这样可以简化输入的处理；

②输入的算术表达式以分号为结束符。计算器应该能输入并计算一系列表达式，遇到一行的第一个字符就是分号时程序结束。

③上题的计算器增加一元函数功能，允许表达式里写 sin, cos, tan, log（自然对数）等函数，还可以考虑加入自定义的其他数学函数。（选做）

13 野人修道士问题

设有 3 个传教士和 3 个野人来到一条河的左岸，打算乘一只船从左岸渡到右岸去。该船的负载能力为 2 人。在任何时候，如果野人人数超过传教士人数，那么野人就会把传教士吃掉。野人绝对服从传教士的指挥和调度。



提示：

基于河的一岸求解问题，比如左岸，不需考虑船在河中央的状态。用三个变量人数、野人数、船是否在左岸三个变量描述问题的状态，比如（M,W,B）。用函数来使问题的状态发生变化，最后到达目标状态。

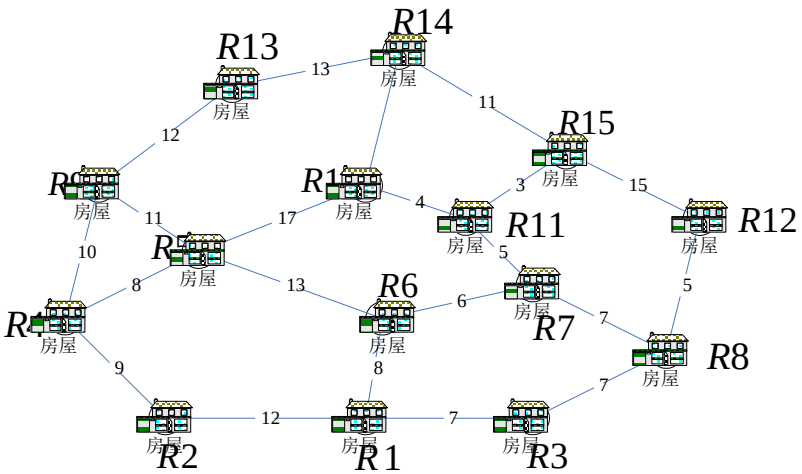
初始状态：（3,3,1），目标状态：（0,0,0）

要求：

编程求解问题；给出中间状态；给出解序列（函数调用序列）

14 中国邮路问题

邮递员的工作是每天在邮局里选出邮件，然后送到他所管辖的客户中，再返回邮局。自然地，若他要完成当天的投递任务，则他必须要走过他所投递邮件的每一条街道至少一次。问怎样的走法使他的投递总行程为最短？这个问题就称为中国邮路问题。



编程要求：

（1）求解用户输入的图形的中国邮路问题

要求用户输入图形，求解输入的图形的中国邮路问题，要求能显示图形和最终结果。

（2）加入图形编辑器

系统自动生成图形，系统求解生成的图形的中国邮路问题，要求能显示图形和最终结果。

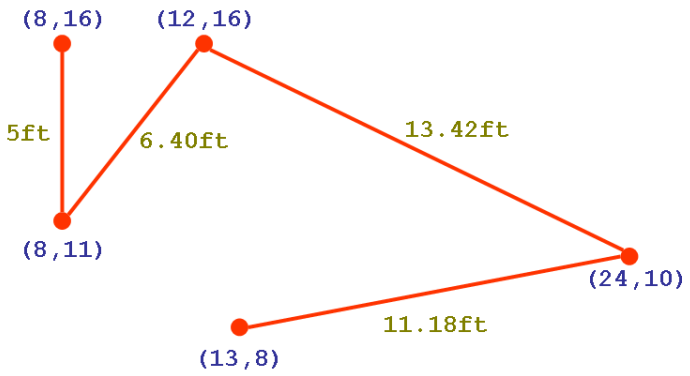
（3）附加要求

能够图形显示求解过程。

提示：此题请查阅相关资料后再做题。

15 网络布线

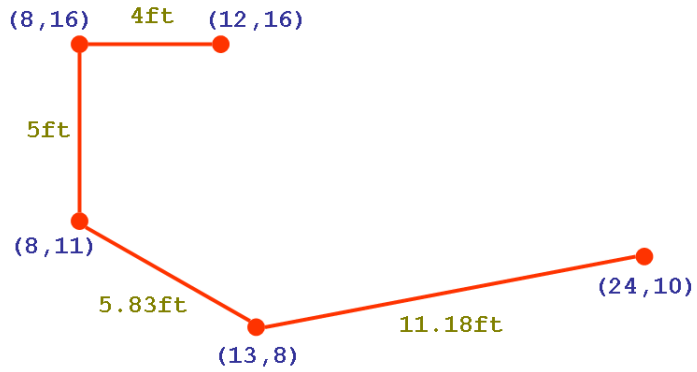
计算机网络要求网络中的计算机被连接起来，本问题考虑一个“线性”的网络，在这一网络中计算机被连接到一起，并且除了首尾的两台计算机只分别连接着一台计算机外，其它任意一台计算机恰连接着两台计算机。图 1 中用圆点表示计算机，它们的位置用直角坐标表示。网络连接的计算机之间的距离单位为英尺。



由于很多原因，我们希望使用的电缆长度应可能地短。你的问题是去决定计算机应如何被连接以使你所使用的电缆长度最短。在设计方案施工时，电缆将埋在地下，因此连接两台计算机所要用的电缆总长度等于计算机之间的距离加上额外的 16 英尺电缆，以从地下连接到计算机，并为施工留一些余量。

下图是计算机的最优连接方案，这样一个方案用电缆的总长度是：

（4 + 16） + （5 + 16） + （5.38 + 16） + （11.18 + 16） = 90.01 英尺



基本要求：

输入网络中的计算机总数和每台计算机的坐标。

输出使电缆长度最短的连接方案。给出最优连接方案中每两台相邻计算机之间的距离，以及总的电缆长度。

提高要求：

参考图 2，用图形化的方式显示结果，包括点的坐标、最优路径、相邻计算机之间的距离。

16 最大匹配问题

（1）问题描述：

写出求一个二分图的最大匹配的算法，并用于解决下面的问题。

第二次世界大战时期，英国皇家空军从沦陷国征募了大量外籍飞行员。由皇家空军派出的每一架飞机都需要配备在航行技能和语言上能互相配合的 2 名飞行员，其中 1 名是英国飞行员，另 1 名是外籍飞行员。在众多的飞行员中，每一名外籍飞行员都可以与其他若干名英国飞行员很好地配合。如何选择配对飞行的飞行员才能使一次派出最多的飞机。对于给定的外籍飞行员与英国飞行员的配合情况，试设计一个算法找出最佳飞行员配对方案，使皇家空军一次能派出最多的飞机。

（2）编程任务：

对于给定的外籍飞行员与英国飞行员的配合情况，编程找出一个最佳飞行员配对方案，使皇家空军一次能派出最多的飞机。

数据输入：

由文件 input.txt 提供输入数据。文件第 1 行有 2 个正整数 m 和 n。n 是皇家空军的飞行员总数（n<100）；m 是外籍飞行员数。外籍飞行员编号为 1~m；英国飞行员编号为 m+1~n。接下来每行有 2 个正整数 i 和 j，表示外籍飞行员 i 可以和英国飞行员 j 配合。文件最后以 2 个-1 结束。

结果输出：

程序运行结束时，将最佳飞行员配对方案输出到文件 output.txt 中。第 1 行是最佳飞行员配对方案一次能派出的最多的飞机数 M。接下来 M 行是最佳飞行员配对方案。每行有 2 个正整数 i 和 j，表示在最佳飞行员配对方案中，飞行员 i 和飞行员 j 配对。

如果所求的最佳飞行员配对方案不存在，则输出‘No Solution!’。

（3）输入输出示例

输入文件示例：

input.txt

5 10

1 7

1 8

2 6

2 9

2 10

3 7

3 8

4 7

4 8

5 10

-1 -1

输出文件示例：

output.txt

4

1 7

2　9

3　8

5　10

17 谣言传播问题

目的：通过本课程设计，应使学生掌握如何用图结构解决实际问题的能力，加深对于图结构的理解和认识。掌握图的存储方法和关于图的经典算法。提高学生的程序设计能力。

要求：股票经纪人往往对谣言很敏感，你的老板希望你能找到一个好方法向他们散布谣言，从而使他在股市占有战术优势。为了达到最佳效果，需要谣言传播的尽量快。不幸的是，股票经纪人只相信从他们信任的信源传播来的消息，因此，在你散布谣言之前，需要对他们的联系网进行详细考察。对于一个股票经纪人，他需要一定时间才能将信息传送给他的联系人，给你这些信息，你的任务是，决定选谁作为第一个传送谣言的人，以使谣言传遍所有人的时间最短，当然，如果谣言不能传遍所有人的话，你也要给出说明。

例如，假设共有 3 个联系人，联系人 1 传递信息给联系人 2 和 3 所有的时间分别为 4 和 5；联系人 2 传送信息给联系人 1 和 3 所有的时间分别为 2 和 6；联系人 3 传送信息给联系人 1 和 2 所有的时间均为 2，则选择联系人 3 作为第一个传送谣言的人，可以使谣言传遍所有的人时间最短，为 2.

（选择有向图中的一个源点，使它到其余各顶点的最短路径中最长的一条路径最短）

18 用单链表存储图的顶点表实现图的相关算法

在图的邻接表存储结构中，顶点表用单链表存储。

【基本要求】

设计顶点表的结点结构。

设计图的创建算法。

设计图的销毁算法。

设计 DFS 算法。

设计 BFS 算法。

设计 Prim 算法。

设计 Kruskal 算法

【提高要求】

设计算法。

设计算法。

19 城市距离问题

问题描述：用无序表实现一个城市数据库。每条数据库记录包括城市名（任意长的字符串）和城市的坐标（用整数 x 和 y 表示）。实现数据的插入、删除、查询功能，并实现指定距离内的所有城市。设计算法实现指定一定数目的具体城市，寻找遍历这些城市并回到出发点的最佳路径，观察随着城市数目的增加，算法执行效率的变化。

编程任务：

①用列表对城市进行记录和管理，实现城市的增加、删除和查询功能，并实现文件保存和读取

②计算城市之间距离，统计输出距离某城市一定范围内的所有城市。

③实现一定规模城市的遍历最佳路径选择。

④分析随着城市数目增加时，算法执行效果的改变，深刻理解旅行商问题。

20 公交线路上优化路径的查询

（1）问题描述

最短路径问题是图论中的一个经典问题，其中的 Dijkstra 算法一直被认为是图论中的好算法，但有的时候需要适当的调整 Dijkstra 算法才能完成多种不同的优化路径的查询。

对于某城市的公交线路，乘坐公交的顾客希望在这样的线路上实现各种优化路径的查询。设该城市的公交线路的输入格式为：

线路编号：起始站名（该站坐标）；经过的站点 1 名（该站坐标）；经过的站点 2 名（该站坐标）；……；经过的站点 n 名（该站坐标）；终点站名（该站坐标）。该线路的乘坐价钱。该线路平均经过多少时间来一辆。车速。

例如：63：A（32,45）；B（76,45）；C（76,90）；……；N（100,100）。1 元。5 分钟。1/每分钟。

假定线路的乘坐价钱与乘坐站数无关，假定不考虑公交线路在路上的交通堵

塞。

对这样的公交线路，需要在其上进行的优化路径查询包括：任何两个站点之间最便宜的路径；任何两个站点之间最省时间的路径等等。

（2）课程设计目的

从实际问题中合理定义图模型，掌握 Dijkstra 算法并能用该算法解决一些实际问题。

（3） 基本要求

①根据上述公交线路的输入格式，定义并建立合适的图模型。

②针对上述公交线路，能查询获得任何两个站点之间最便宜的路径，即输入站名 S，T 后，可以输出从 S 到 T 的最便宜的路径，输出格式为：线路 x：站名 S，…，站名 M1；换乘线路 x：站名 M1，…，站名 M2；…；换乘线路 x：站名 MK，…，站名 T。共花费 x 元。

③针对上述公交线路，能查询获得任何两个站点之间最省时间的路径（不考虑在中间站等下一辆线路的等待时间），即输入站名 S，T 后，可以输出从 S 到 T 的考虑在中间站等下一辆线路的等待时间的最省时间的路径，输出格式为：线路 x：站名 S，…，站名 M1；换乘线路 x：站名 M1，…，站名 M2；…；换乘线路 x：站名 MK，…，站名 T。共花费 x 时间。

④针对上述公交线路，能查询获得任何两个站点之间最省时间的路径（要考虑在中间站等下一辆线路的等待时间），即输入站名 S，T 后，可以输出从 S 到 T 的考虑在中间站等下一辆线路的等待时间的最省时间的路径，输出格式为：线路 x：站名 S，…，站名 M1；换乘线路 x：站名 M1，…，站名 M2；…；换乘线路 x：站名 MK，…，站名 T。共花费 x 时间。

（4）实现提示

需深入考虑，应根据不同的应用目标，即不同的优化查询来建立合适的图模型。

21 求第K短的最短路径

（1）问题描述

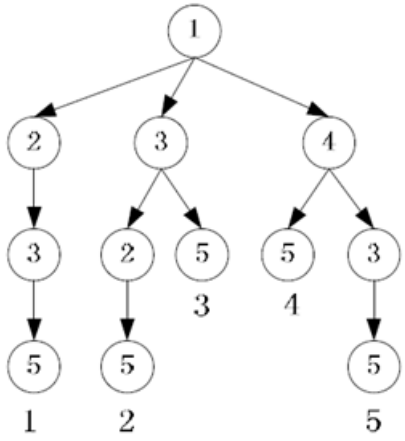
最短路径问题是图论中的一个经典问题，主要研究成果有 Dijkstra、Floyd 等优秀算法，Dijkstra 算法一直被认为是图论中的好算法。但这两个算法有一个

共同的缺陷：这里的最短路径指两点之间最短的那一条路径，不包括次短、再次短等等路径。实际上，在使用咨询系统或决策支持系统时，希望得到最优的决策参考外，还希望得到次优、再次优等决策参考。这同样反映在最短路径问题上，如一个交通咨询系统，除了希望得到最短路径以外，由于交通堵塞等问题，可能需要获知次短、第 K 短的最短路径。因此，有必要将最短路径问题予以扩充，能求出第 K 短最短路径。形式的表述就是想要在图中求出从起点到终点的前 k 短的路径（最短、第 2 短、第 3 短……第 k 短），并且需要这些路径都是无环的。

常见的较好的求解前 k 短无环路径的算法是 Yen 算法（以发明者名字命名的）。现在简要地描述一下 Yen 算法。设 Pi 为从起点 s 到终点 t 的第 i 短的无环路径。一开始是 P_1，也就是从 s 到 t 的最短路径，可以通过 Dijkstra、Bellman-Ford 或 BFS 等算法轻易地求出。接下来要依次求出 P_2，P_3……P_k。可以将 P_1~P_i 看成一棵树，称为 T_i，它的根节点是 s，所有叶节点都是 t。例如假设 s=1，t=5，求出的 P1~P5 为，P1：1->2->3->5；P2：1->3->2->5；P3：1->3->5；

P4：1->4->5；P5：1->4->3->5。此时的 T5 就是如图所示的一棵树。

定义 devi 为 P_i 的偏离点（deviation node），定义为在 Ti 上，Pi 对应的那一分枝中，第 1 个（按从 s 到 t 的顺序）不在 Ti - 1 上的点（i > 1）。为描述的方便，设 dev1 为 P_1 的第 2 个点。因此，在图 1 中，dev1~dev5 的编号分别为 2、3、5、4 和 3。显而易见，devi 至少是 Pi 的第 2 个点。接下来是 Yen 算法的核心部分，即每当求出一个 P_i 时，都可以从 Pi 上发展出若干条候选路径。发展的方法是这样的，对于 P_i 上从 devi 的前一个点到 t 的前一个点这一段上的每个点 v，都可以发展出一条候选路径。用 $P_{i\ sv}$ 表示 P_i 上从 s 到 v 的子路径，用 $P_{i\ vt}$ 表示从 v 到 t 的满足下列条件的最短路径：condition 1：设点 v 在 Ti 上对应的点为 v'，则 $P_{i\ vt}$ 上从点 v 出发的那条边不能与 T_i 上从点 v'出发的任何一条边相同。condition2：$P_{i\ vt}$ 上，除了点 v，其它点都不能出现在 $P_{i\ sv}$ 上。如果找得出 $P_{i\ vt}$，则把 $P_{i\ sv}$ 和 $P_{i\ vt}$ 连起来就组成了一条候选路径。其中 condition 1 保证了候选路径不与 P1~Pi 重复；condition 2 保证了候选路径无环。



路径P_1~P_5对应的树T_5

以图中的例子为基础，可以举一个发展候选路径的例子。在求出了 P5 之后，要在 P5 上发展候选路径。P5 的偏离点是 3 号点。因此 v 的范围是{4, 3}。

当 v = 4 时，Pi sv = 1 -> 4，因此，根据 condition 2，在 Pi vt 上不能出现 1 号点。找到 P5 上的 4 号点在 T5 上对应的那一点，也就是图 6-中位于阴影 3 号点上面的 4 号点，在 T5 上从它出发的有（4, 5）和（4, 3）这两条边，因此，根据 condition 1，在 Pi vt 上不能出现这两条边。假设在这样的情况下，求出了从 4 号点到 t 的最短路径为 4 -> 2 -> 5，它就是 Pi vt。此时发展出的候选路径就是 1 -> 4 -> 2 -> 5。

当 v = 3 时，Pi sv = 1 -> 4 -> 3，因此，根据 condition 2，在 Pi vt 上不能出现 1 号点和 4 号点。找到 P5 上的 3 号点在 T5 上对应的那一点，也就是图 6-66 中阴影的 3 号点，在 T5 上从它出发的只有(3, 5)这一条边，因此，根据 condition 1，在 Pi vt 上不能出现边（3, 5)。假设在这样的情况下，我们求出了从 3 号点到 t 的最短路径为 3 -> 2 -> 5，它就是 Pi vt。此时发展出的候选路径就是 1 -> 4 -> 3 -> 2 -> 5。

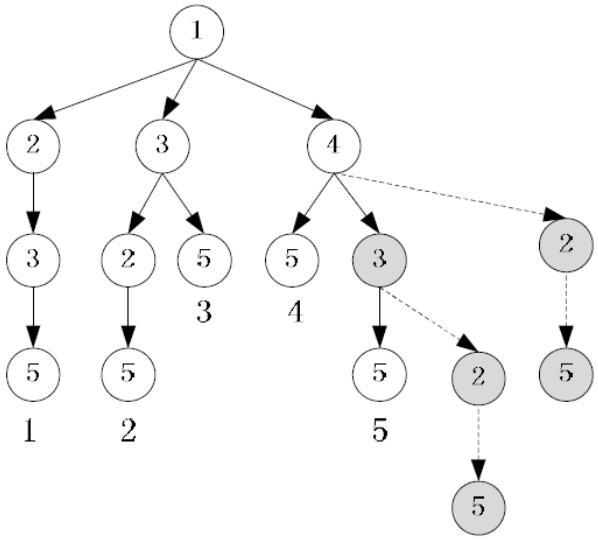


图6-66 扩展路径P_5产生的候选路径

显而易见，在从 Pi 发展出的所有候选路径中，只有当 v 是 devi 的前一个点时，条件 1 才有可能阻挡掉 2 条或 2 条以上边。当 v 不是 devi 的前一个点时，条件 1 只会阻挡掉 1 条边，那就是本身位于 Pi 上，从 v 出发的那条边。不仅从 Pi，从之前的 P1~Pi - 1，我们都发展过若干条候选路径。从候选路径的集合中取出最短的一条，就是 Pi + 1。把 Pi + 1 从候选路径的集合里删掉；然后再从它发展新的候选路径，添加到候选路径的集合里，如此循环，直到求出 Pk 为止。如果在求到 Pk 之前，候选路径的集合就空了，那么说明 P_k 不存在。

（2）课程设计目的

学习、掌握、编程实现 Yen 算法，知道如何求解第 K 短最短路径。

（3）基本要求

①给定一个加权图，编程实现 Yen 算法，依次输出所有的无环最短路径。

②候选路径集合用堆存储实现，方便快速选取最短的一条。

③分析 Yen 算法的时间复杂性。

（4）实现提示

在 Yen 算法中会调用 Dijkstra 算法，图可用邻接矩阵表示。

22 长整数的代数运算

（1）问题描述

设计数据结构完成长整数的表示和存储，并编写算法来实现两长整数的加、减、乘、除等基本代数运算。

（2）课程设计目的

能够应用线性数据结构解决实际问题。

（3）基本要求

①长整数长度在一百位以上。

②实现两长整数在取余操作下的加、减、乘、除操作，即实现算法来求解 a+b modn, a-b mod n, a×b mod n, a÷b mod n。

③输入输出均在文件中。

④分析算法的时空复杂性。

（4）实现提示

需将长整数的加法转化为多个一般整数加法的组合。

23 基于细胞自动机（Cellular Automata）实现tribute模型的模拟与分析

（1）问题描述

细胞自动机是一个时间和空间都离散的动态系统，每个细胞自动机都是由细胞单元（cell）组成的规则网格，每个细胞元都有 k 个可能的状态，其当前状态由其本身及周围细胞元的前一状态共同决定。其具体细节在上题中已有所描述，此处略去。

（2）课程设计目的

了解细胞自动机的基本原理，并能实现对细胞自动机的模拟以及在此基础上做有价值的分析。

（3）基本要求

①编写 tribute 模型的模拟（最好有图形界面，可用 C 的图形库）。

②该模型是一个一维模型，包含 N 个参与者（细胞元，N 可配置，缺省为 10，此处就用 10 说明问题），将其命名为 actors。这 10 个 actors 组织成一个环形，每个 actor 都有一定的初始化财富量（可随机确定，在 300~500 之间均匀分布，也可设定），标记为 W。

③该模型的基本规则为：

1.Actors 只可以和它的邻居（左邻居和右邻居）进行交互，只有两种可能的交互动作：一是向其邻居索取供品（demand tribute），二是和其邻居结为联盟

（alliance）。

2.一个离散的时间片（表示 1 年）内，3 个随机选择的 Actors 一个接一个的被激活，每个 Actor（A）会向他的其中一个邻居（T）索取 Tribute，T 可以 pay，也可以拒绝并 fight，如果 pay，T pay A min（250,WT）；如果 fight，交战双方都会造成财富损失，每方损失的是对方财富量的 25%，如果其中一方的财富不足以承担这一数量时，双方的损失将成比例缩减，此时财富不足的那一方（ B ）将失去全部财富， 引起另一方（ A ） 的财富损失为 0.25*（B/（0.25*A））*B。

3.A 选取 T 的原则是使得 WT * （WA－WT）/WA 最大，但如果没有 T 使得该值大于 0，A 选择不动作。

4. T 选取 pay 或 fight 的哪一动作依据的是哪一种动作花费更少。

5.另外，每一年，每个 Actors 都有少量额外的财富（20）补充进来。

6.作为A 和B 交互的副产品，会产生一个A 对B 的承诺值（Commitments），简称 C，C 会在如下三种情况下增加：A pays tribute to B；A receives tributefrom B；和 A fights on the same side as B。而当 A fight on opposite sides withB 时 C 会降低。

7.C 值的范围为 0~1，每次增加及减少幅度都是 0.1。

8.初始化时，各 Actors 对其它 Actor 的 C 值为 0。又由于规则 6 的对称性，所以任何时刻 A 对 B 的 C 值和 B 对 A 的 C 是相同的。

9.引入 Commitments 后，A 选取 T 的范围扩大，不仅仅是其邻居 Actor，可以是任何一个 Actor。当 A 收取 T 的贡品时，位于 A 和 T 之间的 Actor 会根据其与 A 和 T 的 C 值大小决定加入 C 值较大的一方，如果两 C 值相同，则保持中立。只有当 A 和 T 之间的所有 Actor（环形的两个方向均可）都加入 A 方，A 才能向 T 收取贡品。对于存在若干个满足上述条件的 T，选取原则将按照规则 3 进行扩展。

10.规则 9 中，如果 K 要加入 A（T），仅当 K 和 A（T）相邻，或 K 和 A（T）之间的所有 Actor 都加入 A（T）。

11. A 形成的联盟的财产计算为：W= WA+C（A1,A）*WA1+…，其中 C（A1,A）为 A1 对 A 的承诺值，A1 为联盟中的一名成员，WA1 为 A1 的财产。收取的贡品仅归 A 所有，而战斗的负担由联盟共同承担，其费用比例就是各 Actor 的 C

值。

12. 所有 Actor 的 C 值、W 值对任何一个 Actor 都是可见的，在任何计算中都可使用。

④ 上述规则描述了 Tribute 模型，规则 1 到 5 是未引入 Commitments 时的基本模型，基本模型的模拟是本设计的要求内容；加上规则 6 到 12 是引入 Commitments 后的扩展模型，扩展模型的模拟是本设计的选做内容。

⑤ 模拟时间应超过 1000 年。

⑥ 根据上述模拟得出 Tribute model 较为全面的实验结果：财富的变化规律；各参数对结果的影响等。

（4）实现提示

可用循环队列实现自动机的存储

24 基于细胞自动机（Cellular Automata）实现Gossip 模型的模拟与分析

（1）问题描述

细胞自动机是一个时间和空间都离散的动态系统，每个细胞自动机都是由细胞单元（cell）组成的规则网格，每个细胞元都有 k 个可能的状态，其当前状态由其本身及周围细胞元的前一状态共同决定。

例如：The Game of Life

1．细胞元形成二维数组。

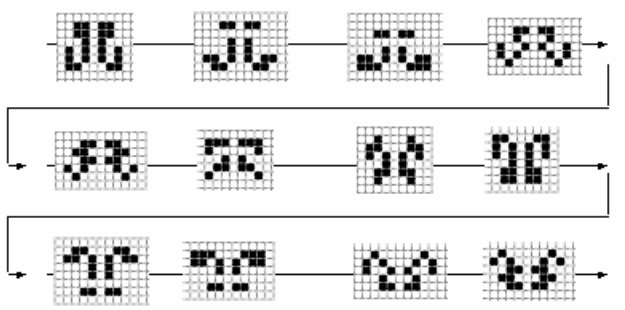
2．每个细胞元有两个状态：living 和 dead。

3．t+1 时刻的状态由 t 时刻的状态决定。

4．状态变迁规则 1：一个 living 的细胞元依旧保持存活，如果其周围存在 2 或 3 个 living 细胞元，否则死亡。

5．状态变迁规则 2：一个 dead 的细胞元依旧保持死亡，除非其周围恰好 3 个 living 细胞元。

一个演化图例如下：



细胞自动机从某种程度上反映了生物群落的演化，也可应用到许多计算机问题中。

（2）课程设计目的

了解细胞自动机的基本原理，并能实现对细胞自动机的模拟以及在此基础上做有价值的分析。

（3）基本要求

①编写 Gossip 模型的模拟（最好有图形界面，可用 C 的图形库）。

②该模型是一个二维模型，并按如下条件处理边界问题：右边界的右邻居是左边界，同理，左边界、上边界、下边界也一样，

③该模型的基本规则为：如果你的邻居中有人得知某消息，那么你将有 5%的概率从其中获知消息的一个邻居那里获得该消息(即如果只有一个邻居有消息，得知概率为 5%，有两个邻居就是 10%，依此类推)；如果你已经得知该消息，那么你将保存。

④依据上述模拟得出 Gossip 模型较为全面的实验结果：消息覆盖率随时间的变化；不同概率值的影响等。

（4）实现提示

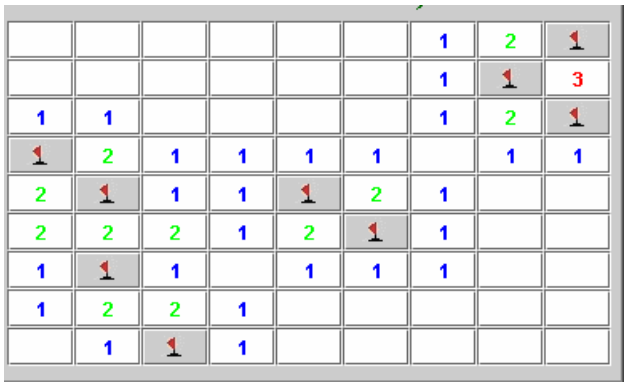
可用数组实现自动机的存储。

25 扫雷游戏

（1）问题描述

本题目做一个 N x M 的扫雷游戏，每个方格包含两种状态：关闭（closed）和打开（opened），初始化时每个方格都是关闭的，一个打开的方格也会包含两种状态：一个数字（clue）和一个雷（bomb）。你可以打开（open）一个方格，如果你打开的是一个 bomb，那么就失败；否则就会打开一个数字，该数字是位于[0，8]的一个整数，该数字表示其所有邻居方格（neighboring squares）所包含

的雷数，应用该信息可以帮助你扫雷。



具体实现要求的细节：

①能够打开一个方格，一个已打开的方格不能再关闭。

②能够标记一个方格，标记方格的含义是对该方格有雷的预测（并不表示真的一定有雷），当一个方格标记后该方格不能被打开，只能执行取消标记的操作，只能在取消后才能打开一个方格。

③合理分配各个操作的按键，以及各方格各种状态如何合理显示。

（2）课程设计目的

掌握线性结构的应用，并体会如何用计算机程序完成一个有趣的游戏。

（3）基本要求

①能够给出游戏结果（输、赢、剩余的雷数、用掉的时间按妙计）。

②游戏界面最好图形化，否则一定要清楚的字符界面。

（4）实现提示

用二维数组表示 N x M 区间，要仔细考虑如何初始的布置各个雷以及各个方格的状态及变化。

26 应用不相交集合生成随机迷宫

（1）问题描述

在本设计题目中，需要使用不相交集合数据结构（disjoint set data structure）来构造一个 N×N 的从左上角到右下角只有一条路径的随机迷宫，然后在这一迷宫上执行深度优先搜索。

该设计共包含如下四个部分：

①不相交集合数据结构的设计和实现

不相交集合即对于任意两个集合 A 和 B，A∩B=ø。不相交集合常可以表示为树，此时两个不相交集合的并的实现很容易，如图所示。不相交集合常可用来根据等价关系对集合进行等价划分。

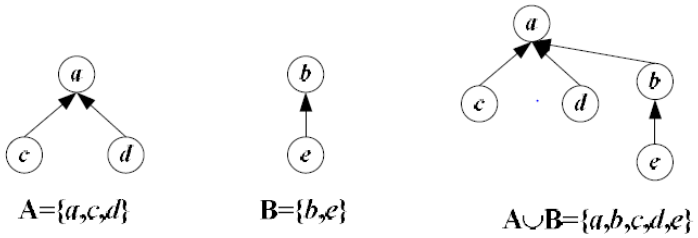


图 6-57 不相交集合的树表示

②构建随机迷宫

应用不相交集合构建迷宫的算法简要描述如下：给定一个 N×N 的方格（cells），初始时每个方格的四面都是墙（walls），如图 6-58（a）所示，其中的 S 是迷宫的开始处，F 是迷宫的结束处。N×N 迷宫的 N^2 个方格 0，1，…，N^2-1 初始时每个方格自己成为一个等价类，即{0}，{1}，…，{N^2-1}。生成随机迷宫的方法是随机选择一个内部墙（连接两个相邻方格的墙），如果该内部墙关联的两个相邻的方格属于不同的等价类就将该墙除去，在除去该墙的同时将这两个等价类合并。直到所有的方格都在一个等价类中，就完成了随机迷宫的生成。

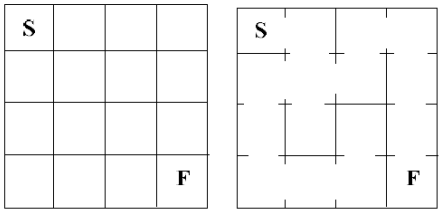


图 6-58 N × N 的迷宫

③寻找迷宫路径

迷宫一旦建立后，将迷宫表示为一个无向图：方格作为图中的顶点，如果两个相邻的方格之间没有墙则两个顶点之间有边。为找到从 S 到 F 的一条唯一的路径，在该图上从 S 处开始出发先深搜索，如果搜索到达了 F，则搜索停止。

④将迷宫和路径用图形方式画出

用图形方式将上述算法获得的随机迷宫及其上的最短路径画出。用线段来表示迷宫中的墙，用在每个方格中心的点来表示路径。

（2）课程设计目的

掌握不相交集合结构的实现和使用，掌握图的基本算法，建立集合、等价划分、图、搜索等事物之间如何关联的认识。

（3）基本要求

①不相交集合应用数组进行物理存储。

②在生成随机迷宫时，考虑如何使选取墙的次数尽量少。

③在先深搜索找到路径时，要求找到最短的路径，即记录最短路径上的方格，所以先深搜索过程应该是采用栈的非递归实现。此时，在先深搜索结束时，栈中就存放了最短路径上的方格，而不是先深搜索过程访问的所有方格。

④迷宫和路径的图形显示的实现方式不限制，如可以选择 VC，TC 或 Java 等。

（4）实现提示

不相交集合可父链数组进行物理存储，先深搜索采用栈的非递归实现可参阅一些资料，这样的资料很多。

27 指针式屏幕时钟

要求：

（1）正确显示系统时钟；

（2）能准确定位时钟刻度和时分秒针的位置；

（3）能随窗口大小的变化而变化。

算法设计及程序设计中技术重点：

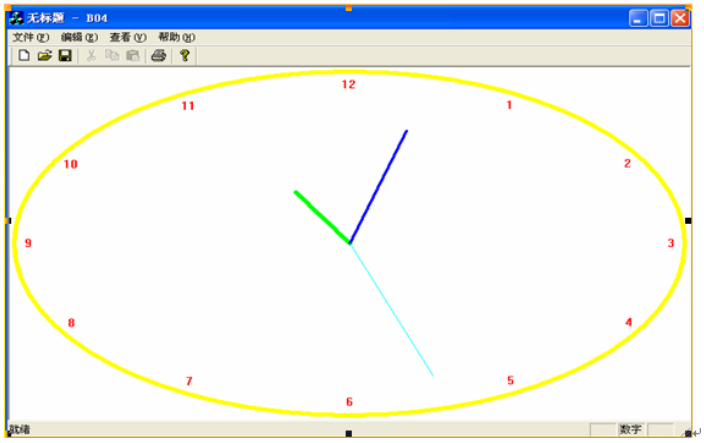
在应用程序中，经常有一些任务在后台处理，实现方式有两种：计时器和 OnIdle 循环处理。计时器是程序中最常用的后台任务机制之一，其时间间隔最低约 55 毫秒，被广泛用于时钟、磁盘备份程序或需要在某一时刻运行的程序等。

多媒体计时器能编程设定 1 毫秒或者更小，它是诸如 MIDI 序列发生器之类的专用型应用程序的理想选择，在 Windows API 中有很多查询时钟的函数，利用它们就可以编写出高精度的计时器。

使用计时器只需要了解两个函数：CWnd::SetTimer（）函数用来设置一个计时器以指定的时间间隔触发，CWnd::SetTimer（）函数用来使一个正在运行的计时器停止。计时器以两种方式通知应用程序计时器间隔时间已到：发送 WM-TIMER 消息和调用应用程序定义的回调函数。其中前者相对比较简单，但

对于多个计时器则应使用回调函数。计时器消息发送给应用程序时都是低优先级，因此只有当消息对列中没有其他消息时才回处理它们。

计时器消息不能在消息队列中积存，这避免了出现永远无法清空消息队列的状态。尽管如此，Windows 应用程序决不应该花费过量的时间来处理消息，除非该处理过程已经委派给辅助线程。如果主线程运行时间过长而没有检查消息队列，则程序的响应能力会受到影响。



程序运行结果图

28 平衡二叉树（AVL树）

编程实现平衡二叉树，功能要求：

能实现插入结点、删除结点、查找元素，并需显示调整平衡的过程。

29 B树

编程实现 B 树，功能要求：

①插入关键字

②删除关键字

③查找关键字

30 B+树

编程实现 B+树，功能要求：

①插入关键字

②删除关键字

③查找关键字

31 Red-Black Tree

（1） 问题描述

一棵具有红黑特征的树是一棵特殊的二元查找树，它在每个结点上增加一种额外属性——颜色（只能是红或黑）。

红黑树（Red-Black Tree）是一棵具有如下特征的二元查找树：

1．每一个结点或者是红色或者是黑色。

2．每个叶结点都是黑色。

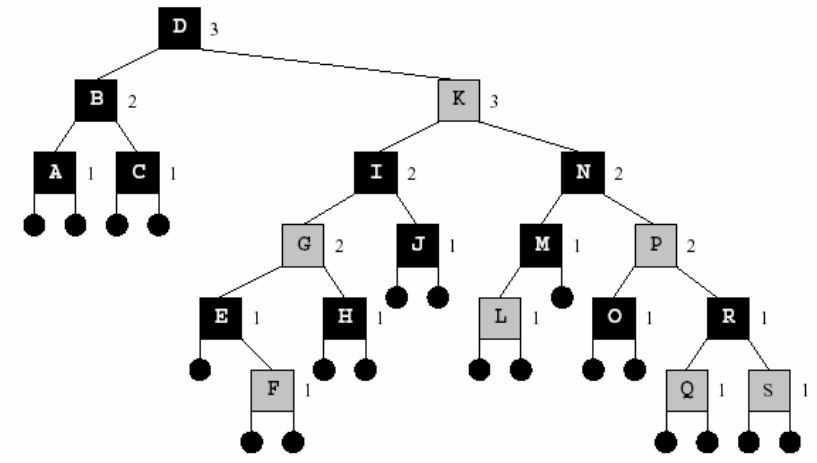
3．如果一个结点是红色，那么它的两个孩子结点都是黑色。

4．从任何一个结点到所有以该结点为子树的叶结点的简单路径上拥有相同的黑色结点。如下图就是一个红黑树的实例：

一般叶结点是不存放数据的，它作为哨兵用来表示已经到达叶结点。

从一个结点 x 到 x 子树中叶结点的路径（不包括 x）上黑色结点的个数称为 x 的黑色高度（black-height），标记为 bh（x）。

对于红黑树我们可以证明如下定理：任何一个具有 n 个内部结点的红黑树其高度至多为 2log（n+1）。该定理决定了红黑树可以保证查找时间复杂性一定时 O（logn）（具有和平衡树类似的性质）。



（2）课程设计目的

认识 Red-Black Tree 结构，并能依据其优点解决实际问题。

（3）基本要求

①设计并实现 Red-Black Tree 的 ADT，该 ADT 包括 Tree 的组织存储以及其上的基本操作：包括初始化，查找，插入和删除等。并分析基本操作的时间复

杂性。

②实现 Red-Black Tree ADT 的基本操作演示（要求应用图形界面）。

③演示实例为依次插入 A L G O R I T H M 并依同样的次序删除。

（4）实现提示

考虑树的旋转。

32 构造哈夫曼树

设计程序以实现构造哈夫曼树，要求如下：

①构造哈夫曼树。

②能演示构造过程。

③求解出所构造的哈夫曼树的带权路径长度。

④给出哈夫曼编码。

33 哈夫曼树编码文件压缩

采用哈夫曼编码思想实现文件的压缩和恢复（解压缩）功能。

要求：

① 实现文本文件的压缩和解压缩功能。

② 运行时的压缩原文件的规模应不小于 5KB。

③ 给出文件的压缩比。

④ 提供恢复文件与原文件的相同性对比功能。

34 Treap结构上的基本操作

（1）问题描述

如果将 n 个元素插入到一个一棵二元查找树中，所得到的树可能会非常不平衡，从而导致上面的查找时间很长。一种通常的处理办法是首先将这些元素进行随机置换，即先随机排列这些元素，然后再依次插入到二元查找树中，此时的二元查找树往往是平衡的，这种二元查找树称为随机二元查找树。但是这样的处理存在一个问题，即这样的处理方法只适用于固定的元素集合（已经预先知道了所有的元素）。如果没有办法同时得到所有的元素的话，即一次收到一个元素，则没有办法进行处理，此时可以用 treap 来处理这种情况，图 6-61 给出了一个 treap 结构。

Treap 结构中每个节点 x 有两个域，一个是其关键字值 key[x]，一个是优先级数 priority[x]（它是一个独立选取的随机数），上图节点中左边的字符就是 key[x]，右边的整数就是 priority[x]。对于 treap 结构，其关键字遵循二元查找树性质，其优先级遵循最小堆性质，即：如果 v 是 u 的左儿子，则 key[v]<key[u]，如果 v 是 u 的右儿子，则 key[v]>key[u]，如果 v 是 u 的儿子，则 priority[v]>priority[u]。所以这种结构被命名为 treap（tree+heap）。

有了 treap，假设依次插入关键字到一棵 treap 中，在插入一个新节点时：首先给该节点随机选取一个优先数；然后将该节点按关键字插入到 treap 中的二元查找树中，此时 priority 可能会不满足堆的性质；最后按照 priority 调整 treap，在保证二元查找树性质的同时调整 treap（需要一系列旋转）使之按 priority 满足堆性质。图 6-62 给出了在 treap 结构上进行插入操作的处理过程。

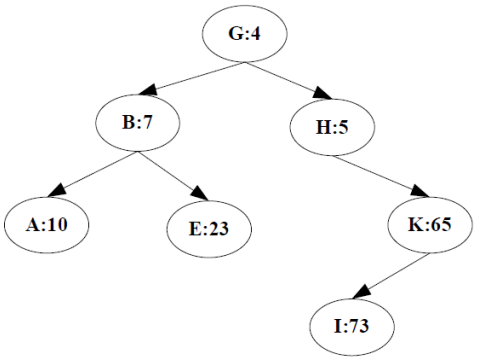
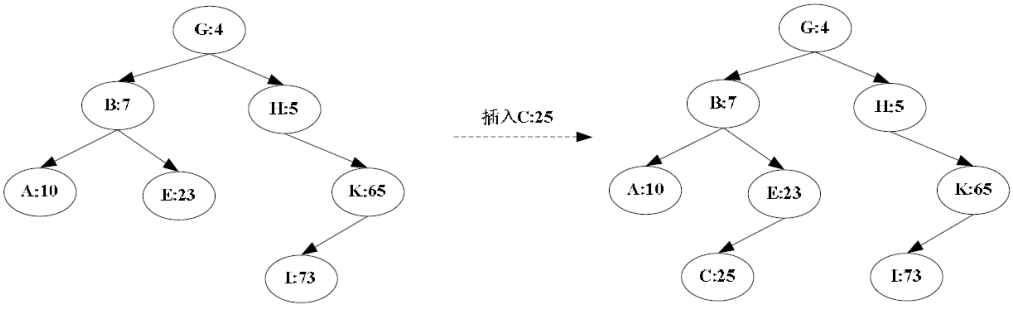
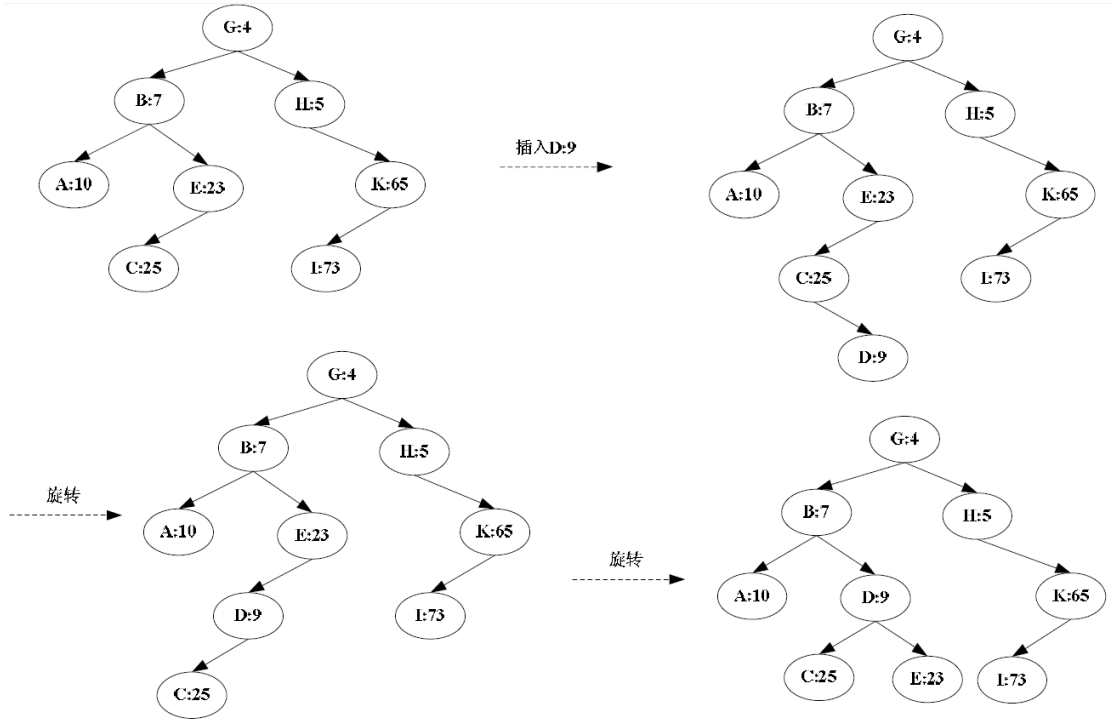


图 6-61 一个 Treap 结构



(*a*)



(*b*)

（2）课程设计目的

认识 treap 结构，对二元查找树和随机二元查找树建立深入的认识，能编程实现二元查找树、堆、treap 等结构以及其上的基本操作。

（3）基本要求

① 用数据结构知识设计和实现 treap 结构以及上的基本操作：插入和删除。

② 编程实现随机二元查找树。

③ 产生若干个依次插入二元查找树的元素序列（元素个数自己设定），针对该序列分别实现三种情况：插入到一般二元查找树、插入到 treap、用随机二元查找树处理。分别得出三种情况处理后的结果二元树的高度，进行实际的数据对比，从对比结果中发现一些规律。

④ 应模拟出各个操作的结果，模拟过程应该是可视的，即可以看到依次插入元素后各结构的变化情况。

⑤ 对随机二元树的高度、对 treap 的高度、对 treap 中的插入操作、对 treap 插入操作中旋转次数等指标进行理论分析。

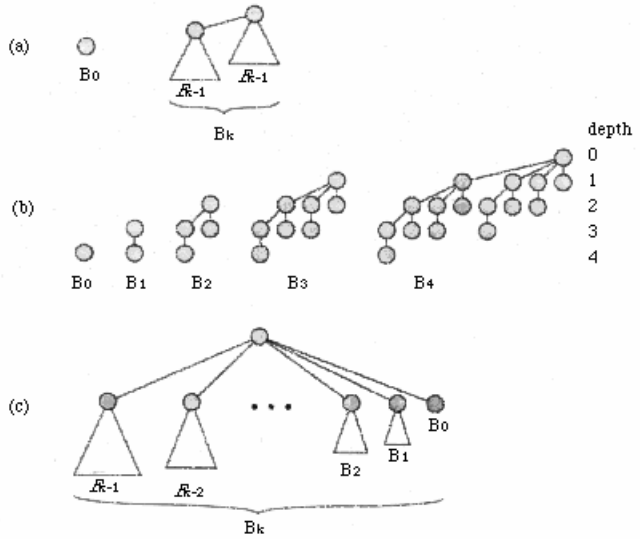
（4）实现提示

需要设计一个合适的随机排列算法来完成随机二元查找树的实现。

35 Binomial heaps的实现与分析

（1） 问题描述

二项堆是二项树的集合，而二项树是基于递归定义的，即二项树 B_k 是一棵在 Bk-1 的基础上递归定义的有序树，它由两个 B_{k-1} 形成链接形成：一棵 B_{k-1} 树的根是另一棵 B_{k-1} 根的最左儿子。下图给出了图示：



图（a）说明二项树的递归定义，图（b）给出 B_0~B_4 的图例，图（c）表明了 B_k 的另一种表现形式。

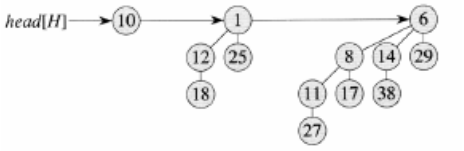
二项堆 H 是满足如下二项堆特征的二项树集合：

1．H 中的每棵二项树都满足堆特征，即每个结点的值都大于等于其父结点的值。

2．H 中的每棵二项树根结点的度互不相同。

其中性质 2 意味着对于任何 k，H 中只可能是存在或不存在 B_k。如果 H 中存在 B_k，就会对 H 贡献 2^k 个结点，用二进制表示就是在第 k 为 1，所以对于 n 个结点的 H，n 的二进制表示[$b_{\log n}$, $b_{\log n-1}$ ….. b_0]中第 i 为 b_i 就表示了 H 中是否存在 B_i，相反的事实也成立。根据这一事实也可以看出 n 个结点的二项堆至多$\lfloor \log n \rfloor + 1$ 个二项树。

下图为一二项堆的实例：



（2）课程设计目的

认识二项树、二项堆数据结构，并能应用该结构解决实际问题。

（3） 基本要求

① 设计二项堆 ADT，其上的基本操作包括：Make Heap （x），Find-Min, Union，Insert，Extract-Min，Decrease Key （x），Delete。

②实现二项堆 ADT，包括实现二项堆的存储结构以及其上的基本操作，并分析基本操作的时间复杂性。

③实现二项堆 ADT 的基本操作演示（要求应用图形界面）。

（4）实现提示

其中的 UNION 是关键操作，部分其它操作可在此基础上构造。UNION 操作可类似二进制加法进行。

36 应用堆实现一个优先队列并实现作业的优先调度

（1）问题描述

优先队列 priority queue 是一种可以用于很多场合的数据结构，该结构支持如下基本操作：

•Insert（S, x）——将元素 x 插入集合 S

•Minimum （S）——返回 S 中最小的关键字

•Extract－Min （S）——删除 S 中的最小关键字

•DecreaseKey（S,n,key）——将 S 中的结点 n 的关键字减小为 key

要求以堆作为辅助结构设计并实现一个优先队列。要将堆结构嵌入到队列结构中，以作为其数据组织的一部分。

应用该优先队列实现作业的优先调度：

一个作业 t_i =（s_i，e_i），s_i 为作业的开始时间（进入时间），e_i 为作业的结束时间（离开时间）。作业调度的基本任务是从当前在系统中的作业中选取一个来执行，如果没有作业则执行 nop 操作。本题目要求的作业调度是基于优先级的调度，每次选取优先级最高的作业来调度，优先级用优先数（每个作业一个优先数 p_i）表征，优先数越小，优先级越高。作业 t_i 进入系统时，即 s_i 时刻，系统给该作业指定其初始优先数 $p_i = e_i - s_i$，从而使越短的作业优先级越高。该优先数在作业等待调度执行的过程中会不断减小，调整公式为：$p_i = p_i - w_i$，其中的 w_i 为作业 t_i 的等待时间：w_i= 当前时间-s_i。一旦作业被调度，该作业就一直执行，不

能被抢占，只有当前执行作业指向完成时，才产生下一轮调度。所以可以在每次调度前动态调整各作业的优先数。

编程实现这样一个作业调度系统。

（2）课程设计目的

熟悉堆结构的应用，优先队列的构造，解决实际问题。

（3）基本要求

①给出优先队列的 ADT 描述，包括队列的逻辑结构及其上基本操作。

②以堆结构为辅助结构实现优先队列的存储表示并实现其上的基本操作。

③作业集合中的各作业随机生成，根据作业的 s 属性和 e 属性动态调整作业队列，不断加入作业，作业结束删除作业。

④要对作业调度的结果给出清晰的输出信息，包括：何和作业进入，何时调度哪个作业，何时离开，每个作业等待多长时间，优先数的动态变化情况等。

37 应用小大根交替堆实现双端优先队列

（1）问题描述

双端优先队列是一个支持如下操作的数据结构：

•Insert （S, x） – 将元素 x 插入集合 S

•Extract – Min （S） – 删除 S 中的最小关键字

•Extract – Max （S） – 删除 S 中的最大关键字

可用小大根交替堆来实现对上述三个操作的支持。小大根交替堆是一个满足如下小大根交替条件的完全二元树：如果该二元树不空，那么其上的每个元素都有一个称为关键字的域，且针对该关键字，二元树按层次形成了小大根交替的形式，即对于小大根交替堆中的任何一个结点 x，如果 x 位于小根层次，那么 x 就是以 x 为根节点的二元树中键值最小的结点，并称该结点为一个小根结点。同样的道理，如果 x 位于大根层次，那么 x 就是以 x 为根节点的二元树中键值最大的结点，并称该结点为一个大根结点。在小大根交替堆中根结点位于小根层次。

下图给出了一个具有 12 个顶点的小大根交替堆实例。每个结点中的数值就是该结点对应元素的键值。

假设要在该图的堆中插入键值为 5 的元素。和普通堆一样，小大根交替堆上的插入算法也要针对从新插入结点到根结点的路径进行处理。对于上面的例子，

需要比较键值 5 和其父结点键值（此处是 10）的大小关系，此处由于 10 位于小根层次且 5<10，就决定了 5 一定比从新加结点到根结点路径上的所有大根结点都要小，也即 5 无论放在那里都不影响大根层次的大根性质，所以插入 5 时只需调整从新加结点到根结点路径上的小根结点就能使得整个堆满足小大根交替性质。具体的调整过程为：首先，由于 5<10，所以 10 下移填充到新加结点中；接着，5 和 7 比较，由于 5<7，所以 7 下移填充到原来结点 10 的位置；最后将结点 5 填入根结点。

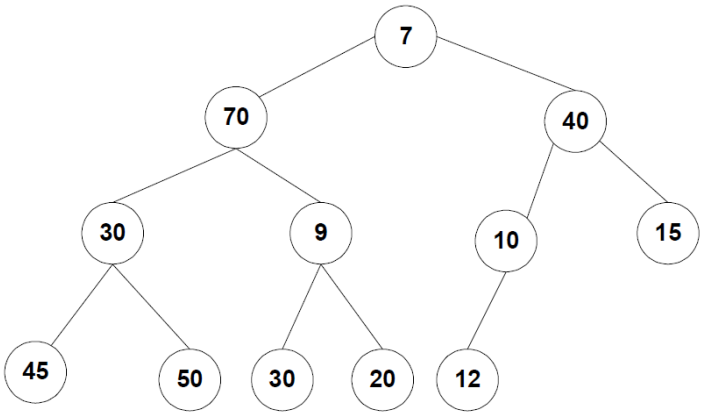


图 6-33 一小大根交替堆实例

下面我们将考虑在上图中插入键值为 80 的元素的情况。此时 10 仍处于小根层次，但由于 80>10，就有 80 大于从新加结点到根结点路径上的所有小根结点，也就决定了插入 80 时只需调整从新加结点到根结点路径上的大根结点就能使得整个堆满足小大根交替性质。上图中只有一个满足条件的结点——键值为 40 的结点。由于 80>40，所以 40 将下移填入新加结点中，而 80 将填入原 40 结点所占据的位置。

如果我们要删除该堆中键值最小的元素，那么删除的显然就是根结点，对于上图所示的小大根交替堆，就是删除结点 7。删除了结点 7 以后，小大根交替堆中就剩下了 11 个元素，并拟用最后一个元素（键值为 12）重填根结点。和普通的小根（大根）堆类似，重填的动作将引起从根到叶的一遍检查。

考虑在堆中实现元素 item 对根结点的重填，需分析如下两种情况：

（1）如果根没有儿子结点。

此时 item 直接填入根结点即可。

（2）如果根至少存在一个儿子结点。

此时，小大根交替堆中键值最小的结点一定出现在根结点的儿子结点或孙子结点中，并且可以很容易的找到该结点，将其标记为 k。然后再分成如下几种情况进行考虑：

a）item.key <= heap[k].key。

此时 item 直接插入根结点处，因为 item 就是堆中键值最小的结点。

b）item.key > heap[k].key 且 k 是根结点的儿子。

由于 k 是大根结点，所以其后代结点中不可能有键值大于 heap[k].key 的结点，也就没有键值大于 item.key 的结点。此时将 heap[k]的元素移到根结点，然后将 item 插入 k 结点所在位置即可。

c）item.key > heap[k].key 且 k 是根结点孙子。

对于此种情况也是先将 heap[k]的元素移到根结点，将 item 插入 k 结点所在位置。然后考察 k 的父结点 parent，如果 item.key>heap[parent].key，就将 heap[parent]和 item 元素互换，这一交换保证了 parent 结点处的键值是以 parent 为根结点的子堆中最大的，即 parent 满足了大根条件。此时需考虑如何将 item 插入到以 k 为根的子堆中，现在该子小大根交替堆的根结点为空，显然此时的情况和初始情况完全相似，因此可以重复上述处理过程。

对于本例有 item.key=12，且根结点的所有儿子和孙子中的最小键值为 9，设 9 所对应的结点用 k 标记，其父结点用 p 标记。由于有 9<12 且 k 是根的孙子结点，属于情形 2（c）。所以键值 9 对应的元素（即 h[k]）将移到根结点处，又由于条件 item.key=12<70=h[p].key，所以不需要交换 x 和 h[p]。现在需将 item 插入到以 k 为根的子小大根交替堆中，由于 k 的所有儿子和孙子结点中最小的键值为 20，将上条件 12<20，属于情形 2（a），将 item 插入到 h[k]中即可，至此完成了插入过程。

（2）课程设计目的

掌握小大根交替堆，并用它实现双端优先队列的构造。

（3）基本要求

①给出双端优先队列的 ADT 描述，包括优先队列的逻辑结构及其上基本操作。

②给出小大根交替堆的 ADT 描述，并实现该 ADT。

③以小大根交替堆结构为辅助结构实现双端优先队列的存储表示并实现其上的基本操作。

④应用双端优先队列的 ADT 实现依据学生成绩实现对学生信息的查询。

⑤学生信息存放在文本文件中（格式自定，内容自行输入）。

（4）实现提示

出队、入队需考虑优先性。

38 N-ary Trie的实现和分析

（1） 问题描述

哈夫曼算法输出的结果就是一个二元 Trie，在二元 Trie 中，每个左子树分支用 0 表示，右子树分支用 1 表示。如下图就是就是一个二元 Trie 的示例图。

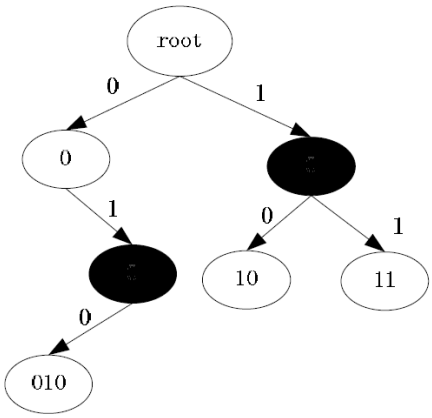


图 6-13 二元 Trie 的结构示意图

可以将二元 Trie 扩展到 N-ary Trie。在 N-ary Trie 中，每个结点都有 0~N 之间任何个儿子结点，其中每个分支都用一个相应的符号表示（在 N-ary Trie 中有 N 个不同的符号）。

例如一个电话本可用一个 N-ary Trie 表示，其中 N=10，分别表示 0~9 的十个数字，具体情况如下图所示：

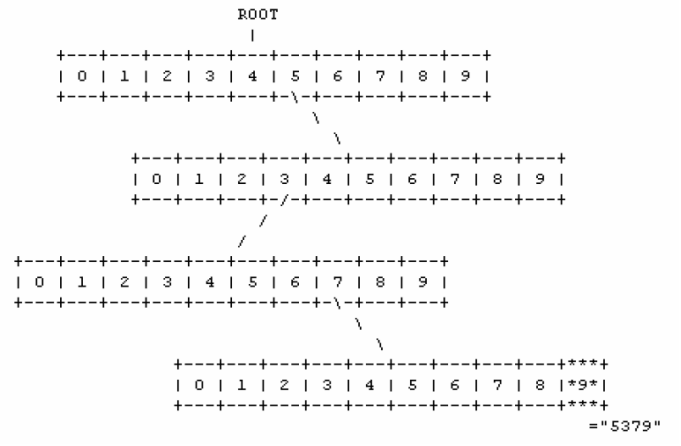


图 6-14 *N* 元 Trie 的结构示意图

（2）课程设计目的

认识 N-ary Trie 结构，并能应用该结构解决实际问题。

（3）基本要求

①设计并实现 N-ary Trie 的 ADT（N=26，建立在英语上的 Trie），该 ADT 包括 Trie 的组织存储以及其上的基本操作：包括初始化，查找，插入，删除等。

② 应用 Trie 结构实现文本文档的索引化。首先扫描文本文档（存放在 txt 文件中），然后在此基础上用 Trie 登记单词出现的位置（行号），最后在 Trie 上实现查询。

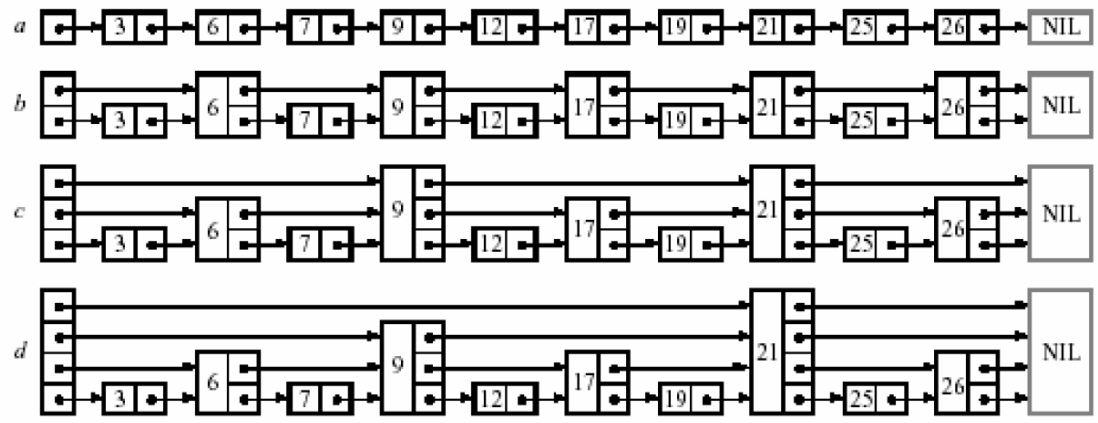
③用户输入的查询可以是针对一个单词的，也可是针对一个字符序列的。

（4）实现提示

在树结构的基础上进行适当的修改。

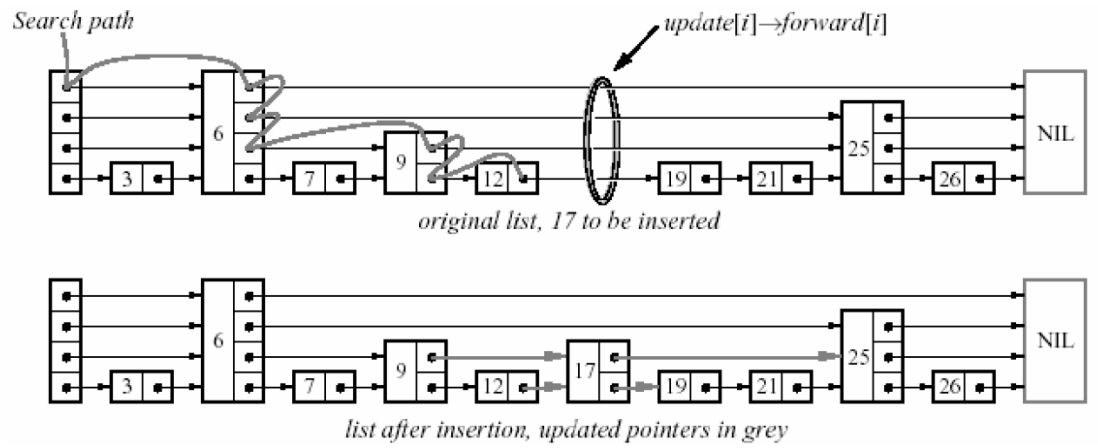
39 Skip List的实现

SkipList 作为有序链表结构的一种扩展，如下图所示，其中 a 是普通的单链表；而 b 是在次基础上加上第二层（level 2）的额外指针，这些额外的指针指向间隔为 2 的下一个结点，skip list 因此得名；类似的 c 是加上 level 3 后的 skip list；d 是加上 level 4 后的 skip list。



Skip List 上查找的基本思想是先从最高的 Level 层上查找，找到 key 所在的范围后，再从较低的层次继续重复查找操作，直到 Level 1。

Skip List 上的插入操作如下图所示。



Skip List 上的删除操作只需直接删除元素即可（包括局部范围内的指针调整）。

本设计题目的基本内容是构造并实现 Skip List 的 ADT，并能对其维护动态数据集合的效率进行一定的实验验证。

（2）课程设计目的

认识并应用 Skip List 数据结构，体会线性表结构的变形形式。

（3）基本要求

①ADT 中应包括初始化、查找、插入、删除等基本操作。

②分析各基本操作的时间复杂性。

③针对实现 Skip List 上基本操作的动态演示（图形演示）。

④能对 Skip List 维护动态数据集合的效率进行实验验证，获得一定量的实验数据，如给定随机产生 1000 个数据并将其初始化为严格 Skip List，在此基础

上进行一些列插入、删除、查找操作（操作序列也可以随机生成），获得各种操作的平均时间（或统计其基本操作个数）；获得各操作执行时间的变化情况，应该是越来越大，当大到一定程度后应该进行适当的整理，需设计相应的整理算法，并从数量上确定何时较为合适；能和其他简单线性数据结构，如排序数组上的折半查找进行各类操作效率上的数量对比。

（4）实现提示

需仔细设计整理算法。

40 稀疏矩阵的完全链表表示及其运算

（1）问题描述

稀疏矩阵的每个结点包含 down，right，row，col 和 value 五个域。用单独一个结点表示一个非零项，并将所有结点连接在一起，形成两个循环链表。使得第一个表即行表，把所有结点按照行序（同一行内按列序）用 right 域链接起来。使得第二个表即列表，把所有结点按照列序（同一列内按行序）用 down 链接起来。这两个表共用一个头结点。另外，增加一个包含矩阵维数的结点。稀疏矩阵的这种存储表示称为完全链表表式。

实现一个完全链表系统进行稀疏矩阵运算，并分析下列操作函数的计算时间和额外存储空间的开销。

（2）设计目的

认识和掌握稀疏矩阵的完全链表表示；能够建立并运用这种存储结构。

（3）基本要求

建立一个用户友好、菜单式系统进行下列操作，并使用合当的测试数据测试该系统。读取一个稀疏矩阵建立其完全链表表示，输出一个稀疏矩阵的内容、删除一个稀疏矩阵、两个稀疏矩阵相加、两个稀疏矩阵相减、两个稀疏矩阵相乘、稀疏矩阵的转置。

（4）实现提示 链表上的操作

41 多项式链式存储结构及其代数运算

（1）问题描述

设计并建立一个链式存储分配系统来表示和操作多项式。为了避免对零和非零多项式进行不同的处理，使用带头结点的循环链表。为了充分利用多项式中不

再使用的结点，维护一个可用空间表 avail,把不再使用的多项式的结点链入其中。当需要一个新结点时，就查看这个单链表 avail。如果表非空，那么可以使用它的一个结点。只有当该表为空时，才使用动态存储分配来创建新结点。

（2）基本要求

设计多项式的存储结构，编写并测试下列函数：

a）get_node 和 ret_node，从/向可用空间表申请和插入一个多项式结点。

b）pread，读取一个多项式，并将其转换成循环存储表示。返回指向该多项式的头结点的指针。

c）pwrite，输出多项式，采用能够清楚显示的形式。

d）padd，计算 d=a+b。不改变 a 和 b。

e）psub，计算 d=a-b。不改变 a 和 b。

f）pmult，计算 d=a*b。不改变 a 和 b。

g）eval，计算多项式在某点 a 的值，其中 a 是一个浮点型常量。返回结果为浮点数。

h）perase，把存储表示为循环链表的多项式返还给可用空间表。

（3）实现提示

为了进一步简化加法算法，把多项式的头结点的指数域设为-1。

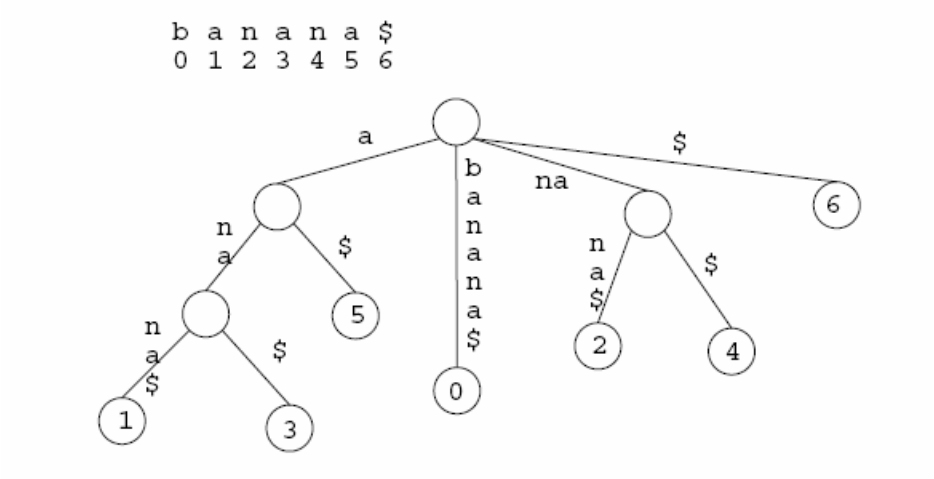
42 后缀树的构造

（1）问题描述

后缀树是一种数据结构，一个具有 m 个字符的字符串 S 的后缀树 T，就是一个包含一个根节点的有向树，该树恰好带有 m+1 个叶子（包含空字符），这些叶子被赋予从 0 到 m 的标号。每一个内部节点，除了根节点以外，都至少有两个子节点，而且每条边都用 S 的一个子串来标识。出自同一节点的任意两条边的标识不会以相同的字符开始。

后缀树的关键特征是：对于任何叶子 i，从根节点到该叶子所经历的边的所有标识串联起来后恰好拼出 S 的从 i 位置开始的后缀，即 S[i,…,m]。

后缀树的图示如下：



（2）设计目的

认识后缀树的结构，掌握其构造方法，并能个根据其特点解决实际问题。

（3）基本要求

a）对任意给定的字符串 S，建立其后缀树；

b）查找一个字符串 S 是否包含子串 T；

c）统计 S 中出现 T 的次数；

d）找出 S 中最长的重复子串。所谓重复子串是指出现了两次以上的子串；

e）分析以上各个算法的时间复杂性。

43 实现计算几何软件包

（1）问题描述

计算几何处理的基本对象是二维平面上的点和线段，以及靠这些点和线段形成的各种图形如凸多边形。在计算几何中，线段及其上的基本操作时计算几何的基础。

线段的基本操作如下：

① 叉积

对于 p1=（x1,y1）和 p2=（x2,y2），p1 和 p2 的叉积 p1　p2 定义为：

$$\begin{aligned} p_1 \times p_2 &= \det\begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} \\ &= x_1 y_2 - x_2 y_1 \\ &= -p_2 \times p_1 . \end{aligned}$$

如果 p1　p2 为正数，相对于原点来说，p1 在 p2 的顺时针方向上，如为负数，则在逆时针方向上。如果是对于公共端点 p0，则计算叉积（p1-p0）（p2-p0）

就可判断线段 p0p1 在 p0p2 的顺时针方向还是逆时针方向。

② 连续线段是向左转还是向右转

该问题考虑的是连续线段 p0p1 和 p1p2 在 p1 处是向左转还是向右转，如下图所示：

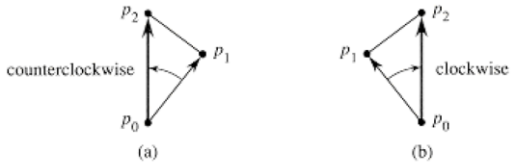


图 6-22 连续线段转向问题

可用叉积（p2-p0） （p1-p0）来计算该值，如果该叉积为负，则在 p1 点要向左转，如果该叉积为正，则在 p1 点要向右转。本课程设计的基本内容是实现上述基本操作并据此解决如下两个基本问题：

①确定两条线段是否相交

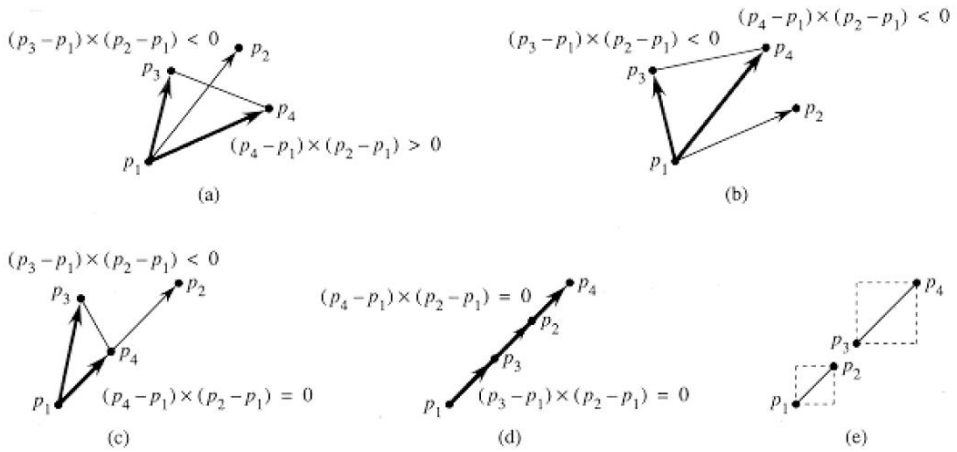


图 6-23 两条线段的相交问题

②实现寻找凸包的 Graham 扫描法

（2）课程设计目的

应用数据结构与算法基本知识解决实际问题，对计算几何建立初步的认识。

（3）基本要求

①定义合适的数据结构，实现叉积、方向判断、相交判断、寻找凸包四个算法。

②算法演示中的点和线段在二维平面随机生成。

③Graham 扫描法要给出分析（最好是证明）算法的正确性，并分析算法的

时间复杂性。

④最好用图形界面说明算法结果。

（4）实现提示

无。

44 简单矢量图形的几何变换

1）问题描述

常见的几何图形（二维图形）都是由点、直线和圆组成，而平面上的点和直线都可用矢量进行表示，如果假设几何图形中不包含圆，那么一个无论多么复杂的几何图形都可用矢量表示并存储，且图形上的几何变换都可通过矢量上的变换得以实现。

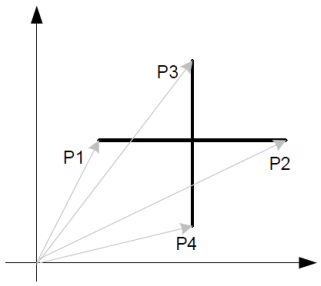


图 6-35 一矢量图形的实例

图中的十字形状就可用矢量 P1,P2-P1 和 P3,P3-P4 表示，当然其中 P1,P2,P3,P4 都是向量，分别表示十字图形的四个顶点。其中 P1 表示直线段的起点，P2-P1 表示该直线的方向和长度。

而该图的平移变换就可表示如下（向右水平平移 5 个单位）：

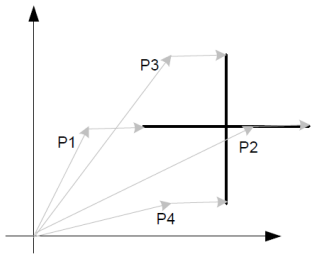


图 6-36 矢量图形向右平移 5 个单位的结果

[P1+H,P2-P1];[P3+H,P3-P4]其中 H 为矢量 5+0i。

（2）课程设计目的

能够应用线性数据结构描述简单的矢量图，并能完成简单的几何变换。

（3）基本要求

①描述并实现矢量 ADT，包括矢量的表示和存储，以及其上的基本操作：矢量相加；相减；共扼；求模；相乘等。

②应用矢量实现简单几何图形（由点和直线组成）的表示和存储。同时该几何图形能够保存在文件中（格式自定），并能从文件中读出。

③实现矢量几何图形的基本操作，包括平移（水平加垂直）；旋转；依直线镜像；依点镜像；放大；缩小等（可自行扩展）。

④提供界面实现用户对图形的操作，包括输入图形，进行变换操作等。

⑤每一步操作的结果都能图形化显示。

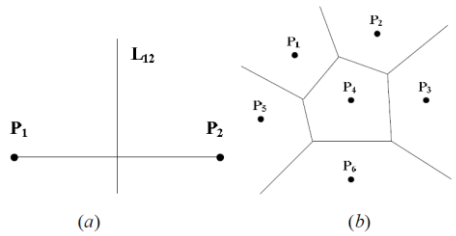
（4）实现提示

用二元组表示一个向量，一个几何图形就是一个数组。

45 Voronoi图及其简单应用

（1）问题描述

Voronoi（一位著名的俄罗斯数学家）图是一种非常有趣的数据结构，该数据结构可以用于表示有关平面点最近邻点的有关信息。图 6-82（a）给出的是两个点的 Voronoi 图，实际上就是这两个点的垂直平分线 L12。此时对于任意的点 X，不用计算 X 和 P1 和 P2 的距离，只要看 X 位于 L12 的哪一边（将 X 的坐标代入 L12 就能判断）就可以了。图 6-82（b）给出的是 6 个点的 Voronoi 图。



Voronoi 图可以用分治算法实现。

算法：构造 Voronoi 图的分治算法

输入：n 个平面点集合 S

输出：集合 S 的 Voronoi 图

第 1 步：如果集合 S 只有 1 个点，则返回。

第 2 步：找出垂直于 x 轴的中线 L，L 将 S 划分为相同大小的集合 SL 和 SR，SL 在 L 的左边，SR 在 L 的右边。

第 3 步：递归的构造 SL 和 SR 的 Voronoi 图，分别用 VD（SL）和 VD（SR）

表示各自的 Voronoi 图。

第 4 步：构造分段直线 HP，删除 HP 右边的 VD（SR）所有的线段，删除 HP 左边的 VD（SL）所有的线段，剩下的图就是 S 的 Voronoi 图。

图 6-83 给出了一个该分治算法的实例。

该分治算法的核心是将两个 Voronoi 图合并成一个 Voronoi 图的算法，现给出该算法。

算法：将两个 Voronoi 图合并成一个 Voronoi 图的算法

输入：VD（SL）和 VD（SR）

输出：VD（S）

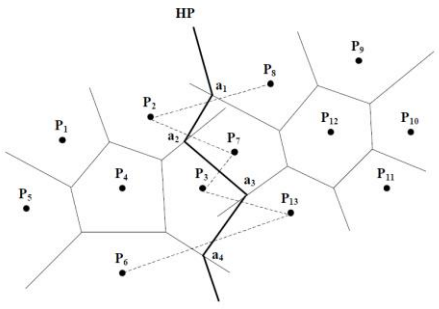
第 1 步：找出 SL 的凸包 Hull（SL），SR 的凸包 Hull（SR）。

第 2 步：找出将 Hull（SL）和 Hull（SR）合并成一个凸包的线段 PaPb 和 PcPd（Pa 和 Pc 属于 SL，Pb 和 Pd 属于 SR），假如 PaPb 在 PcPd 的上面，令 x=a，y=b，SG=PxPy，HP=　。

第 3 步：找出 SG 的垂直平分线，用 BS 表示。令 HP=HP　BS。如果 SG=PcPd，转向第 5 步，否则转向第 4 步。

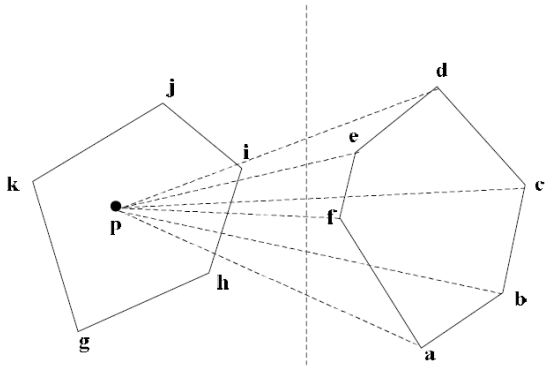
第 4 步：令 BS 首先与来自 VD（SL）或 VD（SR）的射线相交，该射线一定是相对于某 z 的 PxPz 或 PzPy 的垂直平分线，如果该射线是 PzPy 的垂直平分线，则令 SG=PxPz，否则令 SG=PzPy。

第 5 步：删除 HP 右边的 VD（SR）所有的边，删除 HP 左边的 VD（SL）所有的边，最终的图就是 S 的 Voronoi 图。



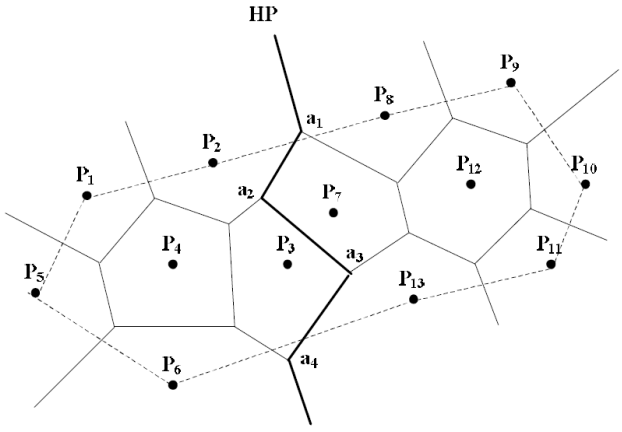
显然在该算法中，有两个算法需要详细描述，一是将 Hull（SL）和 Hull（SR）合并成一个凸包的两条线段；另一个是找出 S 的凸包 Hull（S）。对于如何找出两个凸包合并成一个凸包的线段，来看如图 6-84 的实例，显然从一个凸包内的任一个点向另一个凸包的所有点连线，将这些线的方向排序，显然向上倾斜和向

下倾斜度最大的两个顶点就是两个凸包合并成一个凸包的线段的在一个凸包中的两个端点，同样可以找到另一个凸包中对应的两个端点，对应的连接这两对端点就能得到将两个凸包合并成一个凸包的线段。



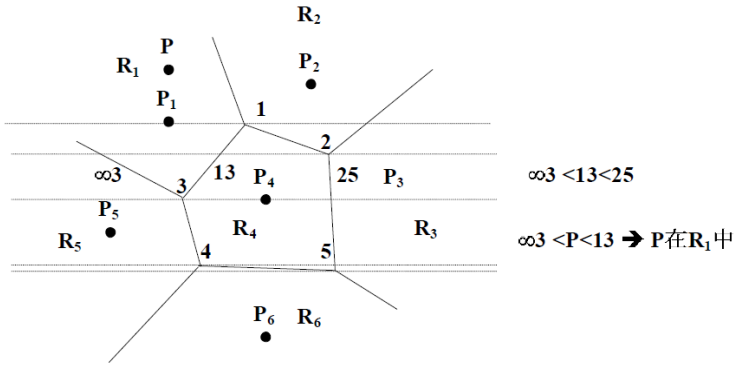
第二个需要回答的问题是如何找出 S 的凸包 Hull（S）。可以用 Voronoi 图的信息来求该凸包，如图 6-85 所示：即检查所有 Voronoi 边直到找到一条射线，设 Pi 是此射线左边的一点，那么 Pi 就是凸包的一个顶点，重复这一过程直到找到所有的射线，找出所有的凸包顶点，就获得了凸包。

下面来看一个 Voronoi 图的应用，解决欧几里得近邻搜索问题：已知平面上的 n 个点 P1，P2，…，Pn，和一个检测点 P，找出和 P 最近（欧几里得距离）的 Pi。一个非常简单的方法是依次求出 P 和每个点的距离，然后求出最小的距离，其时间复杂性是 O（n）。可以用 Voronoi 图来提高该搜索的时间。



下面给出应用 Voronoi 图解决欧几里得近邻搜索问题的基本想法：首先将所有的 Voronoi 顶点按照其 y 值排序，然后在每个 Voronoi 顶点处画一条水平线，这些线将整个空间分割成多个片隙。在每个片隙内可以排序 Voronoi 边，根据这些边可以很容易判断该片隙中的各个区域，因此可以根据 P 和这些 Voronoi 边的关系来判断 P 位于哪个区域，从而判断 P 和哪个点最近（由于已经排序，所以

可以用二分查找）。如图 6-86 所示。



（2）课程设计目的

认识 Voronoi 图，能够编程实现 Voronoi 图及其上的简单应用。

（3）基本要求

①编程实现上述求解 Voronoi 图的分治算法。

②编程实现应用 Voronoi 图解决欧几里得近邻搜索问题的算法。

③平面点在平面上随机放置，最好给出解的图形表示，以方便验证结果的正确性。

④在一台机器上对应用 Voronoi 图解决欧几里得近邻搜索问题的算法和求出所有距离后找最近的两个算法，并对这两个算法进行实验对比，点数分别为 100，300，500，1000，10000，5000。

⑤分析求解 Voronoi 图的分治算法时间复杂度和应用 Voronoi 图解决欧几里得近邻搜索问题的算法的时间复杂度。

（4）实现提示

需仔细思考 Voronoi 图的存储结构。

46 决策树算法实现

简介：决策树是通过一系列规则对数据进行分类的过程。它提供一种在什么条件下会得到什么值的类似规则的方法。它是一个从上到下、分而治之的归纳过程，是决策树的一个经典的构造算法。应用于很多预测的领域，如通过对信用卡客户数据构建分类模型，可预测下一个客户他是否属于优质客户。

分类过程：分类是数据挖掘、机器学习和模式识别中一个重要的研究领域。数据分类是一个两步过程。第一步，使用已知类别标记的训练数据集建立一个分类模型。例如：图 1 是一个决策树模型。第二步，对未知标记的数据使用模型进

行分类。例如，根据图 1 的决策树模型，运用自顶而下的属性测试过程，将表 2 中的样例 1-6 分别分类为“Y”、“Y”、“Y”、“Y”、“N”、“N”。

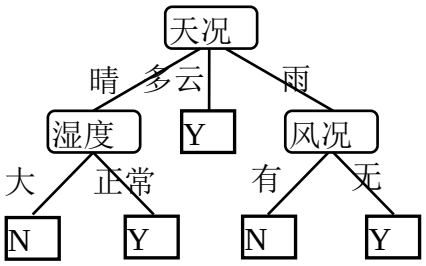


图 1.一个决策树模型的例子

算法描述：

输入：训练样例集 S，未标记的节点 T，属性集 A

输出：以 T 为根的决策树

①如果 S 中所有样例都是正例，则标记节点 T 为“Y”，并结束；

②如果 S 中所有样例都是反例，则标记节点 T 为“N”，并结束；

③否则，从 A 中选择一个属性 X，（可随机选）标记节点 T 为 X;

④设 X 的所有取值为 V1,V2,…,Vn，依据这些取值将 S 划分为 n 个子集 S1,S2,…,Sn，建 T 的 n 个孩子节点 Ti，并分别以 Vi 作为从 T 到 Ti 的分支标号;

⑤对每对（Si，Ti，A-{X}），递归调用 ID3 算法建立一棵以 Ti 为根的子树；

END

举例：对下表运用算法构建决策树

表 1.一个训练数据集

编号	天况	温度	湿度	风况	分类
1	晴	热	大	无	N
2	晴	热	大	有	N
3	多云	热	大	无	Y
4	雨	中	大	无	Y
5	雨	冷	正常	无	Y
6	雨	冷	正常	有	N
7	多云	冷	正常	有	Y
8	晴	中	大	无	N
9	晴	冷	正常	无	Y
10	雨	中	正常	无	Y
11	晴	中	正常	有	Y
12	多云	中	大	有	Y

13	多云	热	正常	无	Y
14	雨	中	大	有	N

对下列样例输入使用构建的决策树模型预测其分类属性：

表 2.一个待分类的数据集

编号	天况	温度	湿度	风况	分类
1	晴	热	正常	无	?
2	晴	热	正常	有	?
3	雨	热	正常	无	?
4	晴	中	正常	无	?
5	晴	冷	大	有	?
6	晴	冷	大	无	?

要求：

①设计合理的数据结构，编程实现决策树构造算法；

②给定训练数据集，运用构建的决策树模型，设计合理的文件格式，保存于外存之中；

③设计决策树分类算法，根据保存在外存的决策树模型，实现决策树的分类过程，完成对未知类别属性数据样例的分类。

47 关联规则求解算法Apriori的实现

简介：关联分析是数据挖掘中的一个重要任务，Apriori 算法是一种典型的关联分析算法。多用于超市的销售决策，如通过统计一段时间内，用户买商品 A 和 B 同时发生的概率，得出了顾客买 A 则很可能会买 B 的一条规则。

本课题要求对给定的训练数据，实现 Apriori 算法，构件关联规则集合。

Apriori 算法描述如下：

输入：训练样例集 S，属性集 A，支持度阈值 minsup，置信度阈值 minconf

输出：关联规则集

（1）令 k=1

（2）生成长度为 1 的频繁项集

（3）重复以下操作直到不在生成新的频繁项集

（a）剪除一些候选项集，它们包含非频繁的长度为 K 子集

（b）从长度为 k 的频繁项集中生成长度为 k+1 的候选项集

（c）通过扫描数据库，计算每个候选项集的支持度

（d）除去那些非频繁的候选项集，只留下那些频繁的候选项集

END

示例：

基本概念表：关联规则的简单例子

TID	网球拍	网球	运动鞋	羽毛球
1	1	1	1	0
2	1	1	0	0
3	1	0	0	0
4	1	0	1	0
5	0	1	1	1
6	1	1	0	0

用一个简单的例子说明。表 1 是顾客购买记录的数据库 D，包含 6 个事务。项集 I={网球拍,网球,运动鞋,羽毛球}。考虑关联规则（频繁二项集）：网球拍与网球，事务 1,2,3,4,6 包含网球拍，事务 1,2,6 同时包含网球拍和网球，支持度（X^Y）/D=0.5，置信度（X^Y）/X=0.6。若给定最小支持度 α=0.5，最小置信度 β=0.6，认为购买网球拍和购买网球之间存在关联。

要求：

①设计合理的数据结构，编程实现算法；

②给定训练数据集，设计合理的文件格式，保存于外存之中；

① 设计 apriori 算法，保存关联规则在外存中。

48 聚类分析的初步实践

（1）问题描述

分类学是人类认识世界的基础科学。聚类分析和判别分析是研究事物分类的基本方法，广泛地应用于自然科学、社会科学、工农业生产的各个领域。聚类分析的基本思想是根据事物本身的特性研究个体分类的方法，原则是同一类中的个体有较大的相似性，不同类中的个体差异很大。

在分类之前首先需要定义分类的根据，根据不同分类的结果也不同，如要想把中国的县分成若干类，就有很多种分类法：可以按照自然条件来分，比如考虑降水、土地、日照、湿度等各方面；也可以考虑收入、教育水准、医疗条件、基础设施等指标。

下面给出一个小的实例，如果想要对 100 个学生进行分类，如果仅仅知道他

们的数学成绩，则只好按照数学成绩来分类；这些成绩在直线上形成 100 个点。这样就可以把接近的点放到一类。如果还知道他们的物理成绩，这样数学和物理成绩就形成二维平面上的 100 个点，

也可以按照距离远近来分类。三维或者更高维的情况也是类似；只不过三维以上的图形无法直观地画出来而已。因此需要定义数据之间的距离概念，从而可以度量数据之间的远近程度，作为聚类的基础。

在定义数据之间距离概念时，此时需要明确两个概念：一个是点和点之间的距离，一个是类和类之间的距离。点间距离有很多定义方式。最简单的是歐氏距离，还有其他的距离距离。当然还有一些和距离相反但起同样作用的概念，比如相似性等，两点越相似度越大，就相当于距离越短。由一个点组成的类是最基本的类；如果每一类都由一个点组成，那么点间的距离就是类间距离。但是如果某一类包含不止一个点，那么就要确定类间距离，类间距离是基于点间距离定义的：比如两类之间最近点之间的距离可以作为这两类之间的距离，也可以用两类中最远点之间的距离作为这两类之间的距离；当然也可以用各类的中心之间的距离来作为类间距离。在计算时，各种点间距离和类间距离的选择是通过统计软件的选项实现的。

不同的选择的结果会不同，但一般不会差太多。

常见的定义点和点之间距离方式，即向量 x=（x1,…,xp）与 y=（y1,…,yp）之间的距离或相似

系数为：

:为：|

①欧氏距离(Euclidean)：$\sqrt{\sum_i (x_i - y_i)^2}$

②绝对距离(Block)：$\Sigma_i|x_i-y_i|$

③Chebychev 距离：$Max_i|x_i-y_i|$

④夹角余弦(相似度量)：$C_{xy}(1) = \cos\theta_{xy} = \dfrac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2}}$

⑤Pearson correlation(相似度量)：$C_{xy}(2) = r_{xy} = \dfrac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$

常见的类 G_p 与类 G_q 之间的距离 D_{pq}($d(x_i,x_j)$表示点 $x_i \in G_p$ 和 $x_j \in G_q$ 之间的距离)。

①最短距离法: $D_{pq} = \min d(x_i, x_j)$

②最长距离法: $D_{pq} = \max d(x_i, x_j)$

③类平均法: $D_{pq} = \frac{1}{n_1 n_2} \sum_{x_i \in G_p} \sum_{x_j \in G_q} d(x_i, x_j)$

④重心法: $D_{pq} = \min d(\overline{x}_p, \overline{x}_q)$

有了上面的点间距离和类间距离的概念，就可以介绍聚类的方法了。此处介绍两种常见的聚类算法：

一种是事先要确定分多少类的 k-均值聚类算法（k-meanscluster）。假定你说分 3 类，这个方法还进一步要求你事先确定 3 个点为“聚类种子”；也就是说，把这 3 个点作为三类中每一类的基石。然后，根据和这三个点的距离远近，把所有点分成三类。再把这三类的中心（均值）作为新的基石或种子（原来的“种子”就没用了），重新按照距离分类。如此叠代下去，直到达到停止叠代的要求（比如，各类最后变化不大了，或者叠代次数太多了）。显然，前面的聚类种子的选择并不必太认真，它们很可能最后还会分到同一类中呢。下面是一个数据

实例：

49 蚁群算法在旅行商问题中的应用

（1）问题描述

蚁群算法是在对真实蚂蚁的观测基础上提出的，单个蚂蚁是不具智能的，但生活在一起的蚁群却总是能够在蚁穴和食物间找到一条几乎是最短的路径。这就要靠蚁群的智能，蚁群算法就基于这一智能。下面的图例给出蚁群智能的基本思想。

蚂蚁会在自己走过的路径上留下生化信息素，同时它也会选择地面上信息素较多的路径行走。对于下图的第二种情况，因为路上有了障碍物，所以蚂蚁需要绕过障碍物，该咋么样绕过障碍物，由于没有信息素作为指导，所以起先只能是随机的选择从左或从右走（第三个图所示）；但是随着行走次数的增加，较短的路径上留下的信息素就会较多，就有更多的蚂蚁行走，信息素进一步增多，最终较短的路就成为了选择的路（第四个图）。这就是蚁群智能的基本原理。

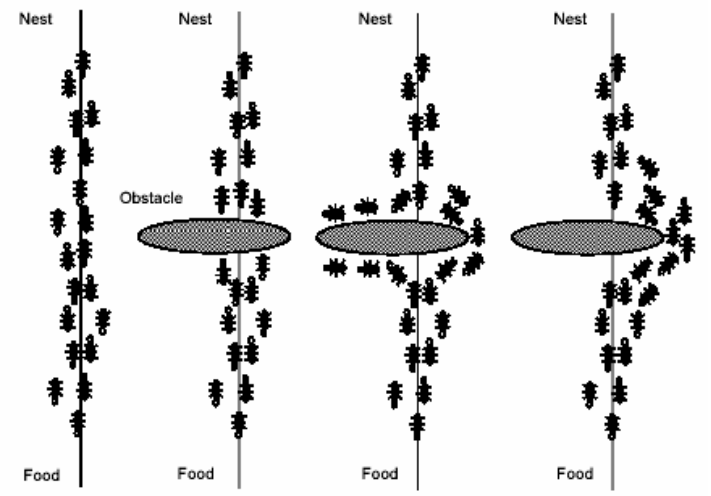


图 6-21 蚁群算法的基本思想

蚁群算法就是应用蚁群智能实现的算法，可以用它来实现在解空间上进行最优解的搜索，很多计算机问题可以转化为搜索最优解的问题。如旅行商问题（求解遍历全部城市的最短路经），就可用蚁群算法求解，在旅行商问题中，蚁穴到食物就对应遍历全部城市，蚂蚁就对应旅行商。

（2）课程设计目的

了解蚁群算法的思想，并能应用蚁群算法求解具体问题。

（3）基本要求

①应用蚁群算法求解 TSP 问题。

②TSP 中的城市数量不少于 30 个，组成完全图，边上的权值自定。

③蚂蚁数量可配置，迭代次数可配置。

④给出较全面的实验结果：结果路经及长度；蚁群算法执行时间；不同参数值（蚂蚁数量，迭代次数）的影响等。

⑤迭代过程需要用图形界面表示。

50 用动态规划方法计算编辑距离并完成自动编辑

（1）问题描述

对于任意两个字符串 s1，s2 的差别，可以通过其编辑距离来度量。编辑距离就是指让 s1 变成 s2 或将 s2 变成 s1 所需要编辑操作的最小次数。此处定义 3 个基本编辑操作：把某个字符 ch1 变成 ch2；删除某个字符；插入某个字符。如对于 s1=12433 和 s2=1233，则可以通过在 s1 中间删除 4 得到 1233 来与 s2 一致，

所以 d（s1，s2）=1（进行了一次删除操作），也即 s1，s2 的编辑距离为 1。可以依靠编辑距离的性质来帮助完成编辑距离的计算。当计算两个字符串 s1+ch1,s2+ch2 的编辑距离有这样的性质:①d（s1,“ ”）=d（“ ”,s1）=|s1|; d（“ch1”,” ch2”）=ch1==ch2？0:1。②d（s1+ch1，s2+ch2）=min（d（s1,s2）+ch1==ch2？0:1，d（s1+ch1,s2）+1，d（s1,s2+ch2）+1）。

具有第二个性质的原因是在 s1+ch1 和 s2+ch2 相互转化时可能的操作有：①把 ch1 变成 ch2，此时 d（s1+ch1，s2+ch2）=d（s1,s2）+ch1==ch2？0:1。②s1+ch1 删除 ch1 和 s2+ch2 转化，此时 d（s1+ch1，s2+ch2）=（1+d（s1，s2+ch2））。③s2+ch2 删除 ch2 和 s1+ch1 转化，此时 d（s1+ch1，s2+ch2）=（1+d（s1+ch1，s2））。④s2+ch2 后添加 ch1 和 s1+ch1 转化，实际上等同于 s2+ch2 和 s1 转化，此时 d（s1+ch1，s2+ch2）=（1+d（s1，s2+ch2））。⑤s1+ch1 后添加 ch2 和 s2+ch2 转化，实际上等同于 s1+ch1 和 s2 转化，此时 d（s1+ch1，s2+ch2）=（1+d（s2，s1+ch1））。有了这样的性质，可以发现在计算 d（m，n）时用到了 d（m-1，n），d（m-1，n-1），d（m，n-1）等，也就是说明该问题具有重叠子问题结构（原问题的递归算法反复的求解同样的子问题，而不产生新的子问题），再由性质②表明的最优子结构（一个问题的最优解包含了子问题的最优解），符合动态规划算法基本要素。因此可以使用动态规划算法来求解该问题。

由此可以得到这样的动态规划算法：设定数组 M[|s1|，|s2|]保存子问题结果，其中 M[i,j]表示子串 s1（0->i）与 s2（0->j）的编辑距离。产生字符之间的编辑距离，E[|s1|,|s2|]，其中 E[i,j]=s[i]=s[j]?0:1。初始化 M 矩阵，M[0,0]=0；M[s1i，0]=|s1i|；M[0,s2j]=|s2j|。

根据性质②计算出矩阵 M，得到编辑距离：M[i，j]=min（m[i-1,j-1]+E[i,j]，m[i,j-1]+1，m[i-1,j]）。

另外，编辑距离也会随着编辑操作而有所变化，如也可以这样来定义编辑：给定两个字符串 x[1..m]和 y[1..n]，编辑的目标是找到一系列的编辑操应用于 x 使它变为 y。用一个中间变量 z 来保存中间结果，算法开始的时候 z 是一个空字符串，算法结束的时候对所有 j=1,2,...,n 有 zj=yj。用变量 i 标记当前正被处理的字符在 x 中的下标，用 j 标记 z 的当前下标，我们的编辑操作就是作用于 i,j 和 z 上面的。开始时有 i=j=1，要求是在转换过程中一一检查 x 中的每一个字符，也

就是说在转换结束的时候必须有 i=m+1。

现定义六种编辑操作：①拷贝：把一个字符从 x 拷贝到 z，即令 zj=xi，之后令 i=i+1,j=j+1。这个操作检查了 xi。②替换：将 x 中的一个字符替换为另外一个，即令 zj=c，之后令 i=i+1,j=j+1。这个操作检查了 xi。③删除：从 x 中删除一个字符，即令 i=i+1，j 不变。这个操作检查了 xi。④插入：向 z 中插入一个字符，即令 zj=c，之后令 j=j+1，i 不变。这个操作没有检查 x 中的任何字符。⑤换位：把 x 中将要处理的下两个字符拷贝到 x 中，但是要颠倒顺序，即令 zj=x[i+1],z[j+1]=xi 之后令 i=i+2,j=j+2。这个操作检查了 xi 和 x[i+1]。⑥截断：删掉 x 中剩下的全部字符，即令 i=m+1。这个操作检查了 x 中当前尚未被检查到的全部字符，这个操作只能作为最后一个操作出现在编辑操作序列中。

例如将源串：algorithm 转换成目标串 altruistic 的一种方法是采用下面的操作序列：

操作	目标串	源串
copy a	a	lgorithm
copy l	al	gorithm
replace g by t	alt	orithm
delete o	alt	rithm
copy r	altr	ithm
insert u	altru	ithm
insert i	altrui	ithm
insert s	altruis	ithm
twiddle it into ti	altruisti	hm
insert c	altruistic	hm
kill hm	altruistic	

上述每一个变换操作都有相应的代价。假定所有编辑操作的代价都是常数并且是已知的，且要求拷贝和替换操作的代价小于等效的删除、插入操作的代价的和。定义给定的编辑操作序列的代价是其中每一个编辑操作的代价的总和，对上面的例子来说，编辑序列的代价是（3*cost（copy））+cost（replace）+cost（delete）+（4*cost（insert））+cost（twiddle）+cost（kill）。显然，需要找到一个从源串到

目标串的具有最小代价的编辑操作序列。

（2）课程设计目的

掌握动态规划方法，能够编程实现。对编辑距离建立一定的认识，并能将某些问题转化为编辑距离问题。

（3）基本要求

①实现问题描述中第一种编辑操作（三个操作）编辑距离计算的动态规划算法，并用测试用例验证算法。

②将问题描述中第二种编辑操作中给出的六个操作都实现，并将其作为串 ADT 的基本操作实现一个串 ADT。

③针对问题描述中第二种编辑操作，设计动态规划算法实现其编辑距离的计算，即找到一个代价最小的编辑操作序列，其中各个操作的代价根据实际情况自己设定。

④应用③获得的最小代价编辑操作序列实现从源串到目标串的自动编辑。

⑤针对问题描述中第二种编辑操作，自行设计可获得自动编辑操作序列（代价不用最小）的一个算法，将其结果编辑操作序列和③获得的最小代价编辑操作序列进行代价上的对比。

⑥（选做）编辑距离问题是 DNA 序列对齐问题的泛化。可以有很多方式对齐 DNA 序列，以衡量它们的相似程度。有一种对齐方式是，可以在两个 DNA 序列的任何位置（包括两头）插入空格，但是要求通过这个过程得到的序列 x1 和 y1 长度相同，而且在两个序列的同一位置上不能都是空格。做到这一步之后我们为每一个位置分配一个“得分”，例如位置 j 按照如下规则得分：如果 x1[j]==y1[j]，1 分；如果 x1[i]!=y1[j]并且两者都不是空格，-1 分；如果 x1[j]或者 y1[j]是空格，-2 分。某种特定对齐结果的得分，是全部位置的得分的总和。例如对序列 x=GATCGGCAT 和 y=CAATGTGAATC，一个可能的对齐如下：

GATCGGCAT

CAATGTGAATC

-*++*+*+-++*

其中+对应 1 分，-对应-1 分，*对应-2 分。

这个对齐的总分是-1-2+2-2+1-2+1-1+2-2=-1-2+1-2=-4。

在上述编辑距离动态规划算法基础上，编写程序求出两个 DNA 序列的最优对齐，即得分最多的对齐。

（4）实现提示

需要查阅关于动态规划算法的有关资料。

51 树搜索之分支限界法

（1）问题描述

一个 8 数码问问题是将如图 6-69（a）所示的初始格局中的小格子中的数字经过若干步移动后（只能水平和垂直移动到一个空白处）形成如图 6-69（b）所示的初始格局。

(*a*) (*b*)

图6-69 8数码问题

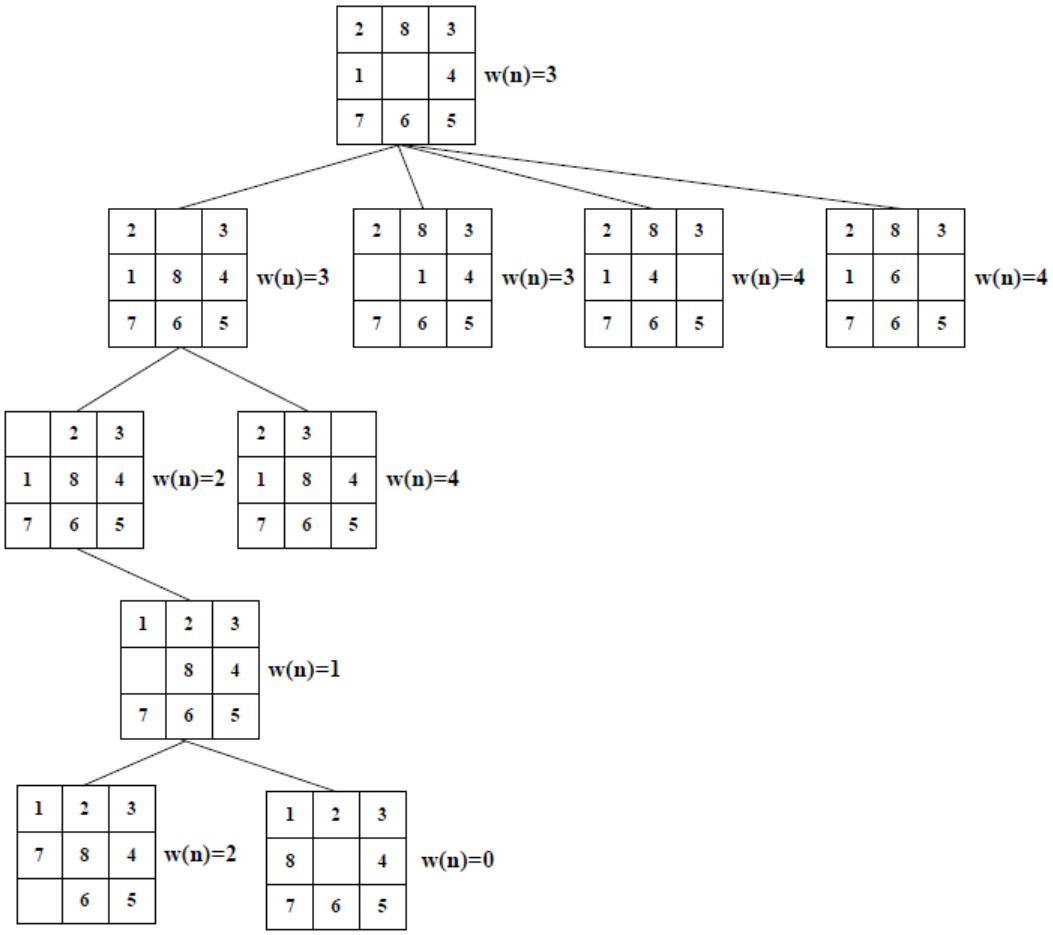


图6-70 用爬山法解决8数码问题

可用树搜索来解决 8 数码问题，如采用爬山法来解决该问题，爬山法的基本思想是根据一个评价准则贪心的选择一个节点进行所有可能的展开（枚举下一部的所有可能情况），这样处理直到找到解（到达目标格局）。对于上面的 8 数码问题，定义评价函数 w（n）=错误放置的节点个数，显然贪心准则就是选取 w（n）最小的节点展开。用爬山法解决上面的 8 数码问题的过程如图 6-70 所示。

有一点需要注意的是上面的爬山法给出的是一个解，并不一定是最优解，即移动次数最少的解，在很多情况下需要得到最优解，那么就需要将所有的节点都进行展开，即穷举搜索，此时爬山法没有任何帮助。接下来给出一种分支限界策略，该策略可以较好的解决这些优化问题，其基本思想是通过优化解的界限来修剪可行解从而减少搜索。如对于图 6-71 所示的最短路径搜索问题，可以用树搜索办法解决。首先用爬山法对该问题建立一个可行解，结果如图 6-72（a）-图 6-72（c）所示。可以求出该可行解对应的路径长度为 5，所以在展开其他节点时，

如果发现了其他路径已经大于等于 5 时就不用进一步展开了，如图 6-72（d）所示，只有小于该长度的节点采用必要进一步展开，即图 6-72（d）中的阴影节点，从而修剪了许多分支，提高了算法效率，这就是分支限界法的基本思想。

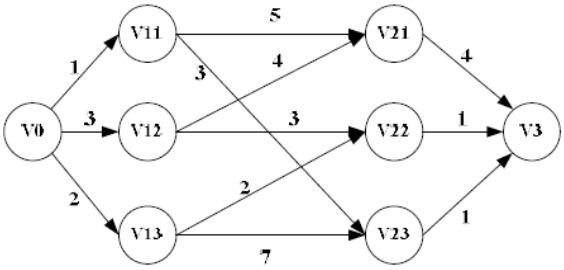


图 6-71 最短路径搜索问题

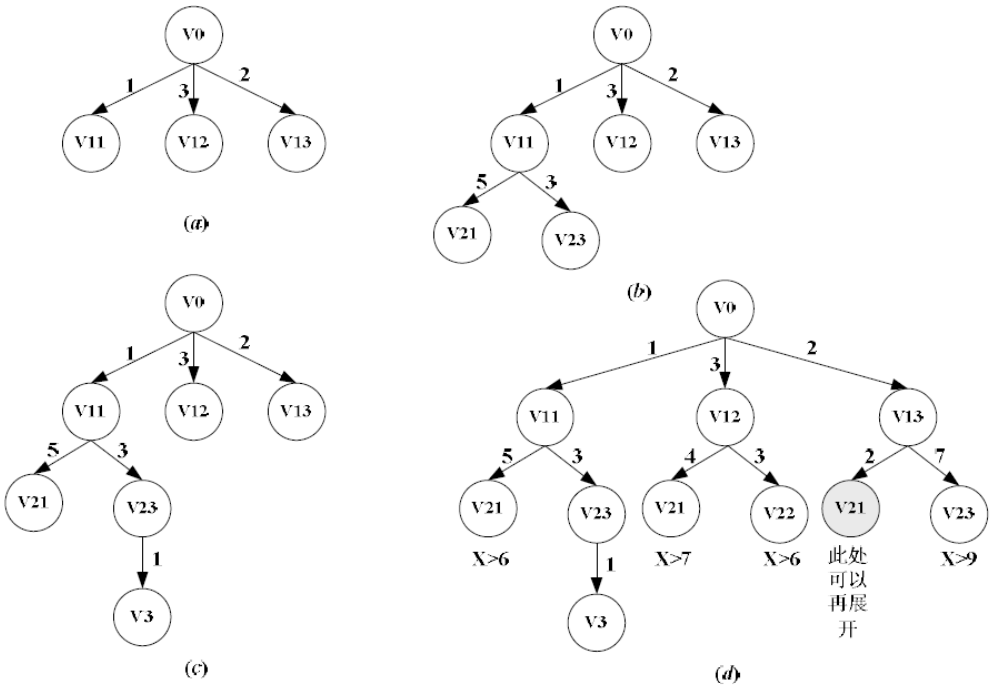


图 6-72 分支界限方法实例

可以用分支限界法解决旅行商问题（travelingsalepersonproblem 或 TSP），旅行商问题是一个在 n 个点的完全有向（或无向）加权图上寻找一个包含所有顶点的代价最小的回路，该问题已被证明是一个 NPC 问题，穷举搜索的时间复杂性是指数函数，可以用分支限界法来避免穷举搜索，减少搜索次数。用分支限界法解决 TSP 问题的方法由两个部分组成。

①用一种方法划分解空间（从而形成一棵搜索树）。

②对每一类解预测其下界（需建立一种预测方法），用爬山法得到最优解的上界（即存在一个代价为次上界的解），如果某个分支的下界大于该上界，则终

止这个分支。以下面的实例来说明这一过程，设一旅行商问题的代价矩阵（初始情况就是邻接矩阵）为。

	1	2	3	4	5	6	7
1	∞	3	93	13	33	9	57
2	4	∞	77	42	21	16	34
3	45	17	∞	36	16	28	25
4	39	90	80	∞	56	7	91
5	28	46	88	33	∞	25	57
6	3	88	18	46	92	∞	7
7	44	26	33	27	84	39	∞

①分支限界法需将解分成两组：一组是包含某条特定的弧的解，另一组是不包含该弧的解。

②按如下方法分析下界：首先注意到从代价矩阵中的任一行和任一列中减去一常数是不会改变最优解的。从该代价矩阵中的每一行分别减去 3，4，16，7，25，3，26。然后再从第 3，4 和 7 列中分别减去 7，1，4 使得代价矩阵中的每一行和每一列都至少包含一个 0。共减去的代价是 3+4+16+7+25+3+26+7+1+4=96，所以该旅行商问题解的代价的下界是 96。

而经过这样处理后的代价矩阵为：

	1	2	3	4	5	6	7
1	∞	0	83	9	30	9	50
2	0	∞	66	37	17	12	26
3	29	1	∞	19	0	12	5
4	32	83	66	∞	49	0	80
5	3	21	56	7	∞	0	28
6	0	85	8	42	89	∞	0
7	18	0	0	0	58	13	∞

③选取将解分成两组时选取的弧是 4-6，选取 4-6 有三个原因：一是该弧代价为 0，选取包含该弧的解的代价应该较小的代价（贪心法）；另一个原因是当该弧代价为 0 时，如果解不包含该弧时，必定包含一条从 4 出发的弧和一条进入

6 的弧，选取两条代价最小的这样的弧 4-1 和 5-6，其代价为 32 和 0，所以这样解的代价的下界应该为 96+32=128；第三个原因是所有的这样的权值为 0 的弧中，不引入这条弧引起的下界的增加中选 4-6 是最大的，如选弧 3-5 进行划分，则引入的下界的增加为 1+17=18。

④选取弧 4-6 后，由于选取了该弧，所以从代价矩阵中删除第 4 行和第 6 列（已经找到了一条边，其他边没有用了），同时弧 6-4 也没有必要再包含了，即设置 C[6，4]=　。此时代价矩阵变为：

	1	2	3	4	5	7
1	∞	0	83	9	30	50
2	0	∞	66	37	17	26
3	29	1	∞	19	0	5
5	3	21	56	7	∞	28
6	0	85	8	∞	89	0
7	18	0	0	0	58	∞

此时发现第 5 行不包含 0，所以对第 5 行减去 3，所以包含弧 4-6 的解的下届应该增 3。

⑤用这样的方法继续处理，直到得到一个解，如图 6-73 所示。可得出该解的代价，为 126，所所以所有的下届大于 126 的节点称为终止节点。

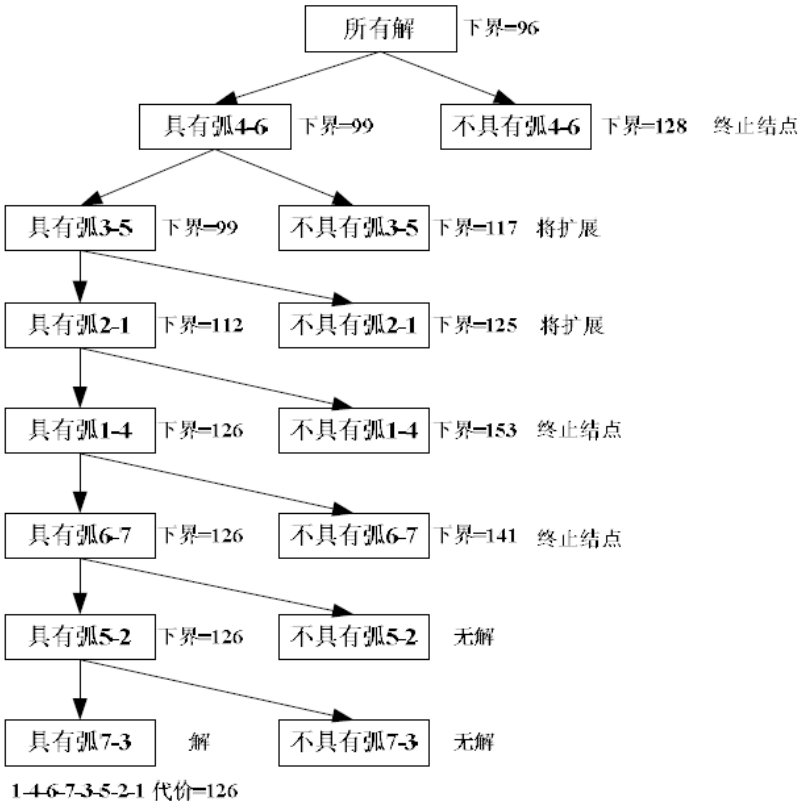


图 6-73 旅行商问题的分支限界解法过程

（2）课程设计目的

对树搜索建立一定的认识，通过编程实现分支限界方法掌握该方法，并能用该方法解决一些实际的优化问题。

（3）基本要求

①针对 8 数码问题应用爬山法和分支限界法分别解决，体会二者的区别。

②针对 TSP 问题，实现上述的分支限界法。

③产生一个 TSP 问题实例（多余 10 个节点），该实例可以从平面上随机产生 n 个点，权值就是两个点之间的笛卡尔距离，用分支限界法解决该问题，并输出其中间过程。

（4）实现提示

爬山的过程可用栈来实现。

52 剪枝搜索策略解决1-中心问题

（1）问题描述

1 中心问题（1-centerproblem）定义为：给定 n 个平面点的集合，问题要求

找到一个覆盖所有 n 个点的最小圆，如图 6-78 所示。实际上该问题的关键就是圆心的确定，因为一旦确定圆心后就可以找到离该圆心最远的点，从而可以确定半径，就可以画出该圆了。用穷举法解决该问题费时费力，所以可以用其他的优化算法来解决该问题。剪枝搜索策略（prune-and-search）世界解决该问题的一个不错的方法。下面介绍一下剪枝搜索策略的基本思想。

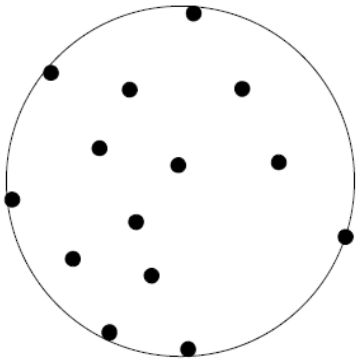


图 6-77 1-中心问题实例

剪枝搜索策略通常由几次迭代组成，每次迭代中都会剪除输入数据的一部分（如剪掉的部分占总体的比例为 f），当然必须要求这些输入数据不影响问题的最终解，然后递归调用同样的算法来处理剩余的数据，从而递归解决该问题。由于每次迭代会剪除输入数据，p 次迭代后输入数据的规模就会变得很小，可以在常数时间内解决。设每次迭代执行的剪枝时间为 O（nk），此时剪枝搜索算法的最坏时间复杂性为：

$$T(n) = T((1-f)n)+O(n^k) \le T((1-f)n) + cn^k \le T((1-f)^2n) + cn^k + c(1-f)^k n^k$$

$$\le c' + cn^k + c(1-f)^k n^k + c(1-f)^{2k} n^k + \ldots + c(1-f)^{pk} n^k$$

$$= c' + cn^k[\ 1 + (1-f)^k + (1-f)^{2k} + \ldots + (1-f)^{pk}\ ]$$

n 趋于无穷大时，T（n）=O（n^k）。这表明剪枝搜索算法的时间复杂度和每次迭代（即剪枝）的时间复杂度是一样的，而通常就是线性时间，即

T（n）=O（n），因此剪枝搜索算法是一个效率非常高的优化搜索算法。

现在来看如何应用剪枝搜索策略解决 1-中心问题，假设现在有如图 6-78 所示的点分布，并假定已知最优解圆心分布的区域（这是剪枝搜索的一个关键点）——图中的阴影区域，其中 L12 和 L34 分别是线段 p1p2 和线段 p3p4 的垂直平分线。因为 L12 和阴影区域没有相交，所以 p1 和 p2 两点中必有其一离最优解

圆心近，而另一点远，显然这一近的点可以被删掉，因为它不影响结果，在本图中就是 p1。

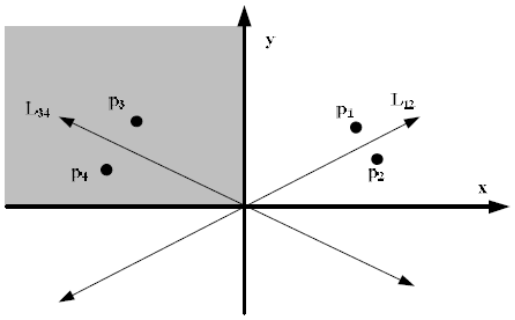


图 6-78 1-中心问题可能的一种点剪除的情况

由上面给出的简单实例可以看出，剪枝算法的关键在于知道最优解的圆心位于哪一部分，哪些点比其他点离圆心近，从而将这些点剪除。此处首先给出一个受约束的 1-中心问题算法，其中的约束是指圆心必须位于一条直线上，不是一般性，假定该直线为$y = y'$，该算法将被通用 1-中心问题解法调用。该受约束的 1-中心问题算法如下：

算法：解决受约束的 1-中心问题算法

输入：n 个点和一条直线$y = y'$

输出：在直线$y = y'$上的满足最小覆盖的圆心

第 1 步：如果 n 小于等于 2，用几何方法直接解决；

第 2 步：构造偶数个不相交（交集为空）点对（p1,p2），（p3,p4），…，（pn-1,pn）。如果有奇数个点，则令最后点对（pn,p1）。

第 3 步：对每个点对（pi,pi+1），求出在直线 y=y 上的点 xi,i+1，使得 d（pi,xi,i+1）=d（pi+1,xi,i+1）。

第 4 步：求出这 $\lceil n/2 \rfloor$个 xi,i+1 的中点，记为 xm。

第 5 步：对所有的 i，计算 pi 和 xm 的距离。另 pj 为离 xm 的最远点，xj 表示 pj 在直线 y=y 上的投影。如果 xj 在 xm 的左方，则最优解 x*必位于 xm 的左方；反之，如果 xj 在 xm 的右方，则最优解 x*必位于 xm 的右方。

第 6 步：如果 x*<xm（图 6-79 就是这种情况）

对每个 xi,i+1>xm，如果 pi 比 pi+1 离 xm 近，那么剪去点 pi，否则剪去点

pi+1。

如果 x*>xm（图 6-79 就是这种情况）

对每个 xi,i+1<xm，如果 pi 比 pi+1 离 xm 近，那么剪去点 pi，否则剪去点 pi+1。

第 7 步：转向第 1 步。

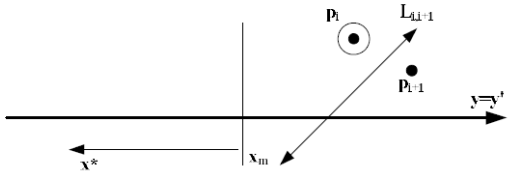
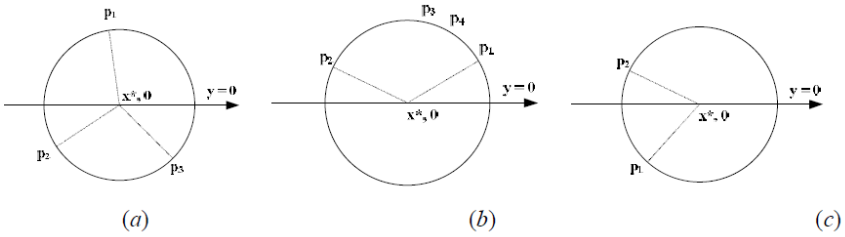


图 6-79 有约束 1-中心问题的点剪除

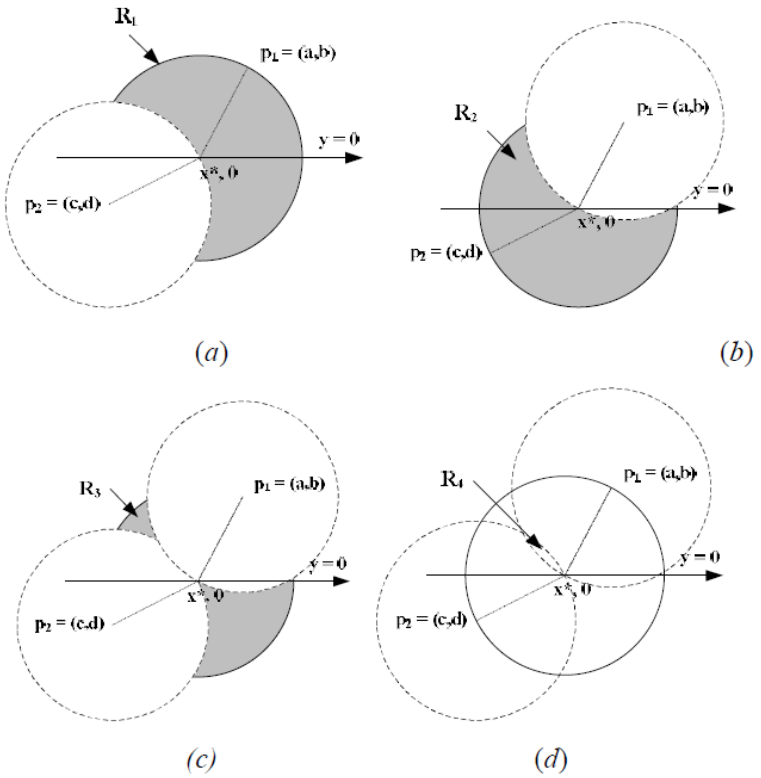
在此基础上可以考虑这样一个问题：如果已经求得在直线 y=0 上的有约束 1-中心问题的解 x*，现假定（xs，ys）为包括所有点的最优解圆心（无约束），则可以根据 x*的信息求出一些关于 xs 和 ys 的信息，即可以确定 ys>0，ys<0 或 ys=0。用同样的方法可以确定 xs>0，xs<0 或 xs=0。确定的基本方法是：令 I 表示距离点（x*，0）最远点组成的集合，则此时有两种情况。情况 1：I 包含一个点，记为 p。此时 p 的 x 坐标必定等于 x*，否则 x*可以沿着 y=0 向 p 移动，此时最优解就不是 x*，而是移动后的点。所以此时的 ys 必定在靠近 p 的一侧，这是由于 p 是唯一的最远点，所以必有 ys 与 p 的 y 坐标正负号相同。情况 2：I 中包含多个点，显然这些点一定在同一个圆上，且该圆的圆心为（x*，0），所以可以找到覆盖 I 中所有点的最小弧，令该弧的两个端点为 p1 和 p2，如果该弧的度数大于等于 180 度，如图 6-80 中的（a），则 ys=0。现说明原因，由于包含某点集的最小圆一定是由这些点中的两个或三个确定的，显然，两个点确定最小圆，这两个点的连线就是该圆的直径，三个点确定最小圆，当且仅当这三个点形成的是锐角三角形，显然这正是 p1p2 弧大于等于 180 度的结果，也就是说此时的最小圆（圆心为（x*，0））已经是最优的，进而可以推断 ys=0。如果 p1p2 弧小于 180 度，如图 6-80 中的（b），此时必定有 p1 和 p2 的 x 坐标必定在 x*的两侧，否则如图 6-80（c）所示，则 x*可向 p1 和 p2 方向移动，约束 1-中心问题的最优解就不是 x*了。因此可以假定 p1=（a,b），p2=（c,d），不是一般性假定 a>x*，b>0，c<x*，d<0。可以画出如图 6-81 所示的四种情况，显然最优解（xs，ys）一定在区域 R4 中，因为在区域 R1、R2、R3 中都有最优解圆心和 p1 或 p2 的距

离大于约束最优解的半径 r，这是不应该的。所以 ys 与（b+d）/2=（y1+y2）/2 同号。



(a) (b) (c)

图 6-80 I 中包含多于 1 点的情况



(a) (b)

(c) (d)

图 6-81 度数小于 180 度时 y_s 的方向

因此就可以给出下面的程序来确定 ys 的方向。

程序：确定 ys 的方向

输入：点集合 S，直线 y=y*，（x*,y*）是对于 S，在 y=y*上的有约束的 1-中心问题解。

输出：ys>y*，ys<y*，或 ys=y*。其中（xs,ys）为 S 的 1-中心问题解。

第 1 步：找到距离（x*,y*）最远点的集合 I。

第 2 步：情况 1。I 中只包含 1 个点 p=（xp,yp）。

如果 yp>y*，输出“ys>y*”并退出；

如果 yp<y*，输出“ys<y*”并退出。

情况 2。I 中包含多于 1 个点。在 I 中找到形成覆盖 I 中所有点的最小弧的两个端点 p1=（x1,y1）和 p2=（x2,y2）。

情况 2.1。p1 和 p2 形成的弧大于等于 180 度，输入 ys=y*并退出。

情况 2.2。p1 和 p2 形成的弧小于 180 度，令 yc=（y1+y2）/2，

如果 yc>y*，输出“ys>y*”并退出；

如果 yc<y*，输出“ys<y*”并退出。

现在可以给出在上面两个算法的基础上如何求解无约束的 1-中心问题：对于任意两个点对（p1,p2）和（p3,p4），分别画出线段 p1p2 和 p3p4 的垂直平分线 L12 和 L34，令 12 和 L34 的交点为 p，旋转 x 轴使 L34 的斜率为负，使 L12 的斜率为正，将坐标原点移动至 p，形成如图 6-78 所示的情况，此时应用约束 1-中心问题的求解算法得到约束为 y=0 的解，然后调用确定 ys 的方向来确定 ys 的方向，对于图 6-78 有 ys 向上。在应用约束 1-中心问题的求解算法得到约束为 x=0 的解，然后调用确定 xs 的方向来确定 xs 的方向，对于图 6-78 有 xs 向左。所以有 1-中心问题的解在图 6-78 中的阴影上，可以剪除点 p1，缩小输入。

总结上述思想可以得到解决 1-中心问题的剪枝搜索算法，算法如下：

算法：1-中心问题的剪枝搜索算法

输入：n 个点的集合 S={p1,p2,…,pn}。

输出：包括 S 中所有点的最小圆。

第 1 步：如果 S 包含不多于 16 个点，用穷举方法解决。

第 2 步：形成不相交（集合不相交）点对，（p1,p2），（p3,p4），…，（pn-1,pn）。对每个点对（pi,pi+1）找到线段 pipi+1 的垂直平分线，记为 Li/2，计算其斜率，将 Lk 的斜率记为 sk。

第 3 步：计算 sk 的中点 sm，旋转坐标系使 x 坐标轴与 y=smx 重合，令 Lk 中斜率为正（负）组成的集合为 I+（I-）。

第 4 步：构建不相交（集合不相交）的直线对（Li+,Li-），Li+ I+，Li- I-，找到每对直线的交点，记为（ai,bi）。

第 5 步：找出 bi 的中点，记为 y*。对 S 应用有约束的 1-中心问题子程序求得其在 y=y*上的圆心，令其为（x ,y*）。

第 6 步：以 S 和（x ,y*）作为参数，应用求 ys 方向程序。

如果 ys=y*，输出（x　,y*）为最优解，退出。

否则输出 ys>y*或 ys<y*。

第 7 步：找出 ai 的中点，记为 x*。对 S 应用有约束的 1-中心问题子程序求得其在 x=x*上的圆心，令其为（x*,y　）。

第 8 步：以 S 和（x*,y　）作为参数，应用求 xs 方向程序。

如果 xs=x*，输出（x*,y　）为最优解，退出。

否则输出 xs>x*或 xs<x*。

第 9 步：

情况 1。xs>x*且 ys>y*

找到所有 ai<x*且 bi<y*的（ai,bi），令（ai,bi）为（Li+,Li-）的交点，Li-为 pj 和 pk 的垂直平分线。如果 pj（pk）比 pk（pj）离（x*,y*）更近，则剪除 pj（pk）。

情况 2。xs<x*且 ys>y*

找到所有 ai>x*且 bi<y*的（ai,bi），令（ai,bi）为（Li+,Li-）的交点，Li+为 pj 和 pk 的垂直平分线。如果 pj（pk）比 pk（pj）离（x*,y*）更近，则剪除 pj（pk）。

情况 3。xs<x*且 ys<y*

找到所有 ai>x*且 bi>y*的（ai,bi），令（ai,bi）为（Li+,Li-）的交点，Li-为 pj 和 pk 的垂直平分线。如果 pj（pk）比 pk（pj）离（x*,y*）更近，则剪除 pj（pk）。

情况 4。xs>x*且 ys<y*

找到所有 ai>x*且 bi<y*的（ai,bi），令（ai,bi）为（Li+,Li-）的交点，Li+为 pj 和 pk 的垂直平分线。如果 pj（pk）比 pk（pj）离（x*,y*）更近，则剪除 pj（pk）。

第 10 步：设 S 为剩余点，转向步骤 1。

（2）课程设计目的

对剪枝搜索算法建立一定的认识，编程实现上述算法，对计算几何和 1-中心问题进行实践。

（3）基本要求

①编程实现上述 1-中心问题剪枝搜索算法。

②点在平面上随机放置，最好给出解的图形表示，以方便验证结果的正确性。

③自己设计一个针对 1-中心问题的穷举算法。

④在一台机器上对上面的两个算法进行实验对比，点数分别为 10，30，50，100，150，200，250，300。

⑤分析 1-中心问题剪枝搜索算法的时间复杂度和 1-中心问题的穷举算法的时间复杂度。

（4）实现提示

无。

53 Chord网络的模拟

（1）问题描述

Chord 是一种非常经典的 P2P 结构化网络，可以在 Chord 上进行分布式 P2P 文件共享等应用。Chord 网络的基本结构如图 6-59 所示，它是以分布式散列表为基础构建的一种逻辑网络。

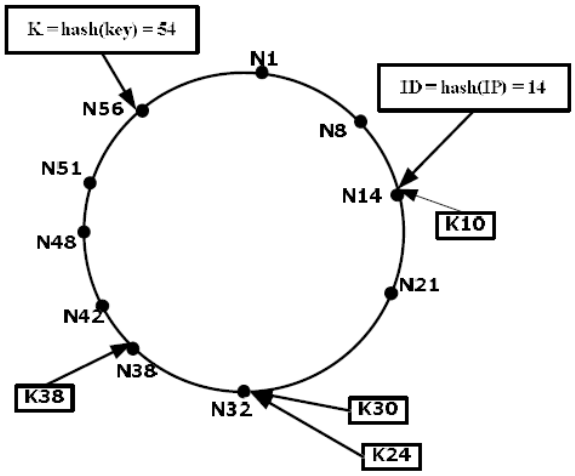


图 6-59 Chord 基本结构

分布式散列表（DHT）实际上是一个由大量结点分布式的共同维护的巨大散列表。散列表被分割成不连续的块，每个结点被分配给一个属于自己的散列块（一个散列值范围），并成为这个散列块的管理者，负责存储散列结果位于该散列块内的信息。DHT 中使用的散列函数通常是加密散列函数（如 MD5，SHA-1 等），通过这些函数，一个对象的名字（如节点的 ID 或其 IP 地址）或关键词（如文件名）被映射为 128 位或 160 位的散列，如 SHA-1（“202.38.64.1”）

=24b92cb1d2b81a47472a93d06af3d85a42e463ea。一个采用 DHT 的系统内，所有计算结点、数据对象等被映射到一个空间内。对于图 6-59 中的 Chord 结构，每个计算节点根据其 IP 可以计算出其 ID，如图中的 N14。

每个文件根据其关键词可以计算出每个信息的 ID，就是图中的 K，如图中的 K=hash（key）=54，这些 K 被放在 Chord 环中机器节点 ID 大于 K 且最近的 K 的节点，如图中将 K=54 的信息放在 ID=56 的节点 N56 上，将 K=30 和 K=24 的信息放在 ID=32 的节点 N32 上。

Chord 结构主要支持如下操作：①Insert（key,V），即将关键词 key 对应的信息 V 对存放到节点 ID 大于 K=hash（key）且离 K 最近的节点上，这样的 ID 可用 Successor（K）表示，其中 Successor（K）是从 K 开始顺时针方向距离 K 最近的节点。②Lookup（K），根据 K 查询相应的 V，主要表现为根据 Chord 中节点的连接（上图中的节点间连线）路由到存放 K 的节点上，即节点 ID=Successor（K）的节点，在该节点上可以找到 K 对应的 V。③Update（K,new_V）：根据 K 更新相应的 V。④Join（NID）：加入一个新节点，NID 是节点的标识，如节点的 IP 地址，在 Chord 中加入一个节点需要建立相应的连接，需要移动信息数据，如上图中新加入一个节点 N26，则需要将 K=24 移动到该节点。⑤Leave（）：某个节点主动离开 Chord，在 Chord 中推出一个节点需要修改相应的连接，也需要移动某些信息数据，如上图中退出一个节点 N14，则需要将 K=10 移动到节点 N21。

Chord 结构的一个非常重要的特点是如果每个节点仅维护其后继节点 ID、IP 地址等信息，则查询消息通过后继节点指针在圆环上传递，直到查询消息中包含的 K 落在某节点 ID 和它的后继节点 ID 之间，这样的查询速度太慢 O（N），N 为网络中节点数，如图 6-60（a）所示。因此在基本 Chord 上，引入了一些能加快查询的 finger 表，finger 表的结构如图 6-60（b）表示，节点为 ID 的 finger 表中存放的是所有的（ID+2i）modN　ID 的 i 从 1 开始且逐个增 1 的 ID=Successor（ID+2i）的那些节点，每个 i 对应了 finger 表中的 1 项。有了 finger 表后，可以加速查询过程，对于路由中的任何一个节点 ID，该节点选择的路由下一跳是 finger 表中满足（ID+2i）modN 小于等于 K 且最接近 K 的那个 i 对应的表项，从这个表项中可以找到查询的下一跳，如图 6-60（c）所示。

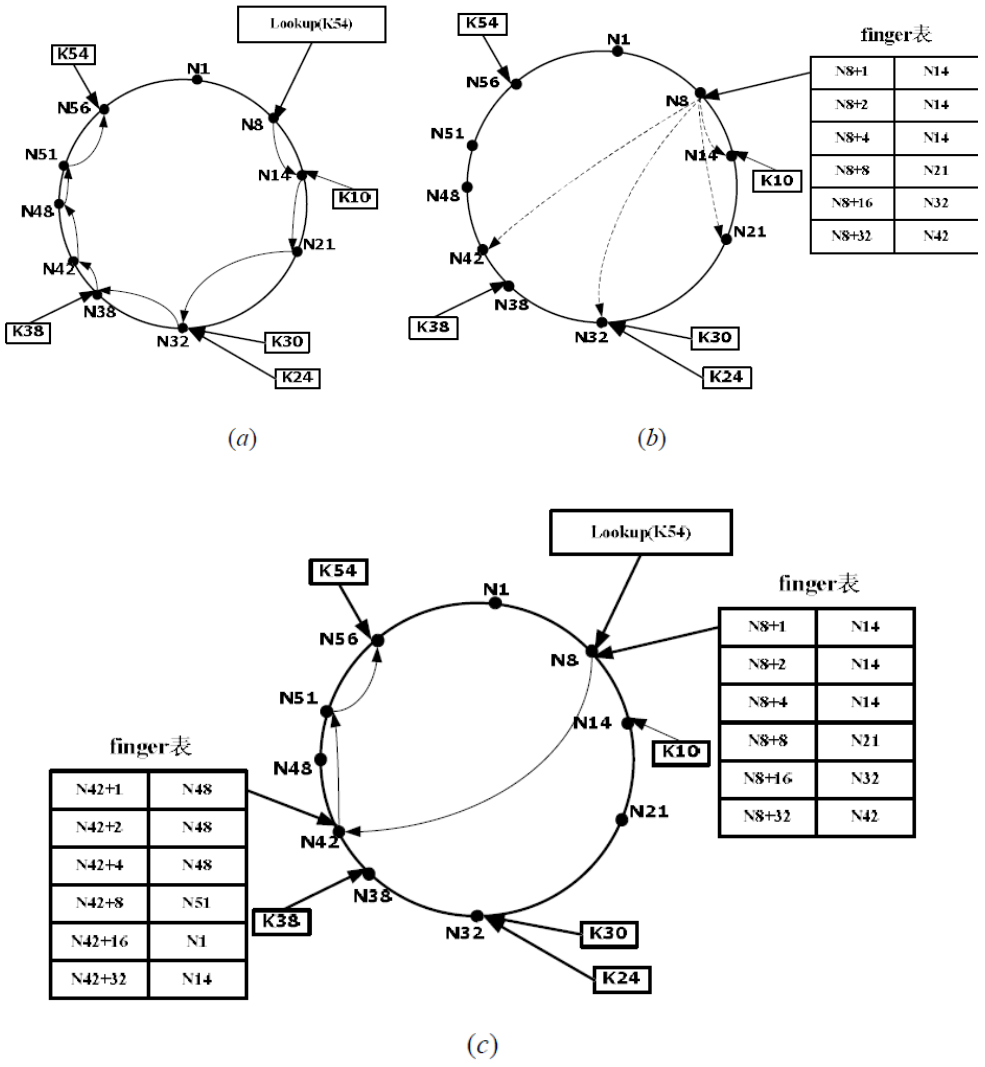


图 6-60 带 finger 表的 Chord 结构

仔细分析可以发现，引入 finger 表后，查询 K 的路由跳数为 O（logN），相比 O（N）而言有很大的提高。

（2）课程设计目的

建立对 Chord 结构的认识，能用数据结构知识完成 Chord 结构的模拟。

（3）基本要求

①用数据结构知识设计和实现 Chord 网络结构。

②实现 Chord 结构上的 Insert、Lookup、Update、Leave、Join 五个操作。

③构建一个 Chord 的单机模拟应用，其中的数据信息可以自己定义，如可以定义 key 为一个一个的英语单词，而其对应的数据为该单词的英文解释。

④应模拟出各个操作的结果，尤其是模拟出 Lookup 的路由过程，模拟过程应该是可视的，即可以看到节点的连接，路由一跳一跳的过程，鼓励使用 VC 实

现。

⑤用实验结果分析验证引入 finger 表后，查询 K 的路由跳数为 O（logN）的理论结果。

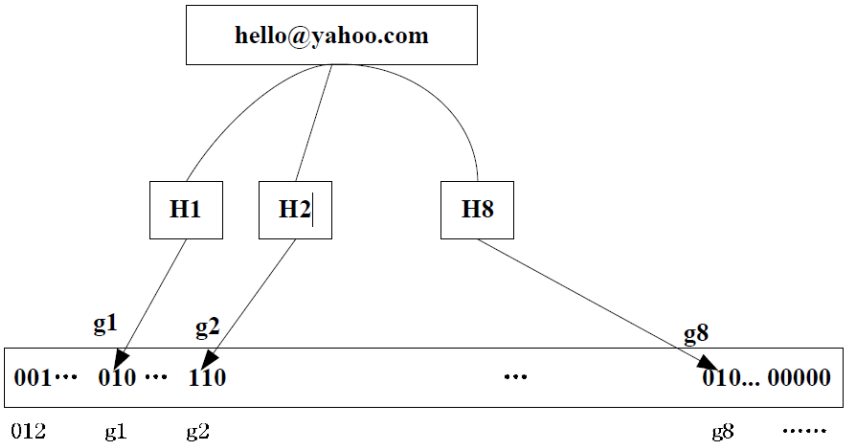
（4）实现提示

可以查阅 P2P 中关于 Chord 的资料，其中用到的散列函数可以直接使用现有的散列函数，如 SHA-1，可以直接下载使用散列函数的源代码。

54 布隆过滤器的实现和应用

（1）问题描述

布隆过滤器是由巴顿.布隆于一九七零年提出的。它实际上是一个很长的二进制向量和一系列随机映射函数。假定需要存储一亿个电子邮件地址，首先建立一个十六亿二进制（比特），即两亿字节的向量，然后将这十六亿个二进制全部设置为零。对于每一个电子邮件地址 X，可以用八个不同的散列函数（H1,H2,...,H8）产生八个从 1 到十六亿之间中的八个自然数 g1,g2,...,g8。然后将这八个自然数对应的八个位置的二进制全部设置为一。同样的方法对这一亿个 email 地址都进行处理后，一个针对这些 email 地址的布隆过滤器就建成了。如图所示：



现在看看如何用布隆过滤器来实现一个电子邮件地址过滤器，首先将那些放在黑名单上的电子邮件地址放在布隆过滤器中。当检测一个可疑的电子邮件地址 Y 是否在黑名单中，仍用相同的八个随机数产生器（H1,H2,...,H8）对这个邮件地址产生八个自然数 s1,s2,...,s8，这八个自然数对应的八个二进制位分别是 t1,t2,...,t8。如果 Y 在黑名单中，显然，t1,t2,..,t8 对应的八个二进制一定是一。这样在遇到任何在黑名单中的电子邮件地址，都能准确地发现。

布隆过滤器决不会漏掉任何一个在黑名单中的可疑地址。但是，它有一条不

足之处。也就是它有极小的可能将一个不在黑名单中的电子邮件地址判定为在黑名单中，因为有可能某个好的邮件地址正巧对应个八个都被设置成一的二进制位。但这种可能性很小，此处将它称为误识概率。在上面的例子中，误识概率在万分之一以下。

因此布隆过滤器的好处在于快速，省空间。但是有一定的误识别率。常见的补救办法是在建立一个小的白名单，存储那些可能别误判的邮件地址。

（2）课程设计目的

学习 BloomFilter 结构，能应用该结构解决一些实际问题。

（3）基本要求

①定义 BloomFilter 结构的 ADT，该 ADT 应支持在 BloomFilter 中加入一个新的数据，查询数据是否在此过滤器中，并完成该结构的设计和实现。

②应用 BloomFilter 结构拼写检查，许多人都对 Word 的拼写检查功能非常了解，当用户拼错一个单词的时候，Word 会自动将这个单词用红线标注出来。Word 的具体工作原理不得而知，但另一个拼写检查器 UNIXspell-checkers 这个软件中就用到了 BloomFilter。UNIXspell-checkers 将所有的字典单词存成 BloomFilter 数据结构，而后直接在 BloomFilter 上进行查询。本课程设计要求针对 C 语言设计和实现上述拼写检查器，即当写了一个正确的关键词，如 int 时，给该词标上颜色，如蓝色。

③针对上述 C 语言关键词拼写检查器进行分析，如错误分析，设计散列函数个数分析，运行时间复杂性、空间复杂性的分析。

④上述 C 语言关键词拼写检查器最好是在 VC++或 Java 等可视化开发环境下实现。

⑤上述 C 语言关键词拼写检查器最好能支持所有的 C++关键词。

（4）实现提示

BloomFilter 结构中的散列函数（包括散列函数的个数和散列函数的设计）是本题目中需要深入思考的一个环节。

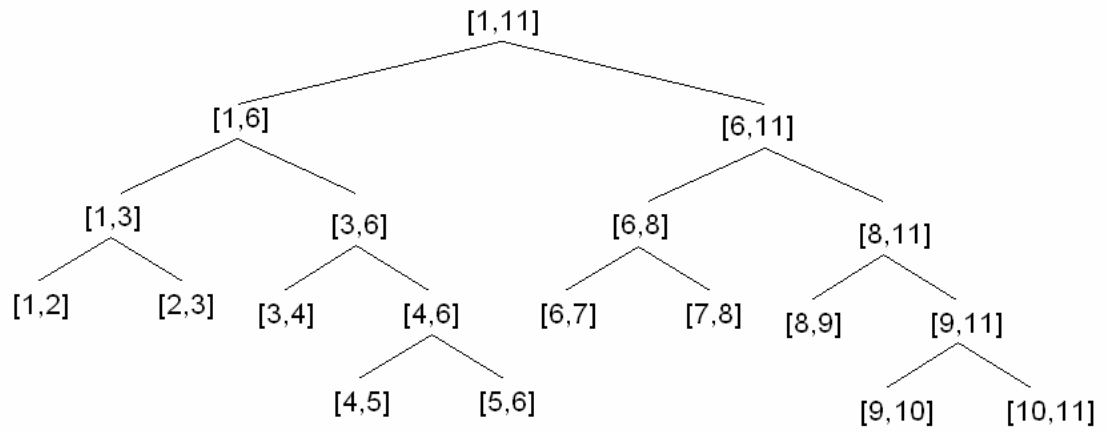
55 用线段树进行数据的动态维护

（1）问题描述

当处理图形的面积、周长等问题的时候，并不需要依赖很深的数学知识，但

要提高处理此类问题的效率却又十分困难。这就需要从根本上改变算法的基础——数据结构，不采用线性表结构，这里需要的就是一种特殊的数据结构——线段树。在此类问题中，需要经常处理可以映射在一个坐标轴上的一些固定线段，例如说映射在 X 轴上的线段。由于线段是可以互相覆盖的，有时需要动态地取线段的并，例如取得并区间的总长度，或者并区间的个数等等。一个线段是对应于一个区间的，因此线段树也可以叫做区间树。

线段树是一棵二叉树，树中的每一个结点表示了一个区间[a,b]。每一个叶子节点上 a+1=b，这表示了一个初等区间（单元区间）。对于每一个内部结点 b-a>1，设根为[a,b]的线段树为 T（a,b）,则进一步将此线段树分为左子树 T（a,（a+b）/2），以及右子树 T（（a+b）/2,b）,直到分裂为一个初等区间为止。如图 6-63 给出的就是一棵对应于区间[1，11]的线段树。注意，这只是线段树的基本结构。通常利用线段树的时候需要在每个结点上增加一些特殊的数据域，并且它们是随线段的插入删除进行动态维护的。这因题而异，同时又往往是解题的灵魂。



区间[1，11]对应的线段树

图 6-63 是一个抽象的线段树，在实际处理问题时是先将任何一个要处理的线段（区间）[a,b]进行了一定的转换。即在处理问题的时候，首先对各个区间根据其端点进行离散化，即抽取出区间的端点，如有 N 个端点，然后对这些端点进行和自然数 1，2，…，N 对应，然后就可以得出如图 6-64 的线段树。如有 6 个区间[1，10]，[5，100]，[20，40]，[30，60]，[60，80]，[200，210]。对这些区间进行从小到大排序{1,5,10,20,30,40,60,80,100,200,210}，对应的标号就是{1,2,3,4,5,6,7,8,9,10,11}。因此图 6-63 对应的实际区间的线段树如图 6-64 所示。

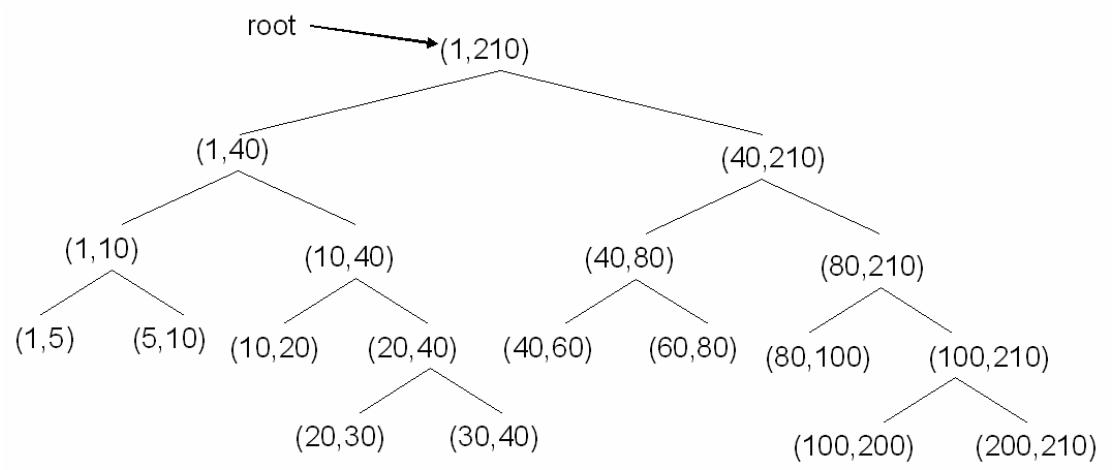


图 6-64 实际区间对应的线段树

线段树节点的数据结构和操作会根据不同的应用来调整，但是它也有基本的结构和操作。线段树节点的基本数据结构：

typedefstructnode{intxleft,xright;//节点对应区间的左端点和右端点坐标

structnode*left,*right;}Node;

线段树的基本操作包括插入操作和删除操作。

在节点 p 上插入一个区间[a,b]，insert（a,b,p）要求 p->xleft　a<b　p->xright，而且 a,b 都是离散点的坐标，insert（a,b,p）的处理流程是：

1．mid=p->left->xright；

2．若 a=p->xleft 且 b=p->xright，则执行有关的插入操作；返回；

3．若 b　mid，则执行 insert（a,b,p->left）；返回；//左边插入，递归执行

4．若 a　mid，则执行 insert（a,b,p->right）；返回；//右边插入，递归执行

5．执行 insert（a,mid,p->left）；执行 insert（mid,b,p->right）；返回；//两边都递归插入

这是一个递归过程。其中第 1 步执行完后，第 2,3,4,5 步只有一步会被执行。若执行了第 2 步，则称节点 p 为一个插入点，关键工作都是在插入点进行的。若执行了第 3,4 或 5 步，则 p 都不会被称为插入点。在插入点进行的关键工作会在不同的应用中体现为不同的操作，如可以给每个节点上定义一个 count 变量，该变量 count 用来记录覆盖该结点的线段条数，此时每次插入操作使 count 变量增 1。

设 a<b 是两个离散点的坐标，root 是线段树的根节点，称操作 insert（a,b,root）

为在线段树上插入区间[a,b]的操作，仍以上面的例子来说明插入操作：

（a）设在线段树上插入区间[1,10]，即运行 insert（1,10,root）。其结果是先后递归运行：

insert（1,10,（1,210））

insert（1,10,（1,40））

insert（1,10,（1,10））

而其中的插入点只有（1,10）这一个节点。

（b）在线段树上插入区间[5,100]，即运行 insert（5,100,root）的结果是先后递归运行：

insert（5,100,（1,210））

insert（5,40,（1,40））

insert（5,10,（1,10））

insert（5,10,（5,10））//插入点

insert（10,40,（10,40））//插入点

insert（40,100,（40,210））

insert（40,80,（40,80））//插入点

insert（80,100,（80,210））

insert（80,100,（80,100））//插入点

这次插入出现了（5,10）,（10,40）,（40,80）,（80,100）共 4 个插入点。

一般说来，若有 N 条离散线段，则线段树的叶子节点有 N 个，那么线段树的节点总数 2N，每次插入操作最多需要 2logN 次递归调用插入过程，也就最多有 2logN 插入点。

线段树的构造主要是对区间线段的处理，它往往被应用于几何计算问题中。线段树的作用主要体现在可以动态维护一些特征，例如说要得到线段树上线段并集的长度，可以增加一个数据域 p->M：如果 p->C>0，p->M=p->xright-p->xright;p->C=0 且 v 是叶子结点，p->M=0；p->C=0 且 v 是内部结点，p->M=p->left->M+p->right->M。只要每次插入或删除线段区间时，在访问到的结点上更新 M 的值，不妨称之为 UPDATA，就可以在插入和删除的同时维持好 M。求整个线段树的并集长度时，只要访问 root->M 的值。这在许多动态

维护的题目中是非常有用的，它使得每次操作的维护费用只有 logn。类似的，还有求并区间的个数等等。

（2）课程设计目的

掌握线段树的基本原理，编程实现线段树及其上的基本操作，应用线段树解决实际问题。

（3）基本要求

①编程实现线段树的数据结构定义、线段树的初始化操作、线段树的插入和删除操作。

②应用线段树解决一维线段上的计算几何问题，如求解这些线段的并集的长度，给定一个点，多少线段覆盖这个点等等。

③用线段树解决二维图形上的计算几何问题，如坐标轴的上半平面上有 N 个矩形。每个矩形都有一条边在 x 轴上，记这条边为区间[Ai,Bi]，记该矩形的高为 Hi，i=1,…,N。其中 1　Ai≤Bi≤10^9，1≤i≤N≤40000。

信息的输入格式：N

A1B1H1

A2B2H2

…

$A_NB_NH_N$

要求输出平面上被这些矩形覆盖的部分的面积。

④应用线段树解决二维图形上的计算几何问题，平面上有 N 个矩形，矩形的边平行于坐标轴。每个矩形左下角和右上角的坐标是（xli,yli）和（xui,yui），i=1,…,N。其中坐标值的范围是[-10000,10000]，1≤N≤5000。

输入格式：

N

xl1yl1xu1yu1

xl2yl2xu2yu2

…

xlNylNxuNyuN

要求输出平面上被这些矩形覆盖的部分轮廓的周长。

（4）实现提示

对于要求③，在节点结构中增加 intheight 即节点对应线段被覆盖的高度，初始为 0。

并在每个插入点 p 处执行 p->height=max{h,p->height}。所有矩形都处理完以后，就可以求总面积了。下面的过程 area（h,p）将节点 p 对应的区间上高度 h 的部分的面积累加到变量 sumarea 中：area（h,p），1.diff=p->height-h；2.若 diff>0，则执行 sumarea+=diff*（p->xright-p->xleft）；h=p->height；3.若 p->left 非空，则执行 area（h,p->left）；area（h,p->right）。

对于要求④，核心是设计一条竖直扫描线，从左到右扫描，每当扫描线固定在某个位置时，计算该扫描线被这些矩形覆盖的部分的长度（不妨称为扫描到的长度）和被这些矩形覆盖的区间的段数（不妨称为扫描到的段数）。例如，当扫描线位于虚线所在位置时，扫描到的段数是 2 段；扫描到的长度自然是这两段长度的和。注意到下面的事实：一是扫描到的长度和段数只有在扫过矩形的竖直边时才会发生变化。二是当扫描线从位置 1 移到位置 2 时：[2*位置 1 扫描到的段数*（位置 2 横坐标-位置 1 横坐标）]就是扫描线从位置 1 到位置 2 扫描到的水平轮廓的长度,[abs（位置 2 扫描到的长度-位置 1 扫描到的长度）]就是扫描线在位置 2 遇到的竖直轮廓的长度。需要说明的是：当扫描线恰好位于一条竖直边上时，其扫描到的长度和段数是有歧义的，定义此时扫描到的长度和段数为扫描线位于竖直边右边任意近位置时扫描到的长度和段数。后面的实际算法是一条竖直边一条竖直边的扫描的，如果有两条竖直边横坐标相同，那么求出的长度和段数与这里定义类似，只是分步得到。由上面的观察，计算轮廓周长 sum 的过程大致如下：（1）初始设 sum=0，scanlength=0，scanseg=0，scanpos=-10000。

（2）找下一条竖直边（x,low,up）=（竖直边的横坐标，下端点的纵坐标，上端点的纵坐标），计算当前扫描到的长度 newscanlength 和段数 newscanseg。

sum+=2*scanseg*（x-scanpos）;

sum+=abs（newscanlength-scanlength）;

scanlength=newscanlength;scanseg=newscanseg;

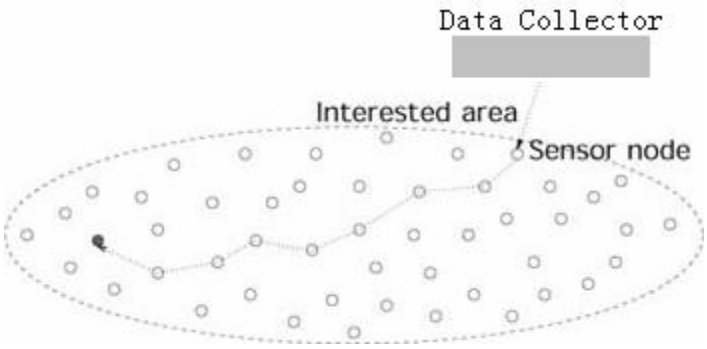
scanpos=x;

（3）若还有竖直边未扫描，则转（2）。

56 模拟sensornetwork的工作

（1）问题描述

Sensornetwork 是一种新型的网络，其基本结构如下图所示：该网络由两部分组成，Sensornode 集和 DataCollector。Sensornode（可简称为 Sensor）能够完成感知环境数据并将其发往 DataCollector 的功能。DataCollector 完成 Sensor 采集数据的收集，它就是一台带有无线接收功能的计算机。



Sensornetwork 图示

Sensornetwork 可应用到很多实际领域中，如在战争中将 Sensor 散播在防线的前沿，可以收集敌人的一些情报（如大规模的部队转移等）。Sensor 散播的地方称为 Interestedarea，Sensor 在这个区域内采集各自所在位置的数据，然后将采集到的数据传送到 DataCollector。各个 Sensor 之间通过无线广播通讯，由于 Sensor 广播能力的限制，它只能和位于自身的一定广播半径内的 Sensor 进行通讯，所以有些 Sensor 就需通过其它 Sensor，经过多次路由后才能到达 DataCollector（如上图）。如何路由由 Sensor 内保存的简单路由表来决定。DataCollector 的位置就在 InterstedArea 的边缘，且固定不动。

（2）课程设计目的

应用数据结构知识模拟一个新型网络系统。

（3）基本要求

①应用数据结构在单机上模拟 Sensornetwork 的工作。

②用 VC++（C 也可以）实现模拟系统，N 个 Sensor 和 1 个 DataCollector，其具体位置随机确定，InterestArea 就是屏幕。N 可配置，缺省为 100。

③Sensor 进行周期性采集，其采集周期可配置。

④Sensor 的广播半径固定，也是可配置的参数。

⑤路由算法自行选择或设计。

（4）实现提示

Sensor 集可组织成数组，它采集及收到的数据包用队列存储。具体细节也可参阅有关资料。

57 二值图像数字水印技术的实践

（1）问题描述

随着计算机通信技术的迅速发展，多媒体存储和传输技术的进步使存储和传输数字化信息成为可能，然而，这也使盗版者能以低廉的成本复制及传播未经授权的数字产品内容。数字水印就是近年来为实现数字产权保护而产生的技术。数字水印是永久镶嵌在其它数据（宿主数据）中具有可鉴别性的数字信号或模式，而且并不影响宿主数据的可用性。作为数字水印技术基本上应当满足下面几个方面的要求：（1）安全性：数字水印的信息应是安全的，难以篡改或伪造；（2）隐蔽性：数字水印应是不可知觉的，而且应不影响被保护数据的正常使用；（3）稳健性：数字水印必须难以被除去，如果只知道部分数字水印信息，那么试图除去或破坏数字水印将导致严重降质或不可用。

数字水印技术是通过一定的算法将一些标志性信息直接嵌到多媒体内容中，水印的嵌入和提取方法如图 6-76 所示：

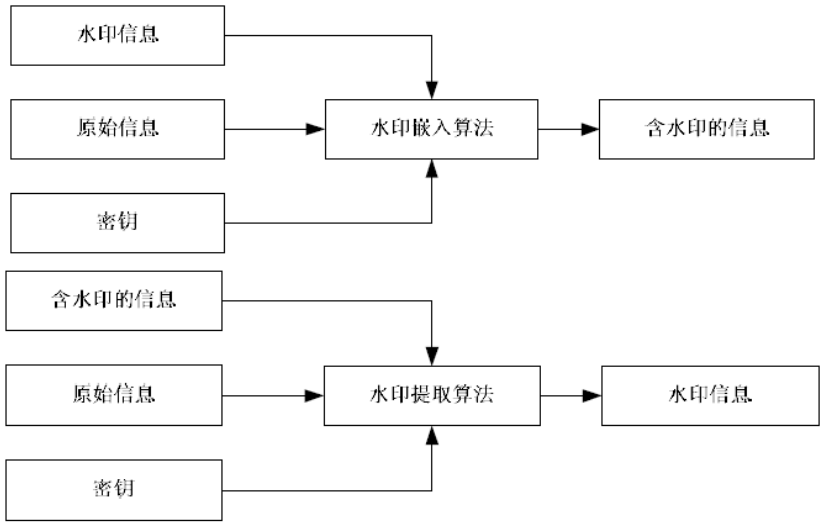


图 6-76 数字水印的嵌入和提取

数字水印的实现可以分为空间域数字水印和变换域数字水印两大类，较早的数字水印算法都是空间域上的，通过改变某些象素的灰度将要隐蔽的信息嵌入其

中，将数字水印直接加载在数据上。空间域方法具有算法简单、速度快、容易实现的优点。特别是它几乎可以无损的恢复载体图象和水印信息，可细分为如下几种方法：最低有效位法，该方法就是利用原始数据的最低几位来隐蔽信息的，具体取多少位以人的听觉或视觉系统无法察觉为原则。

Patchwork 方法及纹理映射编码方法，该方法是通过任意选择 N 对图象点，增加一点亮度的同时，降低相应另一点的亮度值来加载数字水印。文档结构微调方法，在通用文档图象（postcript）中隐藏特定二进制信息的技术，主要是通过垂直移动行距，水平调整字距，调整文字特性等来完成编码。

基于变换域的技术可以嵌入大量比特的数据而不会导致不可察觉的缺陷，往往通过改变频域的一些系数的值，采用类似扩频图象的技术来隐藏数字水印信息。这类技术一般基于常用的图象变换，如离散余弦变换（DCT）、小波变换（WT）、付氏变换（FT 或 FFT）以及哈达马变换（HadamardTraansform）等。频域方法具有如下优点：（1）在频域中嵌入的水印的信号能量可以分布到所有的象素上，有利于保证水印的不可见性；（2）在频域中可以利用人类视觉系统的某些特性，可以更方便、更有效的进行水印的编码。不过，频域变换和反变换过程中是有损的，同时其运算量也很大，对一些精确或快速应用的场合不太适合。目前常用的方法有平面隐藏法和基于 DCT 或 DFT 的系数隐藏法。其中基于分块的 DCT 是最常用的变换之一，现在所采用的静止图像压缩标准 JPEG 也是基于分块 DCT 的。

下面概要描述一种简单的二值图像的数字水印算法，本设计的基本目标就是该水印算法的编程实现。

二值图像又称为单色图像或黑白图像，一般用 1 或 0 表示黑色或白色像素的颜色值。二值图像数字水印的常见方法是：一、使用一个特定图像区域中的黑色像素个数来编码水印信息。把一个二值图像分解成一系列图像块 Bi，分别令 P0（Bi）和 P1（Bi）代表黑白像素在该块图像中所占的百分比。基本做法是判断如果 P1（Bi）>50%，则在该块中隐藏一个 1，如果 P0（Bi）>50%，则在该块中隐藏一个 0。隐藏时需要修改图像块中的一些像素的颜色，该修改是在那些邻近像素有相反的颜色的像素中进行的。二、将二值图像进行分块，使用一个与图像块同样大小的密钥二值图像块，该密钥图像块和每一格图像块按像素进行“与”

运算，根据“与”的结果确定是否在该块中隐藏数据，并隐藏咋样的数据。但这两种简单处理方法会导致图像质量下降，如会在一个全白的图像块中插入一个黑点，如也应避免在图像中的细线（一个像素宽）、直线边的中间像素、孤立像素等区域隐藏像素。所以二值图像的数字水印嵌入算法的关键是隐藏点的选取。

二值图像的数据隐藏应该着眼于图像黑（或白）色区域的边界上，但有些边界仍然不适合隐藏信息。下面给出不适合隐藏数据的图像边界：该像素既是其所在区域的左边界，又是其右边界（实际上是一条直线）；该像素既是其所在区域的上边界，又是其下边界；该像素只是左边界、右边界、上边界、下边界 4 种边界情况中的一种；该像素的周围 8 个像素中与该像素同色的所有像素都是既是左边界或右边界，同时又是其上边界和下边界（实际上是一个小点）。

此处给出一个算法，设二值图像为 I，其像素矩阵为 AM　N；数字水印为 W，是一个长度为 L 的二进制比特流 W={w1，w2，…，wL}。设 A（i:j,s:t）为矩阵 A 的从第 i 行到第 j 行，从第 s 列到第 t 列的子矩阵。数据隐藏算法如下：

第 1 步：计算图像边界。

A_L=A（0:（M-1）,0:（N-2））–A（0:（M-1）,1:（N-1））

A_R=A（0:（M-1）,1:（N-1））–A（0:（M-1）,0:（N-2））

A_T=A（0:（M-2）,0:（N-1））–A（1:（M-1）,0:（N-1））

A_B=A（1:（M-1）,0:（N-1））–A（0:（M-2）,0:（N-1））

将 A_L，A_R，A_T，A_B 中值为-1 的元素置为 0，得到 B_L，B_R，B_T，B_B。则 B_L，B_R，B_T，B_B 为图像 I 的左、右、上、下边界矩阵。

第 2 步：选择隐藏点。

$B=B_L+B_R+B_T+B_B$

$B_{LR}=B_L+B_R$

$B_{TB}=B_T+B_B$

矩阵 B_{LR} 和 B_{TB} 的元素值可能为 0，1，2。值为 2 表示该像素同时为左右或上下边界。

矩阵 B 中元素的可能值为 0，1，2，3，4，值为 3 和 4 的一定是左右或上下边界，值为 2 的像素可能会同时为左右边界或上下边界，此时需根据 BLR 和 BTB 的值加以区分。

计算 B′=（bij′），其中 bij′=0，如果 bij=3,4；bij′=0，如果 bij=2 且 bij^{LR}=2 或 bij^{TB}=2；bij′=0，如果 i=0 或 M 或者是 j=0 或 N；bij′=1，如果 bij=2 且 bij^{LR}≠2 与 bij^{TB}≠2。

再计算 B′′=（bij′′），其中 bij′′=0，如果 bij′=0；bij′′=0，如果 bij′=1 且

$$\sum_{u=i-1}^{i+1}\sum_{v=j-1}^{j+1}a_{uv}=\sum_{u=i-1}^{i+1}\sum_{v=j-1}^{j+1}b'_{uv}；\ b_{ij}''=1，\text{其他情况。}$$

第 3 步：水印数据处理。为确保水印数据的安全，输入一个密钥 K，根据该密钥产生一个长度为 L 的伪随机比特流 R={r1,r2,…,rL}（R 的产生方法是应用一个高级语言提供的伪随机函数，如 C 语言的 rand 函数，将 K 作为该函数的种子产生随机数，可产生 0-1 的随机数，如果该数小于等于 0.5 就输出 0，否则输出 1。也可以用其他方法获得。)将 W 与 R 按位异或就得到了 W ={r1 ,r2 ,…,rL }。

第 4 步：数据隐藏。设计一个遍历图像矩阵 A 诸元素的遍历算法，根据 B 和 W 计算获得 A 。A =（ai,j ），ai,j =ai,j，如果 bi,j =0；ai,j =wl ，如果 bi,j =1，其中 l=1,2,3…。A 对应的就是含有水印信息的新图像 IW。

第 5 步：水印数据的提取。第 1 步，第 2 步同上，在 A（图像 I）上可以计算出矩阵 B′′，此时可以获得隐藏在 A′（图像 IW）中的数据 S，注意需按照同样的遍历方法，然后根据 K 获得同样的伪随机序列 R，R 和 S 按位异或后就是数字水印。

（2）课程设计目的

对数字水印技术建立一定的认识，能建立位矩阵、位向量等 ADT，并能用这些 ADT 完成上述二值图像数字水印的嵌入和抽取。

（3）基本要求

①设计并实现位矩阵、位向量等 ADT，要求支持+、-、与、或、异或等基本操作。

②针对二值图像实现上述水印嵌入和提取算法。

③针对二值图像实现实现另一个水印基本做法：判断如果 P1（Bi）>50%，则在该块中隐藏一个 1，如果 P0（Bi）>50%，则在该块中隐藏一个 0。也可以自行设计一个嵌入方式。

④针对一幅二值图像（要求是 BMP 文件格式），分别用上面的两种方法嵌

入水印，查看嵌入水印后的图像 IW 的区别，水印信息可以设定为一段文字。

（4）实现提示

可以查阅关于数字水印方面的相关资料。

58 实现简单的数字图像处理

一幅图像是包含位置集和颜色集的数据。考虑二维灰度图像，位置集就是一个矩阵的行和列，矩阵的内容为颜色值，颜色为 0~255 间的整数，表示该位置的灰度等级，0 为黑色，255 为白色。

图像处理就是与该矩阵相关的计算，一种常见的计算就是通过一点和周围 8 个点的信息，共同决定该点的新值：如一点的新值为该点和周围 8 点颜色之和的平均，这一操作可用下图表示。

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

这样处理后图像会变得平滑，因此，称为平滑操作。如果将上述操作变为下图

-1	-1	-1
-1	9	-1
-1	-1	-1

操作后图像的边缘变得更加突出，被称为锐化操作。

实现上述图像的平滑和锐化操作。

编程任务：

①常见格式图像的读写（灰度图）

②设计上述平滑算子和锐化算子

③实现平滑操作和锐化操作

④观察处理后图像的变化，分析算子的作用。

59 统计英文单词数

给出一篇英文文章，文件不小于 1MB 的大小。统计其中的每个不同英文单词和总单词的数量。

实现提示：分别用链表和哈希表来实现，注意要给出不同大小文件耗费的时间，对时间性能进行进一步分析。关于英文文章，请自动生成文本文件。也可以从网络上下载几篇英文的文章，然后合并生成。

60 本科生导师制问题

问题描述：在高校的教学改革中，有很多学校实行了本科生导师制。一个班级的学生被分给几个老师，每个老师带领 n 个学生，如果老师还带研究生，那么研究生也可直接负责本科生。

本科生导师制问题中的数据元素具有如下形式：

⑴导师带研究生：（老师,（(研究生 1,（本科生 1,…,本科生 m)）,…)）

⑵导师不带研究生：（老师,（本科生 1,…,本科生 m)）

导师的自然情况只包括姓名、职称；

研究生的自然情况只包括姓名、班级；

本科生的自然情况只包括姓名、班级。

功能要求：要求完成以下功能：

⑴插入：将某位本科生或研究生插入到广义表的相应位置；

⑵删除：将某本科生或研究生从广义表中删除；

⑶查询：查询导师、本科生（研究生）的情况；

⑷统计：某导师带了多少个研究生和本科生；

⑸输出：将某导师所带学生情况输出。

61 表达式求值

【问题描述】

表达式求值是实现程序设计语言的基本问题之一，也是栈的应用的一个典型例子。设计一个程序，演示用算符优先法对算术表达式求值的过程。

【基本要求】

以字符序列的形式从终端上输入语法正确的、不含变量的整数表达式。利用教材中给出的算符优先关系，实现对算术四则混合运算表达式的求值，并仿照教材例 3-1 演示在求值中运算符栈、运算数栈、输入字符和主要操作的变化过程。

【实现提示】

（1）设置运算栈和运算数栈辅助分析算符优先关系。

（2）在输入表达式的字符序列的同时，完成运算符和运算数（整数）的识别处理，以及相应的运算。

（3）在识别出运算数的同时，要将其字符序列形式转换成整数形式。

62 校园导游咨询

【问题描述】

设计一个校园导游程序，为来访的客人提供各种信息查询服务。

【基本要求】

（1）设计你的学校的校园平面图，所含景点不少于 10 个。以图中顶点表示学校各景点，存放景点名称、代号、简介等信息；以边表示路径，存放路径长度等相关信息。

（2）为来访客人提供图中任意景点的问路查询，即查询任意两个景点之间的一条最短的简单路径。

（3）为来访客人提供图中任意景点相关信息的查询。

【测试数据】

由读者根据实际情况指定。

【实现提示】

一般情况下，校园的道路是双向通行的，可设校园平面图是一个无向网。顶点和边均含有相关信息。

63 图书管理信息系统的设计与实现

图书管理一般包括：图书采编、图书编目、图书查询及图书流通（借、还书）等，请编程实现上述功能。

具体设计要求：

（1）设计图书管理的存储结构，输入若干种书的记录。

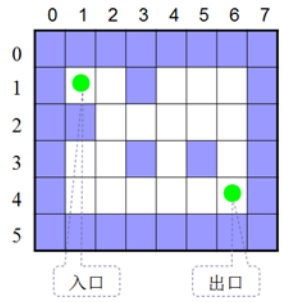
（2）实现关于书号、书名、作者及出版社的图书查询；

（3）实现图书的借还子系统，包括建立读者文件、借还书文件、读者管理及图书借还等相关处理

64 利用栈解决迷宫问题

一个迷宫的实例，如下图所示：

图中紫色方块为障碍，不能通行；白色方块可以通行；行进方向 4 个或 8 个；需求出从入口到出口的一条通道。



65 最近点对问题

问题场景：在应用中，常用诸如点、圆等简单的几何对象代表现实世界中的实体。在涉及这些几何对象的问题中，常需要了解其邻域中其他几何对象的信息。例如，在空中交通控制问题中，若将飞机作为空间中移动的一个点来看待，则具有最大碰撞危险的 2 架飞机，就是这个空间中最接近的一对点。这类问题是计算几何学中研究的基本问题之一。

问题描述：给定平面上 n 个点，找其中的一对点，使得在 n 个点的所有点对中，该点对的距离最小。严格地说，最接近点对可能多于 1 对。为了简单起见，这里只限于找其中的一对。

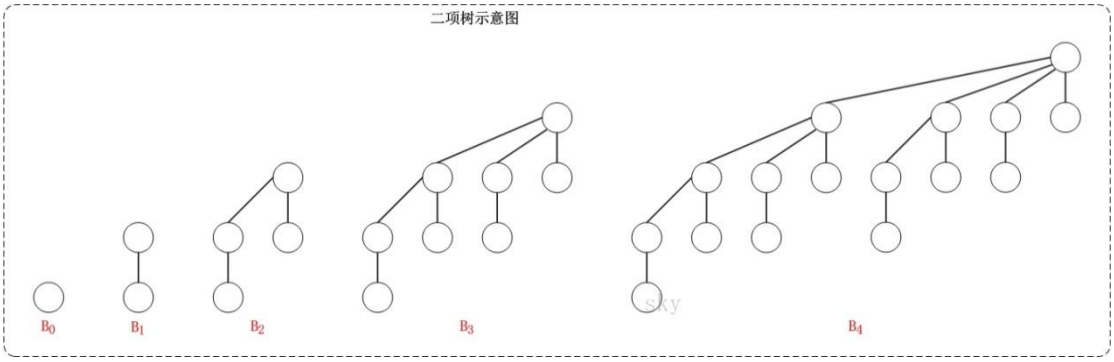
66 二项树的实现

二项树是一种递归定义的有序树。它的递归定义如下：

(01) 二项树 B0 只有一个结点；

(02) 二项树 Bk 由两棵二项树 B(k-1)组成的，其中一棵树是另一棵树根的最左孩子。

如下图所示：



上图的 B0、B1、B2、B3、B4 都是二项树。对比前面提到的二项树的定义：B0 只有一个节点，B1 由两个 B0 所组成，B2 由两个 B1 所组成，B3 由两个 B2

所组成，B4 由两个 B3 所组成；而且，当两颗相同的二项树组成另一棵树时，其中一棵树是另一棵树的最左孩子。

二项树具有以下性质：

[性质一] Bk 共有 2k 个节点。

如上图所示，B_0 有 2^0=1 节点，B_1 有 2^1=2 个节点，B_2 有 2^2=4 个节点，...

[性质二] Bk 的高度为 k。

如上图所示，B_0 的高度为 0，B_1 的高度为 1，B_2 的高度为 2，...

[性质三] B_k 在深度 i 处恰好有 C(k,i)个节点，其中 i=0,1,2,...,k。

C(k,i)是高中数学中阶乘元素，例如，C(10,3)=(10*9*8) / (3*2*1)=240

B_4 中深度为 0 的节点 C(4,0)=1

B_4 中深度为 1 的节点 C(4,1)= 4 / 1 = 4

B_4 中深度为 2 的节点 C(4,2)= (4*3) / (2*1) = 6

B_4 中深度为 3 的节点 C(4,3)= (4*3*2) / (3*2*1) = 4

B_4 中深度为 4 的节点 C(4,4)= (4*3*2*1) / (4*3*2*1) = 1

合计得到 B_4 的节点分布是(1,4,6,4,1)。

[性质四] 根的度数为 k，它大于任何其它节点的度数。

节点的度数是该结点拥有的子树的数目。

注意：树的高度和深度是相同的。关于树的高度的概念，《算法导论》中只有一个节点的树的高度是 0，而"维基百科"中只有一个节点的树的高度是 1。本文使用了《算法导论中》"树的高度和深度"的概念。

要求：实现二项树类，并实现各种基本操作。

67 点在多边形中的判断

在二维环境下，判断一个点是否在多边形内部。

注：请考虑各种多边形。

68 泊松分酒问题

法国著名数学家波瓦松在青年时代研究过一个有趣的数学问题：假设某人有 12 品脱的啤酒一瓶，想从中倒出六品脱，但是恰巧身边没有 6 品脱的容器，仅

有一个8品脱和一个5品脱的容器,怎样倒才能将啤酒分为两个6品脱呢？现在，请你设计一个程序，可以根据输入的满瓶容量（a），和两个空瓶的容量（b 和 c）对倒，获得最终需要的容量（d）。

69 游戏设计

针对传统的贪吃蛇游戏、俄罗斯方块等游戏，进行重新设计，实现一个新的游戏。

70 马踏棋盘

问题描述：将马随机放在国际象棋的 8* 8 棋盘 Bord[8Ⅱ8]的某个方格中，马按走棋规则进行移动。要求每个方格上只进入一次，走遍棋盘上全部 64 个方格。

任务要求：编制非递归程序，求出马的行走路线 ，并按求出的行走路线，将数字 1，2，…，64 依次填入一个 8* 8 的方阵，输出之。

测试数据：由读者指定,可自行指定一个马的初始位置。

实现提示：每次在多个可走位置中选择一个进行试探，其余未曾试探过的可走位置必须用适当结构妥善管理，以备试探失败时的“回溯”(悔棋)使用

71 BP神经网络的实现

利用 C++语言实现 BP 神经网络，并利用 BP 神经网络解决螨虫分类问题：

蠓虫分类问题：对两种蠓虫（A 与 B）进行鉴别，依据的资料是触角和翅膀的长度，已知了 9 支 Af 和 6 支 Apf 的数据如下：A: (1.24,1.27)，(1.36,1.74)，(1.38,1.64)，(1.38,1.82)，(1.38,1.90)，(1.40,1.70)，(1.48,1.82)，(1.54,1.82)，(1.56,2.08).B: (1.14,1.82)，(1.18,1.96)，(1.20,1.86)，(1.26,2.00)，(1.28,2.00)，(1.30,1.96).

要求：（1）阐述 BP 神经网络的结构构成及数学原理；

（2）利用 C++实现 BP 神经网络；

（3）利用 BP 神经网络实现螨虫分类

72 CNN开放题

自学 CNN、RNN 相关视频课程，自拟题目完成课设，题目拟定后需与代课老师商定。