《数据结构与算法》

课程设计报告

学　　号：20171002608

班级序号：33

姓　　名：徐鸿飞

指导教师：李圣文

成　　绩：

中国地质大学信息工程学院软件工程系

2018 年 12 月

数据结构综合应用

1．链表求交集

【问题描述】对两个链表求交集，并链入另一个链表。

【实现思路】

分别用 pa，pb，pc 保存链表 la，lb，lc 的头指针，对 la 链表进行遍历，每次遍历中，对 lb 链表进行遍历，如果两个遍历指针指向的值相等，则将 lb 上的该结点链入 lc，并修改相关指针，若不相等，直接指针后移。

【运行结果】

```
D:\code\shujujiegou\大实习\链表交集\Debug>链表交集.exe
链表1:
1 2 4 5 6 8 9 15 28 49 100
链表2:
2 3 5 8 9 19 29 30 50 100
交集:
2 5 8 9 100
```

【核心代码】

链表求交集.cpp

2．双向链表【Lintcode 378】

【问题描述】 将一棵二叉搜索树按照中序遍历转换成双向链表。要求：输入一棵二叉搜索树，将该二叉搜索树转换成一个排序的双向链表。其中：空间复杂度 O(1)，不能创建任何新的结点，只能通过调整树中结点指针的指向来完成。

【样例输入】

【样例输出】 1<->2<->3<->4<->5

【通过截图】

【实现思路】

因为 Lintcode 中的链表节点和二叉树结点是两个不同的类，所以无法不创建新的结点，只通过调整树中结点指针来完成转换。我的思路为：按照书上的算法中序遍历二叉树，对于每次取出的结点，如果此时的双向链表为空，创建一个新的结点作为这个二叉链表的首结点，

并用临时指针保存，如果链表不为空则让创造新的结点，让临时指针的 next 指向该结点，该结点的 prev 指向临时指针，然后指针后移，最后返回首指针即可。

【核心代码】

双向链表.cpp

3. 最大矩形元素之和【Lintcode 1278】

【问题描述】 不超过 K 的最大矩形元素之和。给定一个非空的二维矩阵 matrix 和一个整数 k，找出 matrix 中的一个矩形，该矩形内的元素之和最大并且不超过 k，返回这个最大和。

【样例输入】 matrix = [[1, 0, 1],
[0, -2, 3]
] k = 2

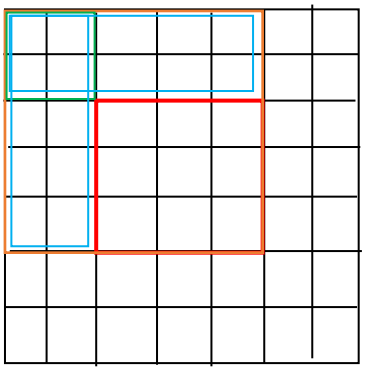
【样例输出】 矩形[[0, 1], [-2, 3]]，矩阵之和最大为 2，且 2 是不超过 k (k = 2)的最大数字。

【通过截图】

【实现思路】

刚开始想的是直接遍历，每次遍历都计算一遍，但是这样做时间复杂度太高，85%的时候就会 Time Limit Exceeded，后来的想法是：先做一个和原矩阵一样大的矩阵，i 行 j 列储存前 i 行 j 列的和，之后在计算每个矩形元素之和的时候就不用做大量计算了，对于左上角（i1，j1）右下角（i2，j2）的矩形来说，计算方式为：先用前 i2 行 j2 列的和减去前 i2 行 j1-1 列的和和前 i1-1 行 j2 列的和，再加上重复减的前 i1 行 j1 列的和。对于每个矩形和判断是否小于等于 k，若是则更新当前返回的结果值。

如图所示，红色矩形的和=橙色-蓝色+绿色
即=sum[i2][j2]-
sum[i2][j1 - 1]-
sum[i1 - 1][j2]+
sum[i1 - 1][j1 - 1]



【核心代码】

最大矩形之和.cpp

4. 最高频的 K 个单词【Lintcode 471】

【问题描述】 给一个单词列表，求出这个列表中出现频次最高的 K 个单词。（你需要按照单词的词频 排序后输出，越高频的词排在越前面。如果两个单词出现的次数相同，则

词典序小的排在前 面。）要求：O(n log k)的时间和 O(n) 的额外空间完成。提示：堆的使用。

【样例输入】 给出单词列表： [

"yes", "lint", "code",

"yes", "code", "baby",

"you", "baby", "chrome",

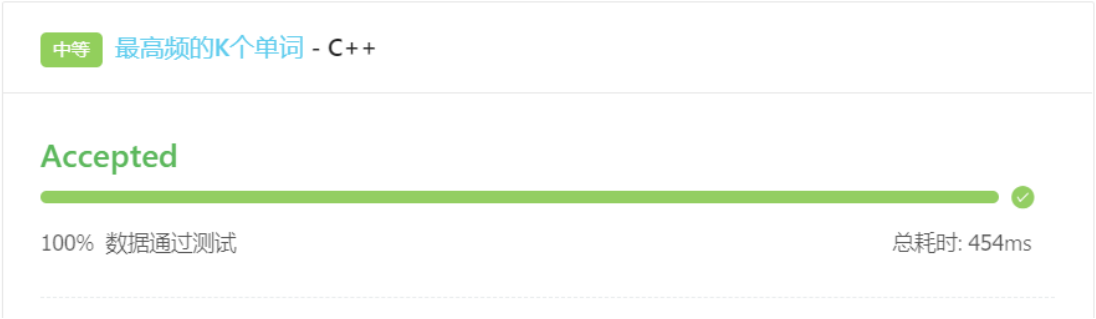
"safari", "lint", "code",

"body", "lint", "code"]

k = 3 k = 4

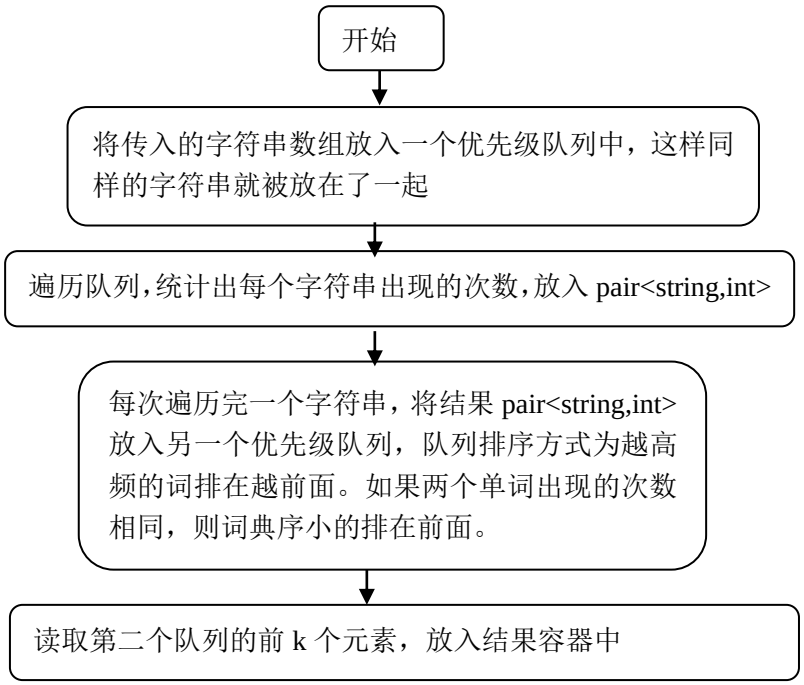
【样例输出】 k = 3, 返回 ["code", "lint", "baby"]。 k = 4, 返回 ["code", "lint", "baby", "yes"]。

【通过截图】



【实现思路】

通过优先级队列实现找出最高频的 k 个单词，思路：



【核心代码】

最高频的 k 个单词.cpp

5. 课程安排【Lintcode 616】

【问题描述】 你需要去上 n 门课才能获得 offer，这些课被标号为 0 到 n-1 。有一些课程需要“前置 课程”，比如如果你要上课程 0，你需要先学课程 1，我们用一个匹配来表示他们：[0,1]。要 求：给你课程的总数量和一些前置课程的需求，返回你为了学完所有课程所安排的学习顺序。 可能会有多个正确的顺序，你只要返回一种就可以了。如果不可能完成所有课程，返回一个 空数组。提示：拓扑排序。

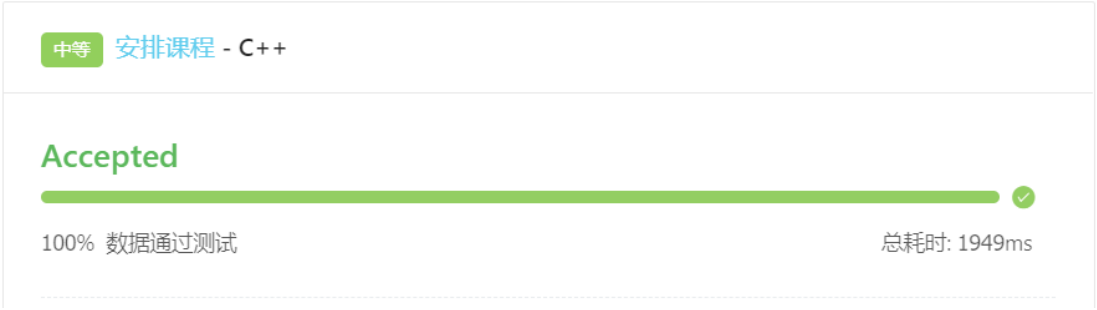
【样例输入】

n = 2, prerequisites = [[1,0]]　　n = 4, prerequisites = [1,0],[2,0],[3,1],[3,2]]

【样例输出】

返回 [0,1]　　返回 [0,1,2,3] or [0,2,1,3]

【通过截图】



【实现思路】利用拓扑排序的思路，遍历图，找出所有入度为 0 的顶点入栈，依次让这些点出栈，每次出栈，即从图中去除该点，该点指向的点入度减 1，若指向点此时入度为 0 则入栈，直至栈空为止。但是在实现时，直接遍历参数 prerequisites 时间复杂度太大，每次都会访问很多不必要的元素，所以为每个点添加辅助数组，在第一次遍历 prerequisites 时，纪录下每个点所指向的点，复杂度降低。

【核心代码】

课程安排.cpp

单词贪吃蛇软件（图搜索）

1. 需求规格说明

【问题描述】 贪吃蛇是经典的手机游戏，它通过手机键盘上下左右控制蛇的方向，寻找吃的东西。每 吃一口就能得到一定的积分，而且蛇的身子会越吃越长，身子越长难度就越大，不能碰墙， 不能咬到自己的身体，更不能咬自己的尾巴。 单词贪吃蛇（Words Snake）是一款单词学习类游戏，它通过键盘控制，不断移动蛇头 去捕食屏幕上的字母，在规定时间内完成对规定单词的拼写。目前已有 Android 和 IOS 两个 版本。如下图所示：

本题要求设计并实现一款 PC 机版本的单词贪吃蛇游戏软件。

【基本要求】 一个完整的单词贪吃蛇软件应具有以下功能：

（1）通过操作键盘上的四个方向键（也可自定义键）来控制蛇身的移动，每次蛇头吃一个“字 母”球，蛇身会增加一个该字母球。

（2）默认 5 个关卡，每一关卡初始有 4 条生命，每次过关需拼对一个单词。每次正确吃到 字母，系统自动加 100 分（屏幕右上角显示）；每得到 1000 分，可以获得 1 条生命的奖励； 每次正确拼写一个单词，进入下一步关卡。

（3）根据时间间隔（如 2 秒），每一次将贪吃蛇的每节身体分别向前移动一格，移动方向为 贪吃蛇当前行走方向。

（4）有小蛇生命数控制。当蛇身碰到外围墙壁时或者碰到自己身体时，会减少 1 条生命； 当生命数减为 0，宣告游戏结束。

（5）每次单词取自《大学英语四级词汇》，每过一关后，屏幕上随机出现一个打散的单词字 母集，字母球与蛇身不能重合。

（6）界面美观，操作方便，背景音乐播放（使用多线程技术）。

（7）最为关键：除了手动，软件还能实现 “会自动寻路的贪吃蛇” 。

【提高要求】

（1）可以设置时间间隔：时间越短，贪吃蛇移动速度越快。

（2）每个关卡可以设置生命条数、需拼对单词的个数。

【实现提示】

（1）贪吃蛇的表示：蛇身的坐标数组或者队列，头位置作插入，队尾作删除。

（2）自动寻路的贪吃蛇：可以使用简单的宽度优先搜索，也可以使用 A*寻路算法，搜索最 短路径。每一次的自动搜索，我们需要知道： ① 蛇当前所在的位置； ② 蛇的形态； ③ 每个字母的位置； ④ 每个字母吃掉的优先级顺序。

（3）游戏中快速贴图，除了 Windows 的位图操作，还可以使用 Cocos2d-X 等开源 2D 游戏引擎。

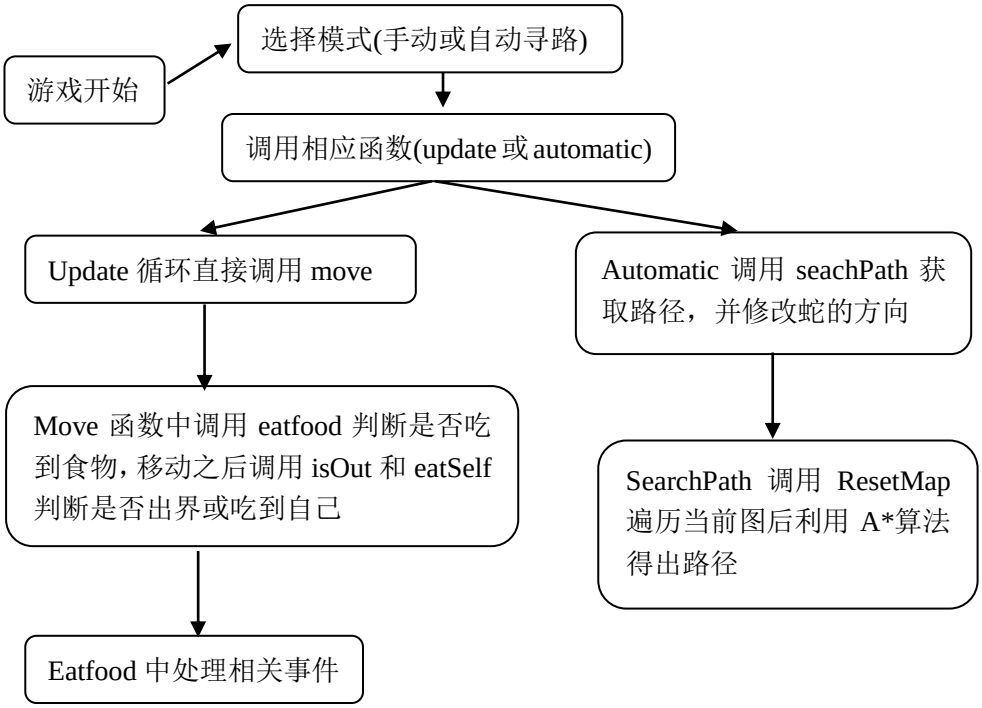
2. 总体分析与设计

（1）设计思想：

（<五号宋体>，具体内容：存储结构、主要的算法思想。）

采用 cocos2d-x 游戏引擎搭配 vs 开发，因为之前未接触过 cocos2d-x 所以基本上是边学边写的，游戏中的贪吃蛇蛇身、食物都通过 cocos2d-x 的 Sprite 类实现，自动寻路算法采用了 A*算法。

（2）设计表示：



（3）详细设计表示：

HelloWorldScene 游戏开始界面场景类：继承自 cocos2d 的 Scene 类，提供三个菜单：手动游戏、自动游戏和退出游戏，init()初始化场景，menuCloseCallback 回调函数，类成员 secondLayer 通过改变游戏类中的静态成员来控制手动或自动游戏。

GameScene 游戏类：继承自 cocos2d 的 Scene 类

GameScene 类		
函数	CreateScene	重写 CreateScene 函数
	Init	游戏初始化，在该函数内处理大量事情包括：背景音乐预加载与播放、打开文件并读取、菜单的初始化、游戏状态的初始化、蛇与食物的初始化、单词、生命与得分初始化、注册键盘监听均在该函数内完成。
	menuCloseCallback	菜单回调函数：①返回主菜单②控制音乐播放③调整游戏速度
	Draw	绘制一个矩形，游戏在该矩形内进行
	Myupdate	手动游戏主循环
	Automatic	自动游戏主循环
	creatFood	创建食物，每次创建一个食物
	onKeyPressed	键盘监听，通过按键修改蛇头方向
	Move	控制蛇身的移动
	Eatfood	判断是否吃到食物，若吃到做出相应调整
	ReadWord	读取一个单词，更新当前的单词与单词释义
	Searchpath	搜寻自动寻路的路径

	ResetMap	遍历当前图更新 map
	EatSelf	判断是否吃到自己
	IsOut	判断是否出界
成员变量	Dir	蛇头的方向
	Head	蛇身的头结点
	Food	食物
	Des	单词释义显示
	ShowWord	显示的单词显示
	Score	分数显示
	Live	生命显示
	SpeedLB	速度显示
	Path	存放自动寻路的路径
	SnakeBody	存放蛇身
	SnakeFood	存放食物
	Fin	文件流
	Word	当前单词
	Description	单词释义
	Showstr	显示的单词
	Score_str	分数字符串
	Live_str	生命数字符串
	N	蛇的长度
	Current	当前单词的第几个字母
	Map	当前游戏的状态 0 表示路，1 表示起点，2 表示障碍，3 表示终点
	live	生命数
	score	分数
	speed	速度
	autoSpeed	自动寻路速度
	isPlay	音乐是否在播放
	SpeedLevel	速度等级
	p_Res	用于修改游戏结束的分数

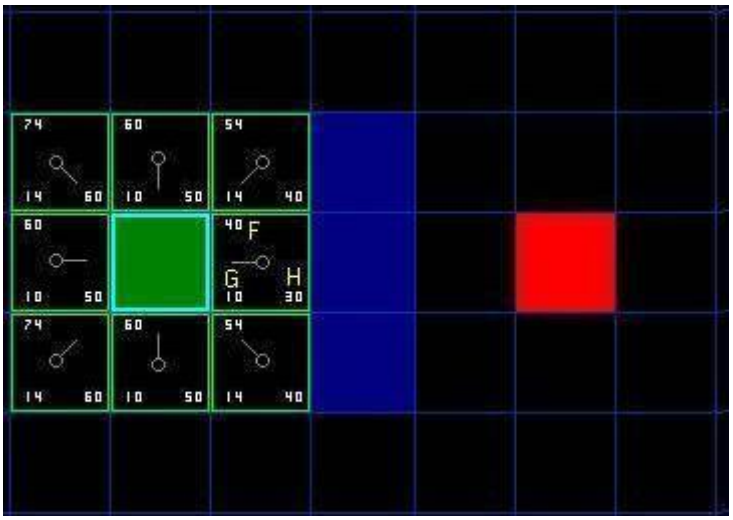
AStar 类：游戏自动寻路的主要算法，传入游戏的地图、起点和终点，返回一个路径。主要算法思想为：

（1） 搜索区域被划分为方形网格，简化搜索区域，是寻路的第一步。这一方法把搜索区域简化成了一个二维数组。数组的每一个元素是网格的一个方块，方块被标记为可通过和不可通过。路径描述为从 A 点到 B 点所经过的方块的集合。

（2） 在 A star 寻路算法中，我们通过从 A 开始，检查相邻方格的方式，向外扩展直至找到目标。首先需要两个列表：

- 一个记录所有被考虑来寻找最短路径的网格集合（open list）
- 一个记录下不会被考虑的网格集合（closed list）

（3） 首先在 closed 列表中添加当前位置 A，然后把所有和它当前位置相邻的可通行网格添加到 open 列表中。如何选择哪个网格是最短路径呢？在 A star 算法中，通过给每个网格赋值权重，被称为路径增量。



（4） 我们将会给每个网格一个 G+H 的权重值：

G 是从开始点 A 到当前方块的移动量。所以从开始点 A 到相邻小方块的移动量为 1，该值会随着离开始点越来越远而增大。

H 是从当前方块到目标点 B 的移动量估算值。该值经常被称为启发式的，因为我们不确定移动量是多少 – 仅仅是一个估算值。

（5） 算法开始：

① 把起始格添加到开启列表。

② 重复如下的工作:

a. 寻找开启列表中 F 值最低的格子。我们称它为当前格。

b. 把它切换到关闭列表。

c. 对相邻的 8 格中的每一个?

- 如果它不可通过或者已经在关闭列表中，略过它。反之如下。
- 如果它不在开启列表中，把它添加进去。把当前格作为这一格的父节点。记录这一格的 F ,G,和 H 值。
- 如果它已经在开启列表中,用 G 值为参考检查新的路径是否更好。更低的 G 值意味着更好的路径。如果是这样，就把这-格的父节点改成当前格，并且重新计算这一格的 G 和 F 值。如果你保持你的开启列表按 F 值排序，改变之后你可能需要重新对开启列表排序。

d. 停止，当你

- 把目标格添加进了关闭列表(注解)，这时候路径被找到，或者
- 没有找到目标格，开启列表已经空了。这时候，路径不存在。

③ 保存路径。从目标格开始，沿着每格的父节点移动直到回到起始格。这就是你的路径。

（参考：https://www.jianshu.com/p/bb317d02d055）

Result 类：用于显示游戏结束时的分数。

3. 编码

（1） 用 python 脚本生成的文件的 sdk 均为 8.1，与电脑上的库不符，报大量错误，查阅资料后解决；

（2） 编译出的程序无法运行，提示“应用程序无法正常启动 0xc000007b”，查阅资料后修复了 DirectX 安装了缺少的 VC++运行库后解决；

（3） 初学 Cocos2d-x，发现这个引擎里用的坐标和平时用的不一样，Cocos2d 中用的是逻辑坐标，而不是像素坐标，转化起来相当麻烦，所以直接改了窗口大小直至 contentScaleFactor 为 1；

（4） 场景之间传参：在要传入参数的场景类中加入静态变量，在要传参数的类中加入第二个类的指针，用指针修改静态变量；

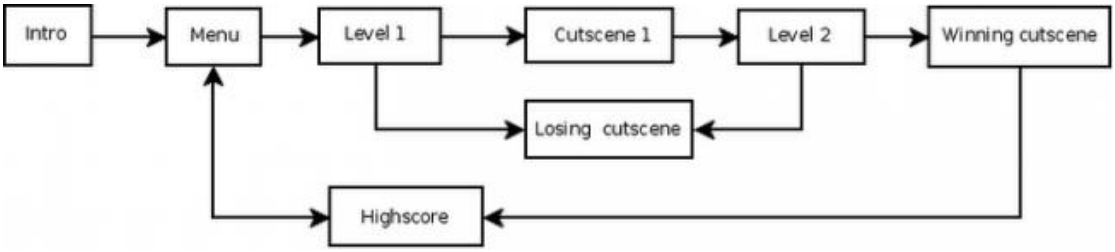
4．程序及算法分析

使用说明：程序运行后，点 Start Game 即可开始手动游戏，Automatic Path Finding 开始自动寻路游戏，Exit Game 退出游戏，游戏界面内，Score 表示分数，Live 表示生命数，点 Speed 可以修改游戏速度，音量键可以打开/关闭声音，Menu 返回主菜单。

游戏自动寻路时每次吃掉食物后会有卡顿，时空复杂度还是蛮大的 。游戏有 bug，生成的食物可能在蛇身上，之后争取改进。

5．小结（必填）

通过写这个贪吃蛇小游戏，基本初步的了解了一下 Cocos2d-x 的概念，主要是 Cocos2d-x 使用导演的概念，这个导演和电影制作过程中的导演一样！导演控制电影制作流程，指导团队完成各项任务。在使用 Cocos2d-x 开发游戏的过程中，你可以认为自己是执行制片人，告诉 导演(Director) 该怎么办！一个常见的 Director 任务是控制场景替换和转换。 Director 是一个共享的单例对象，可以在代码中的任何地方调用。然后就是场景和精灵的概念，以前好像有层的概念，现在已经被弱化了，基本上就是一个场景中一个层，一个层上对应多个精灵。



在自动寻路算法部分，可能 A*算法并不是一个全解的算法，无法吃满整个屏幕。有时会无故报错。

6．附录

见附件

英语四级电子辞典的实现

1．需求规格说明

【问题描述】

电子辞典是一种将传统的印刷词典转成数码方式、进行快速查询的学习工具，成为 21 世纪学生学习生活的掌上利器。

在很多实际应用中，动态索引结构在文件创建或初始装入记录时生成，在系统运行过程中插入或删除记录时，为了保持较好的检索性能，索引结构本身将随之发生改变。教材上已经介绍的动态查找数据结构包括：二叉搜索树（BST）、平衡二叉树（AVL）、红黑树（RBT） 、B-树。

本题要求选取一种已经学过的动态搜索树结构，设计并实现一个基于英语四级词汇的电

子辞典软件。

【基本要求】 一个完整的电子辞典软件应具有以下功能：

（1）支持词库数据的导入、导出操作，外部数据采用 TXT 格式。注意：文本文件打开一次 后，下次不用再装载。

（2）支持英文单词的添加、修改、删除等功能；

① 添加操作要防止重复输入；

② 修改或删除某个单词的信息，在操作前，需先进行查找定位。

（3）支持英文单词的各种查询操作，具体包括：

① 逐条翻看　　能按照英文单词的顺序，显示所有的单词记录，支持分页滚动查看。

② 单词查找　　输入英文单词，给出其汉语意思。要求支持两种方式：精确查找（按单词查）和模 糊查找（按前几个字母查）。

③ 单词记忆　　最近浏览过的 10 个单词，允许回退/前进进行浏览。

（4）要求使用 BST 或者 AVL 实现动态索引结构。

（5）用户界面友好，考虑输入的容错性。

【提高要求】

（1）支持“根据中文反查英文单词”的功能。提示：可考虑由汉字机内码构造英文哈希表；

（2）背单词功能：支持“单词填空”的简单背单词软件功能。提示：可参考的软件包括：新 东方背单词：（http://www.neworiental.org/）；轻轻松松背单词（http://www.pgy.com.cn/）等。

（3）支持英文发音功能，读出英文单词。提示：可使用微软的 TTS（Text to Speech）开发 包来完成，TTS 发音是利用 CPU 将任意组合的文本件转化为声音文件。

（4）使用红黑树或者 B-树的数据结构，来实现动态索引结构。

【测试数据】 《2010 年大学英语四级词汇.txt》、《英语四级高频词汇 700 个》

【实现提示】

（1）设计合适的电子辞典数据文件格式；

（2）设计合适的索引文件格式；

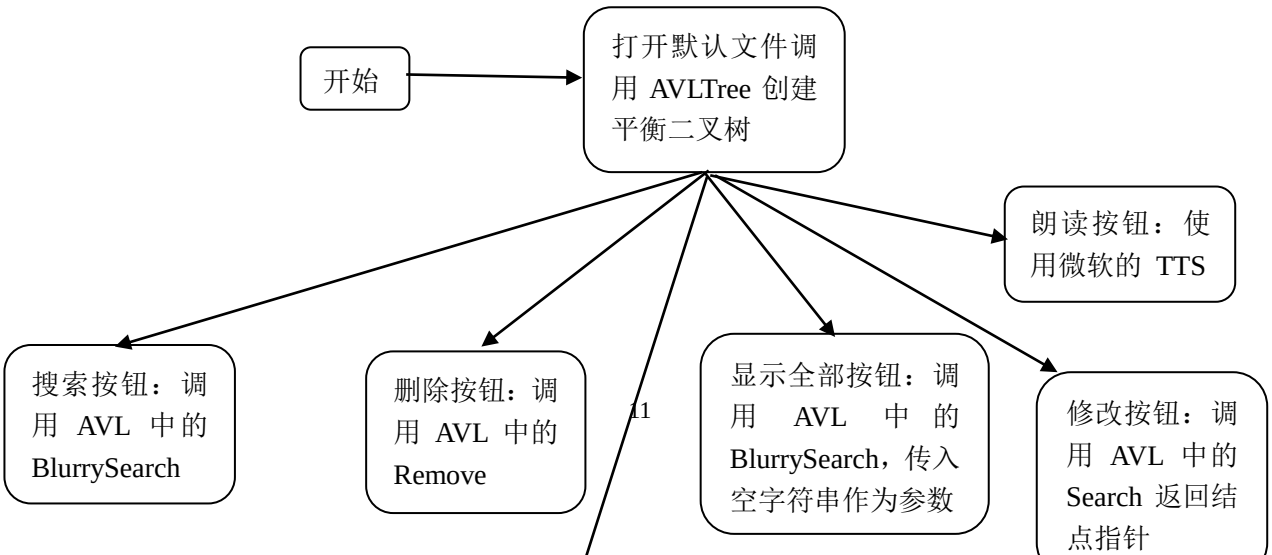
（3）可能用到的 MFC 控件包括：CListBox、CComboBox、CListCtrl 等，支持滚动条。

2．总体分析与设计

（1）设计思想：

本程序采用 Qt 开发，采用平衡二叉树来存储每个单词结点，平衡二叉树基于书上的代码扩充，单词的列表显示用了 qt 的 QlistWidget 控件。

（2）设计表示：



添加按钮：调用 AVL 中的 Insert

（3）详细设计表示：

WordNode：表示单词的结构体，其中含有 key（单词），description（单词的释义），以及一些重载函数和构造函数；

AVLNode：树的结点的结构体，其中含有 bf（平衡度），data（一个单词），以及构造函数；

这里主要使用的算法就是关于 AVL 树的算法了，其中的核心算法（创建、调整）书上已经讲的很清楚了，主要说一下扩充的部分：

blurrySearch（模糊搜索）：添加一个新的字符串函数 isContain，若两个单词的前 n 个字母都一样（n 为短字符串的长度），则认为包含，搜索时，用 AVL 的搜索方法搜到第一个包含搜索字符串的单词，然后在以该单词为根节点的树下中序遍历搜索。

Export：传入一个 qt 的文件流，采用其中的数据建立 AVL 树。

3. 编码

只是在书上的代码进行扩充，在建树上未遇到麻烦。在写模糊搜索时，遇到了顺序显示不正确，单词重复显示问题，改写算法后解决；然后就是在每次搜索单词时，第二次搜索清楚 listWidget 时程序终止，经查阅是要解绑 listWidget 上的槽函数，解决。

4. 程序及算法分析

程序运行会自动打开一个默认文件，也可使用菜单导入，导入文件格式为：每个单词两行，第一行为该单词，第二行为该单词的释义，菜单中也可以导出；其他操作不一一赘述。

5. 小结（必填）

通过本程序的编写，让我更加深入的了解了 AVL 树的构造、搜索以及其他算法，也体验了一下模糊搜索的概念。

可以改进之处：未能实现“根据中文反查英文单词”的功能，争取之后加入；在删除单词后，历史纪录中还可以访问。

6. 附录

见附件

图分割：待完成

【参考资料】

1、 Sartaj Sahni 著，《数据结构、算法与应用》，机械工业出版社

2、 殷人昆 等编著，数据结构(用面向对象方法与 C++语言描述)》，清华大学出版社

3、 严蔚敏，吴伟民 编著，《数据结构题集》，清华大学出版社

4、 严蔚敏，陈文博 编著，《数据结构及应用算法》，清华大学出版社

另外，希望在报告后给出对数据结构课程、课程实习的建议，便于改进。