实验题目与指导

实验分为基础实验、应用实验、扩展实验三类。

1）基础实验：主要验证教材中提到的基础类，深化理解和掌握理论知识；

2）应用实验：主要目标是应用教材中教授的某一个知识点，自己设计方案解决实际的问题，培养学生简单的应用能力；

3）扩展实验：该类实验逻辑结构较为复杂，需要将多个知识点融会贯通，设计较为复杂的方案，以解决实际的问题并具备扩展到数据结构课程设计的功能。该类实验代码实现量较大，一般可两人合作完成。

实验一线性表

1 实验目的

通过选择下面四个题目之一进行实现，掌握如下内容：

- 熟悉 C++语言的基本编程方法，掌握集成编译环境的调试方法
- 学习指针、模板类、异常处理的使用
- 掌握线性表的操作的实现方法
- 学习使用线性表解决实际问题的能力

2 实验内容

2.1 题目 1——基础实验

根据线性表的抽象数据类型的定义，选择下面任一种链式结构实现线性表，并完成线性表的基本功能。

线性表存储结构（五选一）：

1、 带头结点的单链表

2、 不带头结点的单链表

3、 循环链表

4、 双链表

5、 静态链表

线性表的基本功能：

1、 构造：使用头插法、尾插法两种方法

2、 插入：要求建立的链表按照关键字从小到大有序（静态链表不要求该项）

3、 删除

4、 查找

5、 获取链表长度

6、 销毁

7、 其他：可自行定义

编写测试 main()函数测试线性表的正确性。

2.2 题目 2——基础实验

有序链表合并问题的求解。

设有两条有序链表（即 data 域元素的关键字由前往后不断增大），试设计算法，将这两条链表合并为一条新的有序链表，原链表不变。两条链表中 data 域关键字相同的元素只选取一个存储到新的有序链表中，不同的元素都存储到新的有序链表中。

要求：

- 直接编写链表的友元函数完成该功能。
- 链表的 data 域可存储用户自定义类对象。
- 编写测试 main()函数测试线性表的正确性

2.3 题目 3——应用实验

利用线性表实现一个通讯录管理，通信录的数据格式如下：

```
struct DataType
{
        int ID;                 //编号
        char name[10];          //姓名
        char ch;                //性别
        char phone[13];         //电话
        char addr[31];          //地址
};
```

要求：

- 实现通讯录的建立、增加、删除、修改、查询等功能
- 能够实现简单的菜单交互，即可以根据用户输入的命令，选择不同的操作。
- 能够保存每次更新的数据
- 能够进行通讯录分类，比如班级类、好友类、黑名单等等（选作）
- 编写测试 main()函数测试线性表的正确性

2.4 题目 4——应用实验

利用线性表实现一个一元多项式 Polynomial

$$f(x) = a_0 + a_1X + a_2X^2 + a_3X^3 + \cdots + a_nX^n$$

提示：

Polynomial 的结点结构如下：

```
struct term
{
```

```
        float coef;     //系数
        int expn;       //指数
};
```

可以使用链表实现，也可以使用顺序表实现。

要求：

- 能够实现一元多项式的输入和输出
- 能够进行一元多项式相加
- 能够进行一元多项式相减
- 能够计算一元多项式在 x 处的值
- 能够计算一元多项式的导数
- 能够进行一元多项式相乘
- 编写测试 main()函数测试线性表的正确性

2.5 题目 5——应用实验

用链表实现大整数加减法操作：

32 位机器直接操作的数据最大为 32 个 bit，若超过 32bit，则需要单独设计算法。在这里，可以用链表每个结点存储大整数的每一位的十进制数字，则可以进行大整数的算数运算，该实验仅实现加减法操作。

要求:

1， 随机产生 2 个 1~50 位的数字串，并存储到 2 个链表中。

2， 进行加法或减法操作，结果存储到新的链表中。

3， 打印运算结果。

2.6 题目 6——应用实验

动态内存管理是操作系统的基本功能之一，用于响应用户程序对内存的申请和释放请求。初始化时，系统只有一块连续的空闲内存；然后，当不断有用户申请内存时，系统会根据某种策略选择一块合适的连续内存供用户程序使用；当用户程序释放内存时，系统将其回收，供以后重新分配，释放时需要计算该内存块的左右是否也为空闲块，若是，则需要合并变成更大的空闲块。

试设计用于模拟动态内存管理的内存池类。

基本要求：

- 实现内存池 MemoryPool(int size)的初始化

- 实现 Allocate(int size)接口
- 实现 Free(void *p)接口
- 实现内存池的析构
- 在分配内存空间时，可选择不同的内存分配策略：最佳拟合策略、最差拟合策略或最先拟合策略。实现其中至少两种分配策略。

编写测试 main()函数对类中各个接口和各种分配策略进行测试，并实时显示内存池中的占用块和空闲块的变化情况。

3 程序要求

1、 注意异常处理的使用，比如删除空链表时需要抛出异常；

2、 注意内存的动态申请和释放，是否存在内存泄漏；

3、 优化程序的时间性能；

4、保持良好的编程的风格：

- 代码要简洁
- 代码段与段之间要有空行和缩近
- 标识符名称应该与其代表的意义一致
- 函数名之前应该添加注释说明该函数的功能
- 关键代码应说明其功能

实验二 扩展线性表

1 实验目的

通过选择下面五个题目之一进行实现，掌握如下内容：

- 进一步掌握指针、模板类、异常处理的使用
- 掌握栈的操作的实现方法
- 掌握队列的操作的实现方法
- 学习使用栈解决实际问题的能力
- 学习使用队列解决实际问题的能力
- 学习使用多维数组解决实际问题的能力

2 实验内容

2.1 题目 1——基础实验

根据栈和队列的抽象数据类型的定义，按要求实现一个栈或一个队列的基本功能（四选一）。要求：

1、 实现一个共享栈

2、 实现一个链栈

3、 实现一个循环队列

4、 实现一个链队列

编写测试 main()函数测试栈或队列的正确性。

2.2 题目 2——基础实验

根据三元组的抽象数据类型的定义，使用三元组表实现一个稀疏矩阵。三元组的基本功能：

1、三元组的建立

2、三元组转置

3、三元组相乘

4、其他：自定义操作

编写测试 main()函数测试三元组的正确性

测试数据：

$$\begin{pmatrix} 0 & 12 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 14 & 0 \\ 0 & 0 & 13 & 0 & 0 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & 0 & 0 \\ 15 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2.3 题目 3——应用实验

题目：八皇后问题。

八皇后问题 19 世纪著名的数学家高斯于 1850 年提出的。他的问题是：在 8*8 的棋盘上放置 8 个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列、同一斜线上。请设计算法打印所有可能的摆放方法。

提示：

1、可以使用递归或非递归两种方法实现

2、实现一个关键算法：判断任意两个皇后是否在同一行、同一列和同一斜线上

2.4 题目 4——应用实验

题目：迷宫求解问题。

迷宫求解问题如下：心理学家把一只老鼠从一个无顶盖的大盒子的入口赶进迷宫，迷宫中设置很多隔壁，对前进方向形成了多处障碍，心理学家在迷宫的唯一出口放置了一块奶酪，吸引老鼠在迷宫中寻找通路以到达出口，测试算法的迷宫如下图所示。

提示：

1、可以使用递归或非递归两种方法实现

2、老鼠能够记住已经走过的路，不会反复走重复的路径

3、可以自己任意设置迷宫的大小和障碍

4、使用“穷举求解”的方法

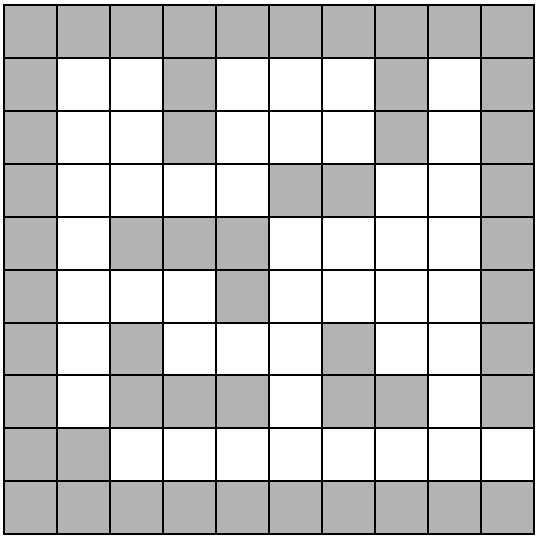


图 1-1　迷宫地图示例

2.5 题目 5——应用实验

表达式求值是程序设计语言编译中最近本的问题，它要求把一个表达式翻译成能够直接求值的序列。例如用户输入字符串“14+((13-2)*2-11*5)*2”，程序可以自动计算得到最终的结果。在这里，我们将问题简化，假定算数表达式的值均为非负整数常数，不包含变量、小数和字符常量。

试设计一个算术四则运算表达式求值的简单计算器。

基本要求：

1、 操作数均为非负整数常数，操作符仅为+、-、*、/、（和）；

2、 编写 main 函数进行测试。

2.6 题目 6——应用实验

利用队列结构实现车厢重排问题。车厢重排问题如下：

一列货车共有 n 节车厢，每个车厢都有自己的编号，编号范围从 1~n。给定任意次序的车厢，通过转轨站将车厢编号按顺序重新排成 1~n。转轨站共有 k 个缓冲轨，缓冲轨位于入轨和出轨之间。开始时，车厢从入轨进入缓冲轨，经过缓冲轨的重排后，按 1~n 的顺序进入出轨。缓冲轨按照先进先出方式，编写一个算法，将任意次序的车厢进行重排，输出每个缓冲轨中的车厢编号。

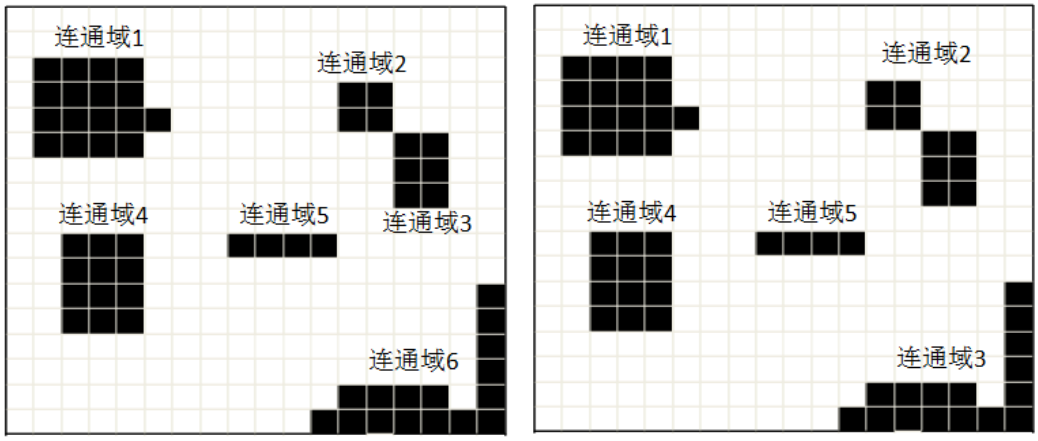
提示：

1、 一列火车的每个车厢按顺序从入轨进入不同缓冲轨，缓冲轨重排后的进入出轨，重新编排成一列货车。比如：编号为 3 的车厢进入缓冲轨 1，则下一个编号小于 3 的车厢则必须进入下一个缓冲轨 2，而编号大于 3 的车厢则进入缓冲轨 1，

排在 3 号车厢的后面，这样，出轨的时候才可以按照从小到大的顺序重新编排。

2.7 题目 7——扩展实验

在仅有黑色像素和白色像素的图像中，将相邻的黑色像素构成的点集称为一个连通域。连通域标记算法把连通区域所有像素设定同一个标记，常见的标记算法有四邻域标记算法和八邻域标记算法。四邻域标记算法中，当前黑点与上、下、左、右任意相邻黑点属于同一连通域，图 4（a）给出了四邻域连通域示意图。八邻域标记算法中，当前黑点与上、下、左、右及左上、左下、右上、右下任意相邻黑点属于同一连通域，图 4（b）给出了八邻域连通域示意图。



（a）四邻域连通域（b）八邻域连通域

图 1-2 图像连通域示意图

试编写二值图像四邻域连通域标记算法，设图像采用 01 矩阵表示。要求：

1、 算法尽可能优化

2、 输出每个像素点所属的连通域标记

3、 编写测试 main()函数测试三元组的正确性

测试数据：

1010101

10　1　0　1　11

1011001

1001111

1110001

0010001

1111111

2.8 题目 8——扩展实验

实现一个识别 BMP 文件的图像类，能够进行以下图像处理。

基本要求：

1、能够将 24 位真彩色 Bmp 文件读入内存；

2、能够将 24 位真彩色 Bmp 文件重新写入文件；

3、能够将 24 位真彩色 Bmp 文件进行 24 位灰度处理；

4、能够将 24 位灰度 Bmp 文件进行 8 位灰度处理；

5、能够将 8 位灰度 Bmp 文件转化成黑白图像；

6、能够将图像进行平滑处理；

7、其他：自定义操作，比如翻转、亮度调节、对比度调节、24 位真彩色转 256 色等。

提示：

1、参考教材《数据结构与 STL》第四章 4.4 小节。

2、灰度处理的转换公式

Grey=0.3*Red+0.59*Blue+0.11*Green

3、平滑处理采用邻域平均法进行，分成 4 邻域和 8 邻域平滑，基本原理就是将每一个像素点的值设置为其周围各点像素值得平均值。

4、亮度调节公式，a 为亮度调节参数，0 < a < 1, 越接近 0，变化越大

R = pow (R, a) * pow (255, 1 - a)

G = pow (G, a) * pow (255, 1 - a)

B = pow (B, a) * pow (255, 1 - a)

5、对比度调节公式，a 为对比度调节参数，-1< a < 1，（中间值一般为 128）

R=中间值+(R-中间值)*(1+a)

G=中间值+(G-中间值)*(1+a)

B=中间值+(B-中间值)*(1+a)

注意：调整对比度的时候容易发生越界，需要进行边界处理

6、24 位真彩色转 256 色，需要手动添加颜色表在 BMP 头结构中，可以使用位截断法、流行色算法、中位切分算法、八叉树算法等方法实现。

3 代码要求

1、 注意内存的动态申请和释放，是否存在内存泄漏；

2、 优化程序的时间性能；

3、 递归程序注意调用的过程，防止栈溢出

4、保持良好的编程的风格：

- 代码要简洁
- 代码段与段之间要有空行和缩近
- 标识符名称应该与其代表的意义一致
- 函数名之前应该添加注释说明该函数的功能
- 关键代码应说明其功能

实验三 树

1 实验目的

通过选择下面两个题目之一进行实现，掌握如下内容：

- 掌握二叉树基本操作的实现方法
- 了解哈夫曼树的思想和相关概念
- 学习使用二叉树解决实际问题的能力

2 实验内容

2.1 题目 1——基础实验

根据二叉树的抽象数据类型的定义，使用二叉链表实现一个二叉树。

二叉树的基本功能：

1、二叉树的建立

2、前序遍历二叉树

3、中序遍历二叉树

4、后序遍历二叉树

5、按层序遍历二叉树

6、求二叉树的深度

7、求指定结点到根的路径

8、二叉树的销毁

9、其他：自定义操作

编写测试 main()函数测试二叉树的正确性

思考问题（选作）：

1、若数据量非常大，如何使得构造二叉树时栈不溢出？使用非递归方式编写新的二叉树的构造函数，建立二叉树。提示：可以使用 STL 中的 stack 来辅助实现。

2、若二叉树的每一个结点具有数值，如何搜索二叉树，找到指定值的叶子结点？

3、若已知叶子结点的指针，如何输出从根到该叶子的路径？

2.2 题目 2——应用实验

利用二叉树结构实现哈夫曼编/解码器。

基本要求：

1、 初始化(Init)：能够对输入的任意长度的字符串 s 进行统计，统计每个字符的频度，并建立哈夫曼树

2、 建立编码表(CreateTable)：利用已经建好的哈夫曼树进行编码，并将每个字符的编码输出。

3、 编码(Encoding)：根据编码表对输入的字符串进行编码，并将编码后的字符串输出。

4、 译码(Decoding)：利用已经建好的哈夫曼树对编码后的字符串进行译码，并输出译码结果。

5、 打印(Print)：以直观的方式打印哈夫曼树（选作）

6、 计算输入的字符串编码前和编码后的长度，并进行分析，讨论赫夫曼编码的压缩效果。

7、 可采用二进制编码方式（选作）

测试数据：

I love data Structure, I love Computer。I will try my best to study data Structure.

提示：

1、用户界面可以设计为“菜单”方式：能够进行交互。

2、根据输入的字符串中每个字符出现的次数统计频度，对没有出现的字符一律不用编码。

3 代码要求

1、 注意内存的动态申请和释放，是否存在内存泄漏；

2、 优化程序的时间性能；

3、保持良好的编程的风格：

- 代码要简洁
- 代码段与段之间要有空行和缩近
- 标识符名称应该与其代表的意义一致
- 函数名之前应该添加注释说明该函数的功能
- 关键代码应说明其功能

实验四 图

1 实验目的

通过选择下面 5 个题目之一进行实现，掌握如下内容：

- 掌握图基本操作的实现方法
- 了解最小生成树的思想和相关概念
- 了解最短路径的思想和相关概念
- 学习使用图解决实际问题的能力

2 实验内容

2.1 题目 1——基础实验

根据图的抽象数据类型的定义，使用邻接矩阵或邻接表实现一个图。

图的基本功能：

1、图的建立

2、图的销毁

3、深度优先遍历图

4、广度优先遍历图

5、其他：比如连通性判断等自定义操作

编写测试 main()函数测试图的正确性

思考问题（选作）：

1、若测试数据量较大，如何使得栈不溢出？使用非递归方式编写新的深度优先遍历函数。提示：可以使用 STL 中的 stack 来辅助实现。

2.2 题目 2——应用实验

根据图的抽象数据类型的定义，使用邻接矩阵实现图的下列算法（三选一）。

1、使用普里姆算法生成最小生成树

2、使用克鲁斯卡尔算法生成最小生成树

3、求指定顶点到其他各顶点的最短路径

编写测试 main()函数测试算法的正确性

思考问题（选作）：

1、最短路径 D 算法，是否可以优化？请写出优化的思路并计算时间复杂度，同时实现一个新的优化的最短路径算法。

2.3 题目 3——应用实验

问题：对下图所示的地图进行染色，要求使用尽可能少的颜色进行染色，完成该算法。

测试数据：

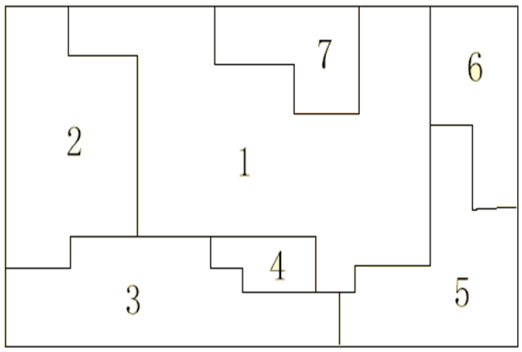


图 1-3　地图示例

提示：

1、利用图的着色思想解决该问题。图的着色方法指的是利用图的顶点存储地图上需要的染色的区域，利用图的边表示图上区域之间是否相邻的关系，比如区域 1 和区域 2 相邻，则图中顶点 1 和顶点 2 之间就画一条边，这是地图的储存。然后将相邻的顶点使用不同颜色进行着色，不相邻的顶点使用相同的颜色进行着色，即可完成该算法。

2.4 题目 4——应用实验

问题：田径会比赛安排问题

设某个田径运动会共有七个项目的比赛，分别为 100 米、200 米、跳高、跳远、铅球、铁饼和标枪。每个选手最多参加 3 个项目，现有六名选手参赛，他们选择的项目如表 1-1 所示。考虑到每个选手的参加的各个项目不能同时进行，则如何设计合理的比赛日程，使运动

会在尽可能短的时间内完成？

测试数据：

姓名	项目 1	项目 2	项目 3
张凯	跳高	跳远	
王刚	100m	200m	铁饼
李四	跳高	铅球	
张三	跳远	标枪	
王峰	铅球	标枪	铁饼
李杰	100m	跳远	

提示：

1、利用图的着色思想解决该问题

2、可以使用 STL 相关内容辅助解决该问题

2.5 题目 5——扩展实验

问题：多叉路口交通灯的问题。

假如一个如下图所示的五叉路口，其中 C、E 是箭头所示的单行道，如何设置路口的交通灯，使得车辆之间既不相互碰撞，又使交通流量最大？

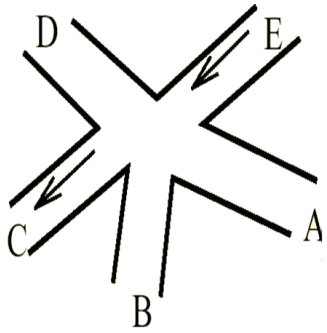


图 1-4　五叉路口

提示：

1、13 种行使路线　AB、AC、AD、 BA、BC、BD、DA、DB、DC、 EA、EB、EC、ED，不能同时行使的路线，比如 AB、BC 等，借助图的顶点表示行使路线，图中的边表示不能同时行使的路线，则可以划出下图的逻辑示意图：

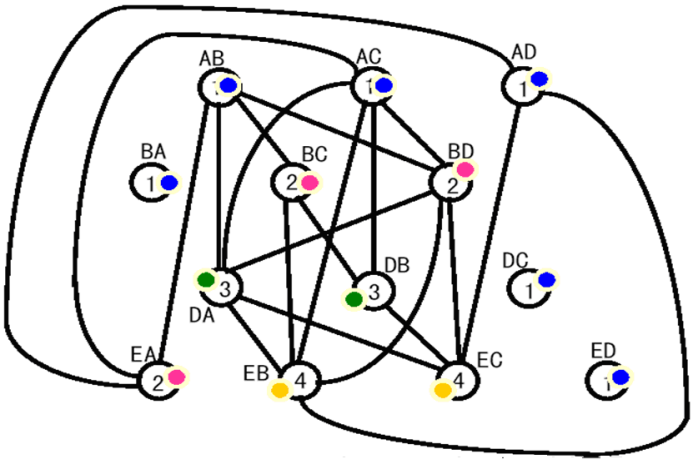


图 1-5　五叉路口建模

2、使用图的着色问题求解方法，使用最少的颜色进行着色，则是最优结果。

3 代码要求

1、 注意内存的动态申请和释放，是否存在内存泄漏；

2、 优化程序的时间性能；

3、保持良好的编程的风格：

- 代码要简洁
- 代码段与段之间要有空行和缩近
- 标识符名称应该与其代表的意义一致
- 函数名之前应该添加注释说明该函数的功能
- 关键代码应说明其功能

实验五 查找

1 实验目的

通过选择下面三个题目之一进行实现，掌握如下内容：

- 掌握树表查找的相关操作和技术优缺点
- 学习使用树表解决实际查找问题的能力
- 学习掌握使用散列技术解决实际问题的能力
- 举一反三，提升扩展现有查找技术优化解决方法

2 实验内容

2.1 题目 1——基础实验

根据二叉排序树的抽象数据类型的定义，使用二叉链表实现一个二叉排序树。二叉排序树的基本功能：

1、二叉排序树的建立

2、二叉排序树的查找

3、二叉排序树的插入

4、二叉排序树的删除

5、二叉排序树的销毁

6、其他：自定义操作

编写测试 main()函数测试二叉排序树的正确性。

2.2 题目 2——扩展实验

根据平衡二叉树的抽象数据类型的定义，使用二叉链表实现一个平衡二叉树。二叉树的基本功能：

1、平衡二叉树的建立

2、平衡二叉树的查找

3、平衡二叉树的插入

4、平衡二叉树的删除

5、平衡二叉树的销毁

6、其他：自定义操作

编写测试 main()函数测试平衡二叉树的正确性。

2.3 题目 3——扩展实验

题目：中文分词

在对中文文本进行信息处理时，常常需要应用中文分词（Chinese Word Segmentation）技术。所谓中文分词，是指将一个汉字序列切分成一个一个单独的词。中文分词是自然语言处理、文本挖掘等研究领域的基础。对于输入的一段中文，成功的进行中文分词，使计算机确认哪些是词，哪些不是词，便可将中文文本转换为由词构成的向量，从而进一步抽取特征，实现文本自动分析处理。

中文分词有多种方法，其中基于字符串匹配的分词方法是最简单的。它按照一定的策略将待分析的汉字串与一个“充分大的”中文词典中的词条进行匹配，若在词典中找到某个字符串，则匹配成功（识别出一个词）。按照扫描方向的不同，串匹配方法可以是正向匹配或逆向匹配；按照不同长度优先匹配的情况，可以分为最大（最长）匹配和最小（最短）匹配；按照是否与词性标注过程相结合，又可以分为单纯分词方法和分词与标注相结合的一体化方法。以上无论哪种方法，判断一个汉字串是否是词典中的词都是必须的，如何快速实现其快速匹配呢？

本题目要求采用散列技术实现基于字典的中文分词。学习设计合理的 Hash 函数构建 Hash 表中是关键来完成题目要求，编写测试 main()函数测试算法的正确性。

提示：

1、在网上查找并获取一个最小词典；

2、了解汉字编码 GB2312 技术。

3 代码要求

1、 注意内存的动态申请和释放，是否存在内存泄漏；

2、 优化程序的时间性能；

3、保持良好的编程的风格：

- 代码段与段之间要有空行和缩近

- 标识符名称应该与其代表的意义一致
- 函数名之前应该添加注释说明该函数的功能
- 关键代码应说明其功能

实验六 排序

1 实验目的

通过选择下面五个题目之一进行实现，掌握如下内容：

- 掌握各种排序算法的实现方法和算法优劣
- 学习使用排序算法解决实际问题的能力
- 举一反三，提升扩展现有排序技术优化解决方法

2 实验内容

2.1 题目 1——基础实验

使用简单数组实现下面各种排序算法，并进行比较。

排序算法：

1、起泡排序

2、直接插入排序

3、简单选择排序

4、希尔排序

5、快速排序

6、堆排序

7、归并排序

8、计数排序

9、桶排序

10、基数排序

要求：

1、测试数据分成三类：正序、逆序、随机数据

2、对于这三类数据，比较上述排序算法中关键字的比较次数和移动次数（其中关键字交换计为 3 次移动）。

3、对于这三类数据，比较上述排序算法中不同算法的执行时间，精确到微秒

4、对 2 和 3 的结果进行分析，验证上述各种算法的时间复杂度

编写测试 main()函数测试排序算法的正确性。

2.2 题目 2——应用实验

使用链表实现下面各种排序算法，并进行比较。

排序算法：

1、插入排序

2、冒泡排序

3、快速排序

4、简单选择排序

5、其他

要求：

1、测试数据分成三类：正序、逆序、随机数据

2、对于这三类数据，比较上述排序算法中关键字的比较次数和移动次数（其中关键字交换计为 3 次移动）。

3、对于这三类数据，比较上述排序算法中不同算法的执行时间，精确到微秒（选作）

4、对 2 和 3 的结果进行分析，验证上述各种算法的时间复杂度

编写测试 main()函数测试排序算法的正确性

2.3 题目 3——应用实验

问题：基于散列技术的排序

假设一个文件包含至多 1 亿条数据，如下图所示，每条数据都是一个 7 位的整数，每个整数至多出现一次，如何利用最小的内存，和无限大的硬盘空间，利用基于散列表技术来实现快速排序？

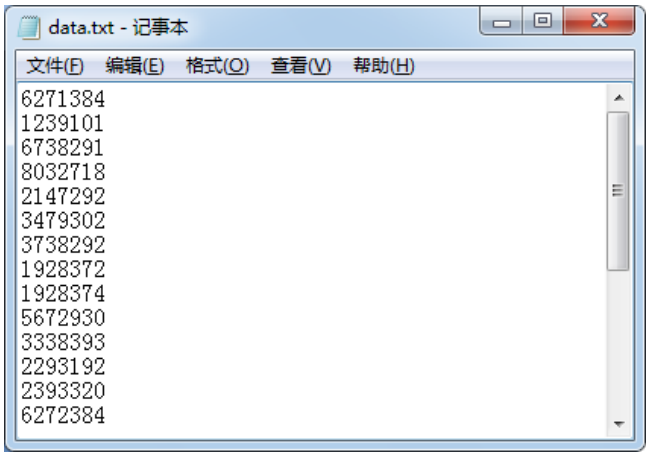


图 1-6 data.txt 文件格式

提示： 1、采用类似位图的方法

2、可采用 STL 进行辅助实现

2.4 题目 4——扩展实验

问题：机器调度问题

有 m 台机器处理 n 个作业，设作业 i 的处理时间为 ti，则对 n 个作业进行机器分配，使得：

1）一台机器在同一时间内只能处理一个作业；

2）一个作业不能同时在两台机器上处理；

3）作业 i 一旦运行，需要连续 ti 个时间单位。

设计算法进行合理调度，使得 m 台机器上处理处理 n 个作业所需要的总时间最短。

测试数据：7 个作业，所需时间分别为{2，14，4，16，6，5，3}，有三台机器，编号为 m1，m2 和 m3。

其中一种可能的调度方案如下：

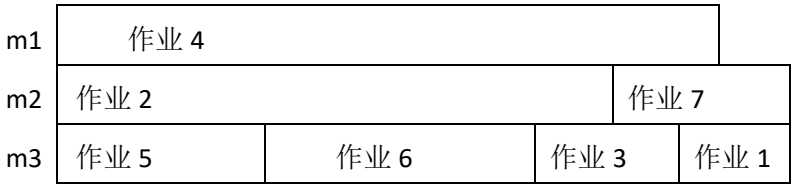


图 1-7 调度方案示例

时间分配：

m1 机器 : 16

m2 机器: 14 + 3 = 17

m3 机器: 6 + 5 + 4 + 2 = 17

总的处理时间是 17。

2.5 题目 5——扩展实验

题目：大数据排序问题

大数据排序问题一般也称为外部排序问题，通常我们将整个排序过程中不涉及数据的内

外存交换，待排序的记录可以全部存放在内存中的排序方法称为内部排序。但对于一个大型文件中的海量数据，显然是不可能将所有待排序数据一次装入有限的内存中，因此在排序过程中需要频繁的进行内外存交换，这种排序称为外部排序。

本题目假设一文件含 10000 个记录，按照内存一次最多可装入 2000 个记录作为约束，编写算法实现这 10000 个记录的排序，并测试排序算法的正确性。

提示：10000 条记录可以随机产生

3 代码要求

1、 注意内存的动态申请和释放，是否存在内存泄漏；

2、 优化程序的时间性能；

3、保持良好的编程的风格：

- 代码段与段之间要有空行和缩近
- 标识符名称应该与其代表的意义一致
- 函数名之前应该添加注释说明该函数的功能
- 关键代码应说明其功能