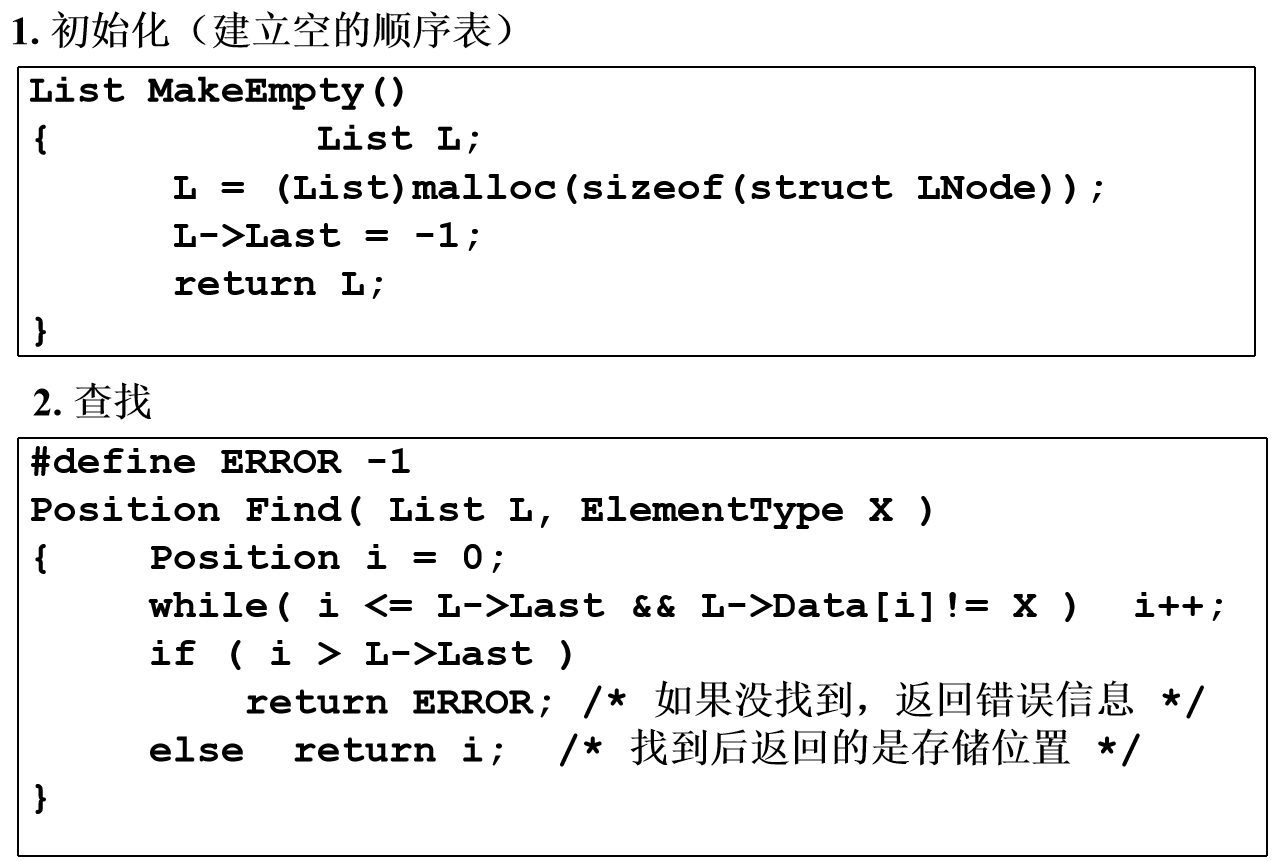
1. 绪论
2. 熟悉各名词、术语的含义，掌握基本概念。

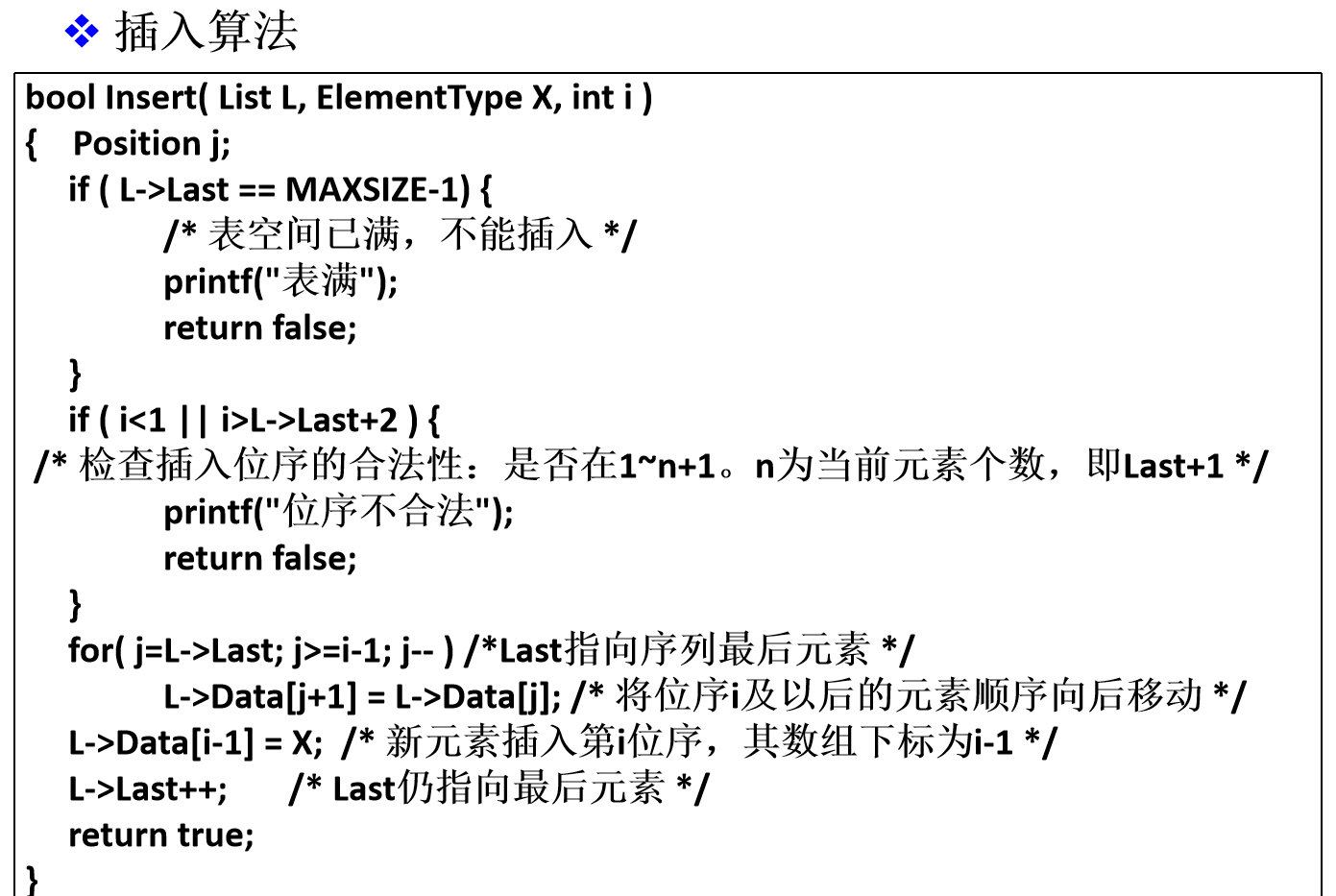
数据，数据元素，数据对象，数据结构，逻辑结果，存储结构，数据类型，抽象数据类型

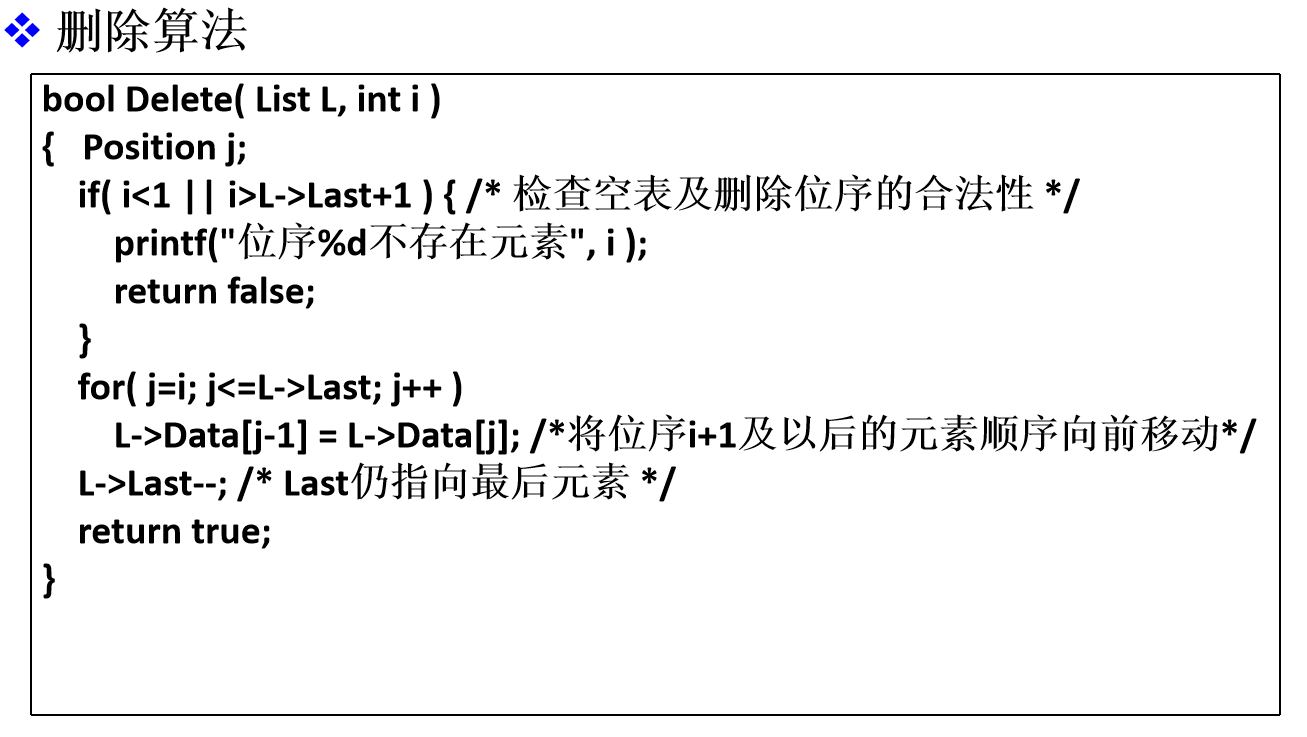
1. 理解算法五个要素的确切含义。
2. 掌握计算语句频度和估算算法时间复杂度的方法。
3. 课后习题
4. 线性表

1.线性表的定义（逻辑结构及其特点）

2.熟练掌握线性表顺序存储结构的描述方法，以及线性表的各种操作的实现。（PPT上的算法）

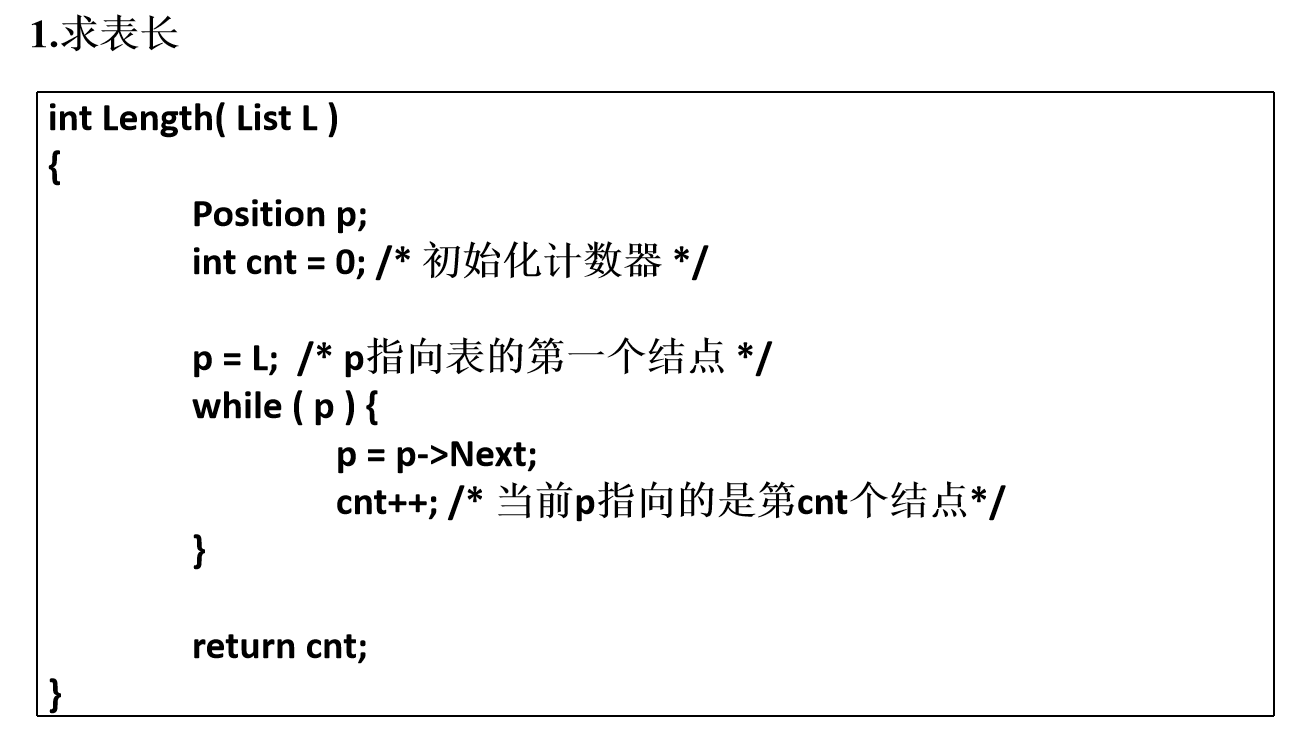


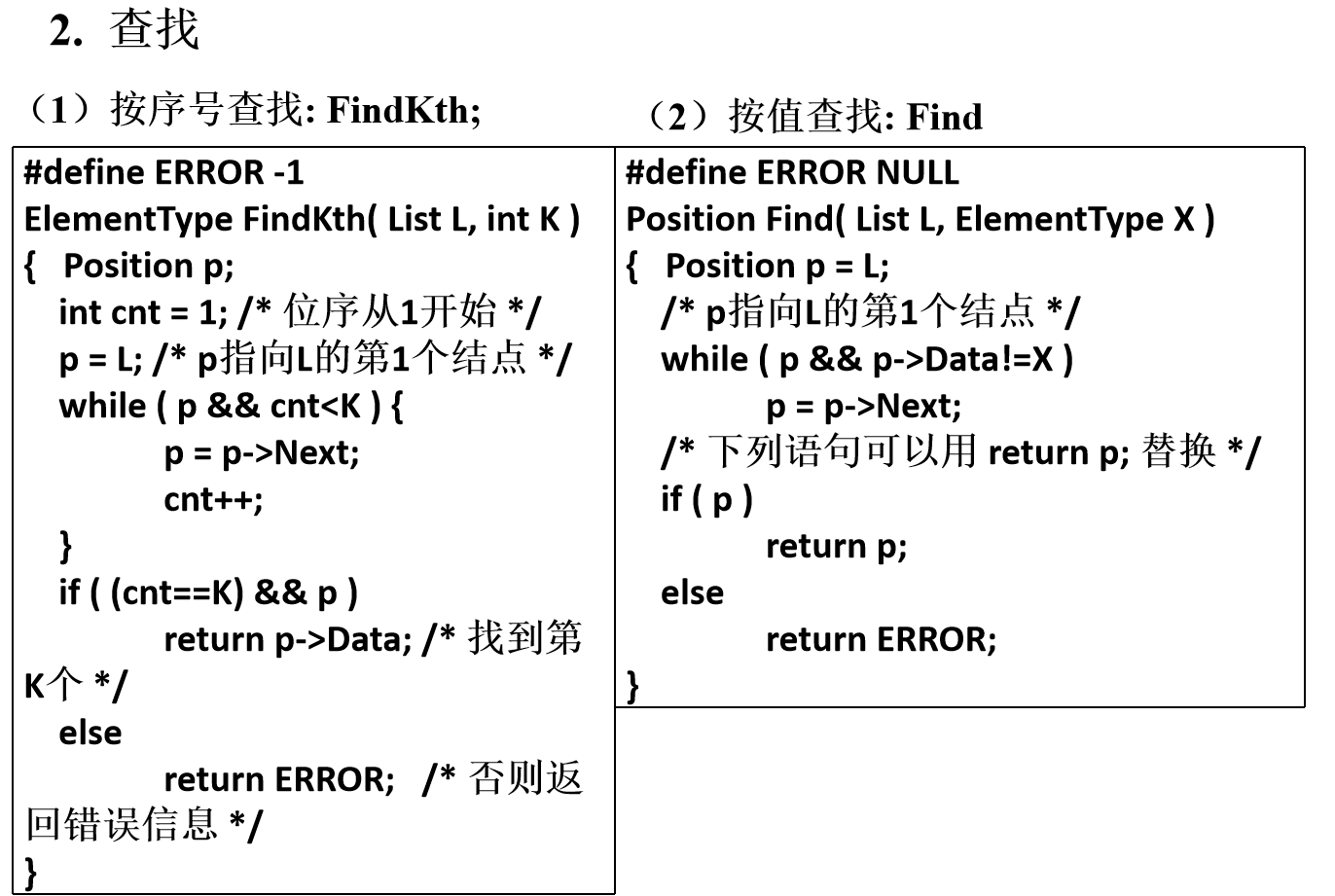


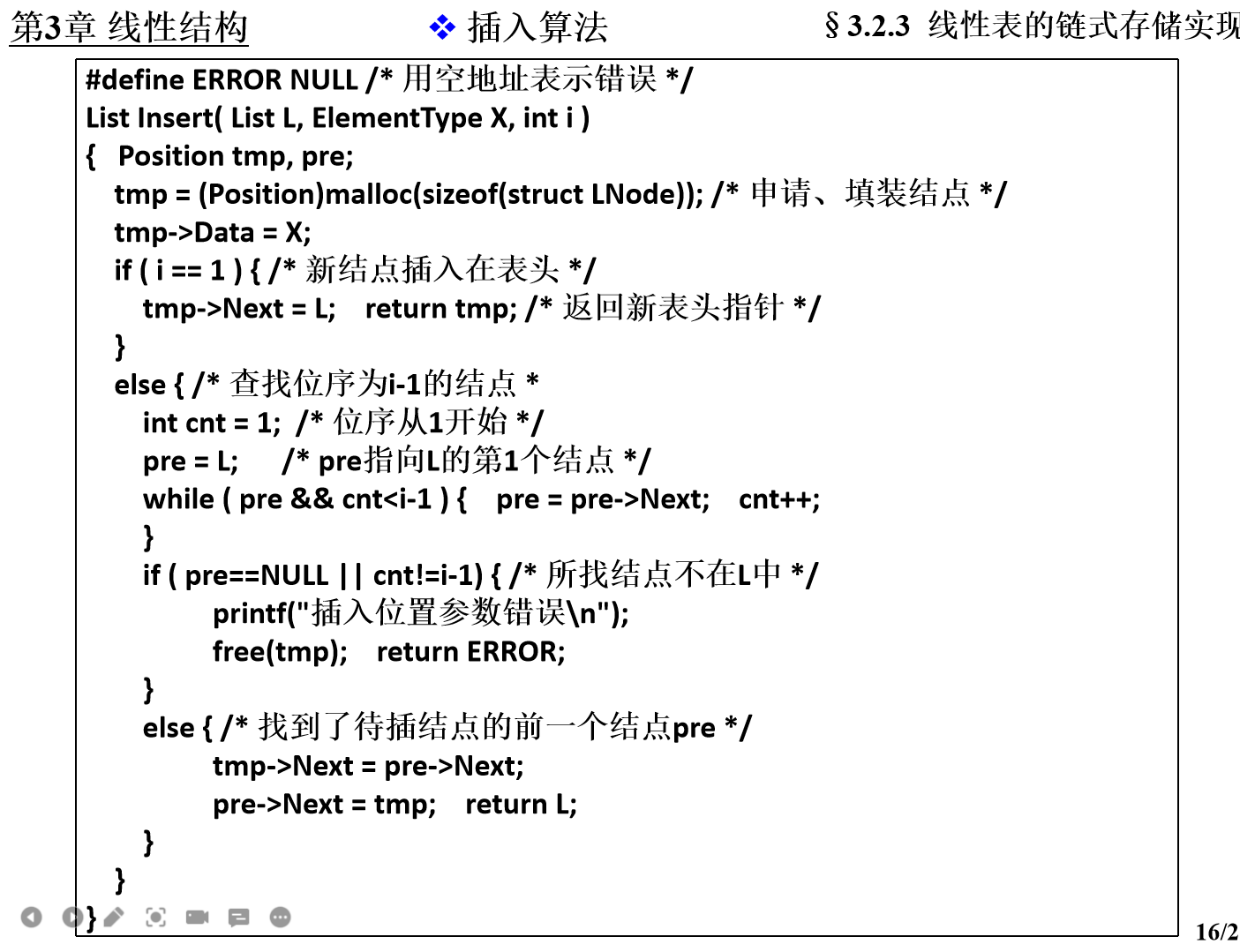


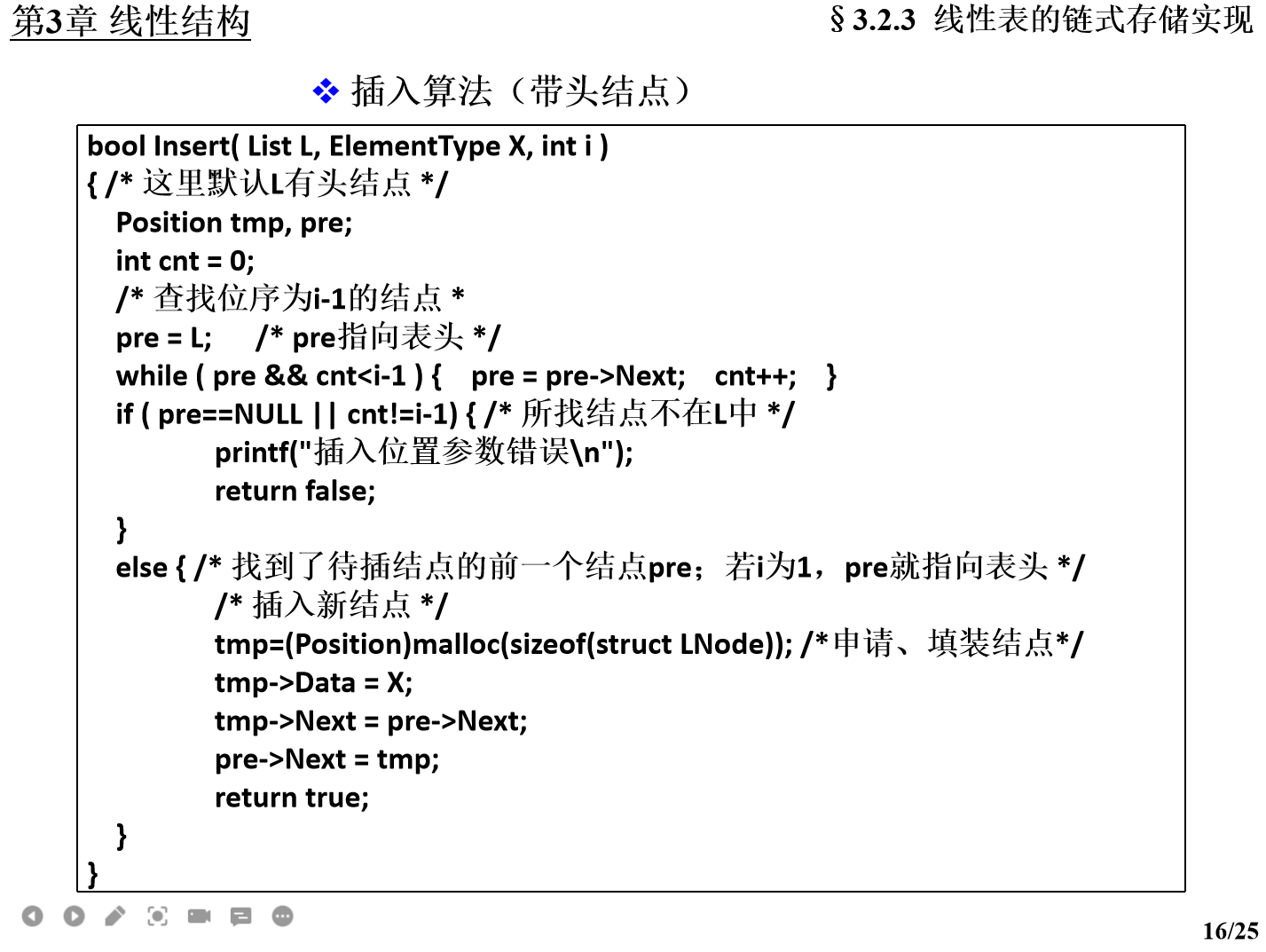
3.从时间复杂度分析顺序存储结构的特点及适用场合。（记住插入、删除时间复杂度，优势）

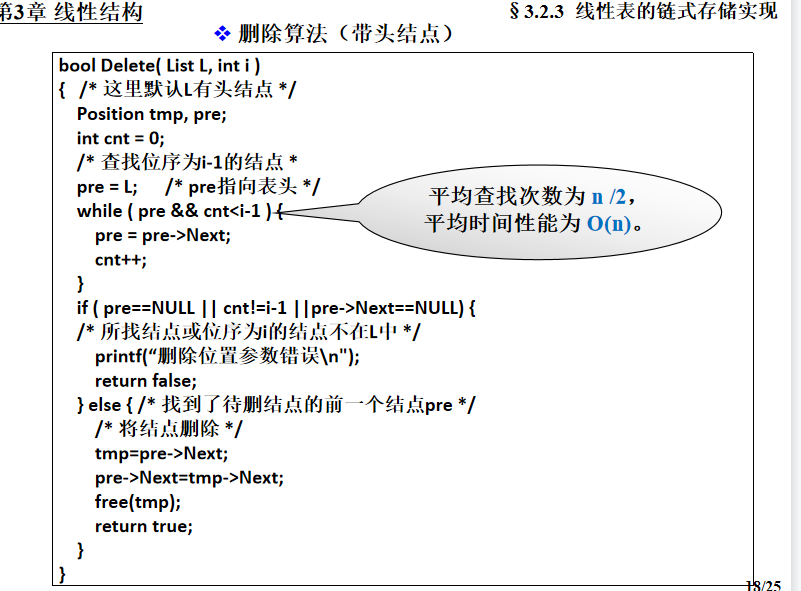
4.熟练掌握线性表链式存储结构的描述方法，以及线性表的各种操作的实现。（PPT上的算法）











5.从时间复杂度分析链式存储结构的特点及适用场合。（优点和缺点）

6.单链表、循环链表、双链表的插入、删除、查找的实现语句。

7.课后习题的算法

1. 栈和队列

3.1栈（定义及特点，课后习题，讨论题，顺序栈和链栈基本操作函数的实现）

3.2栈的应用举例（阶乘，括号匹配，表达式求值等的算法）

3.3栈与递归的实现（函数调用的过程，递归与栈之间的关系）

3.4队列（定义及特点，课后习题，讨论题，队列的顺序存储的实现方法）

第四章 串

4.1串的类型定义

1. 数组和广义表

5.1数组的定义（了解）

5.2数组的顺序表示和实现（根据下标计算出存储的位置，低下表优先，高下标优先）

5.3数组的压缩存储（三元组的定义及存储的方式）（不要求算法）

5.4广义表的定义（熟练掌握定义，表的长度，表的深度，空表等）

5.5广义表的存储结构（表头、表尾表示法，取表头，取表尾，存储结构的描述）

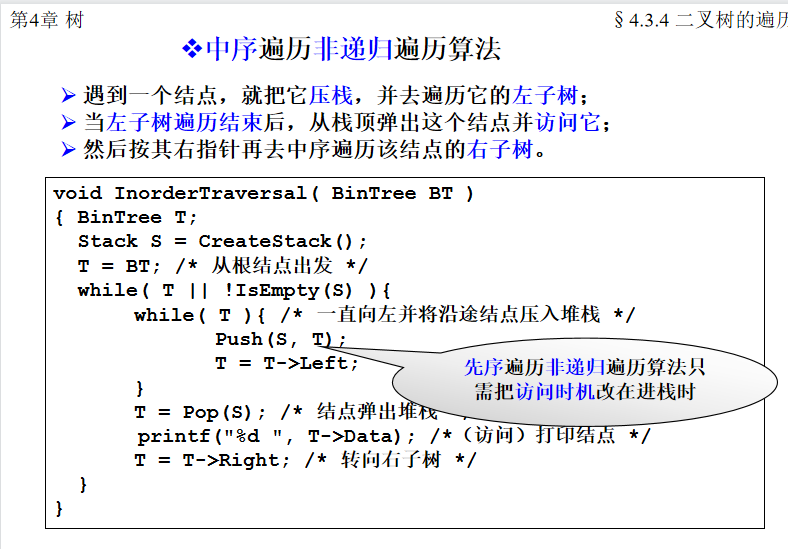
5.6课后习题

1. 树和二叉树

6.1树的定义和基本术语（结点之间的关系，父子，兄弟，树的度，度的深度等，参考PPT）

6.2二叉树（5种形态，二叉树的性质）

6.3遍历二叉树和线索二叉树（各种遍历，遍历算法，中序遍历的非递归算法，线索树的定义及会画出一棵线索树）



6.4树和森林（树和森林的遍历，树与二叉树之间的转换）

6.6哈夫曼树及其应用（给出条件会建立一棵哈夫曼树，并完成编码，哈夫曼树的性质，基本概念）

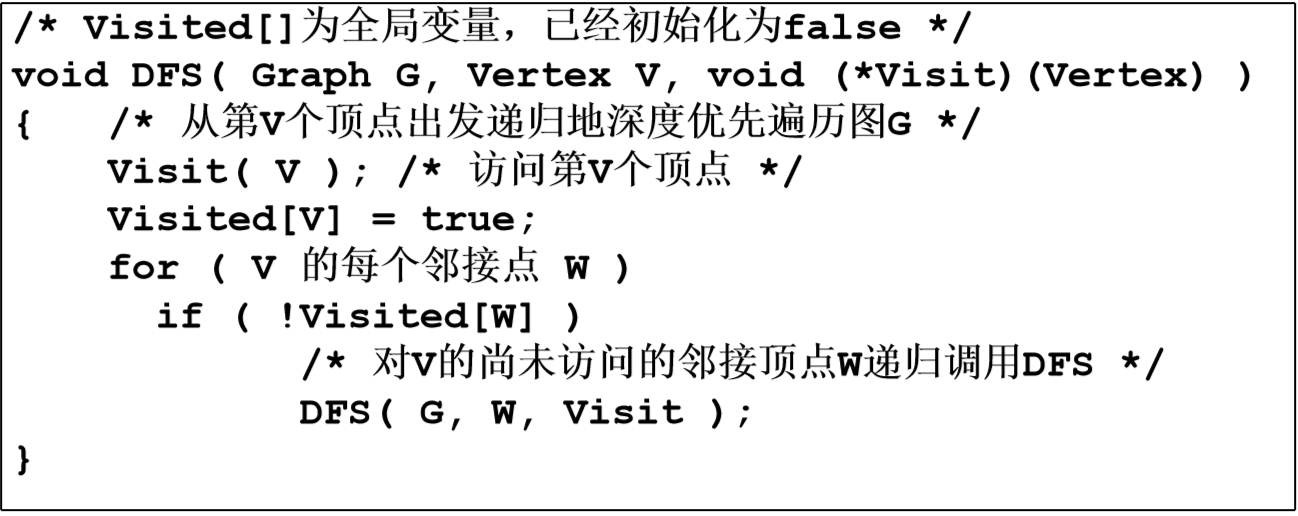
这章的课后习题

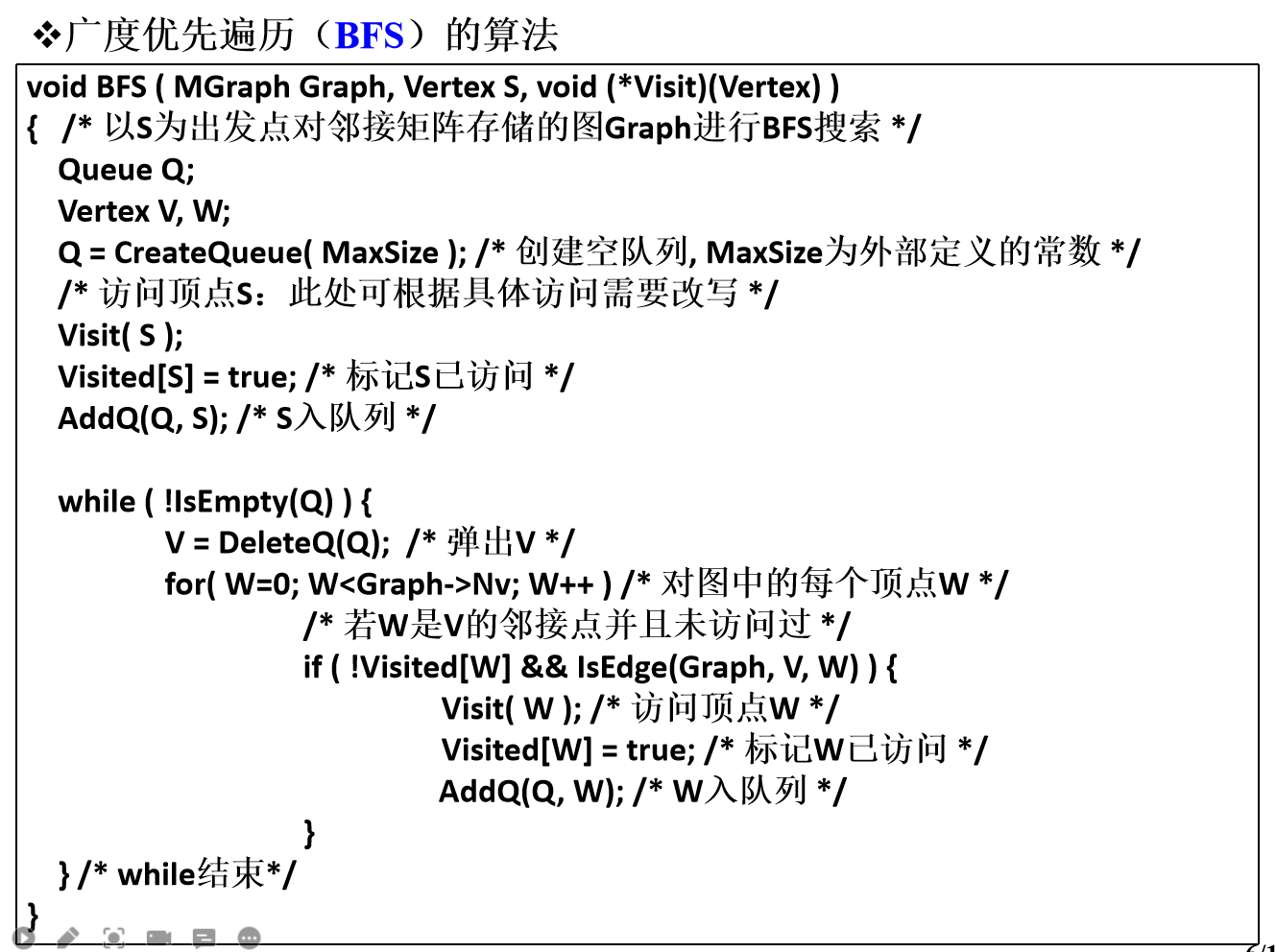
1. 图

7.1图的定义和术语（参考ＰＰＴ）

7.2图的存储结构（给出图，能够正确画出两种存储结构）

7.3 图的遍历（给出存储结构，能够得出两种遍历的结果，每种遍历应用的是栈还是队列，图的遍历算法）





7.4图的连通性问题（连通图、强连通图的定义，最小生成树的定义，按照普里姆算法生成最小生成树的过程，按照克鲁斯卡尔算法生成最小生成树的过程）

7.5有向无环图及其应用 （拓扑排序的定义，给出一个图能够得出拓扑排序的结果，拓扑排序的算法，关键路径的求解方法，最早开始时间，最迟开始时间的求解，关键活动）

Status TopologicalSort ( ALGraph G ) {

//有向图采用邻接表存储结构

//若G无回路，则输出G的顶点的一个拓扑序列并返回OK，否则返回ERROR

FindInDegree( G, indegree );

InitStack(S);

for ( i = 0; i < G.vexnum; ++i )

if ( !indegree[i] )

push(S,i); //入度为0进栈

count = 0; //对输出顶点计数

while(!StackEmpty(S)){

pop(S,i);

printf(“%d%d”, i, G.vertices[i].data);

++count;

for(p=G.vertices[i].firstarc; p; p=p->nextarc){

k=p->adjvex;

if(!(--degree[k]))

push(S,k);

}

}

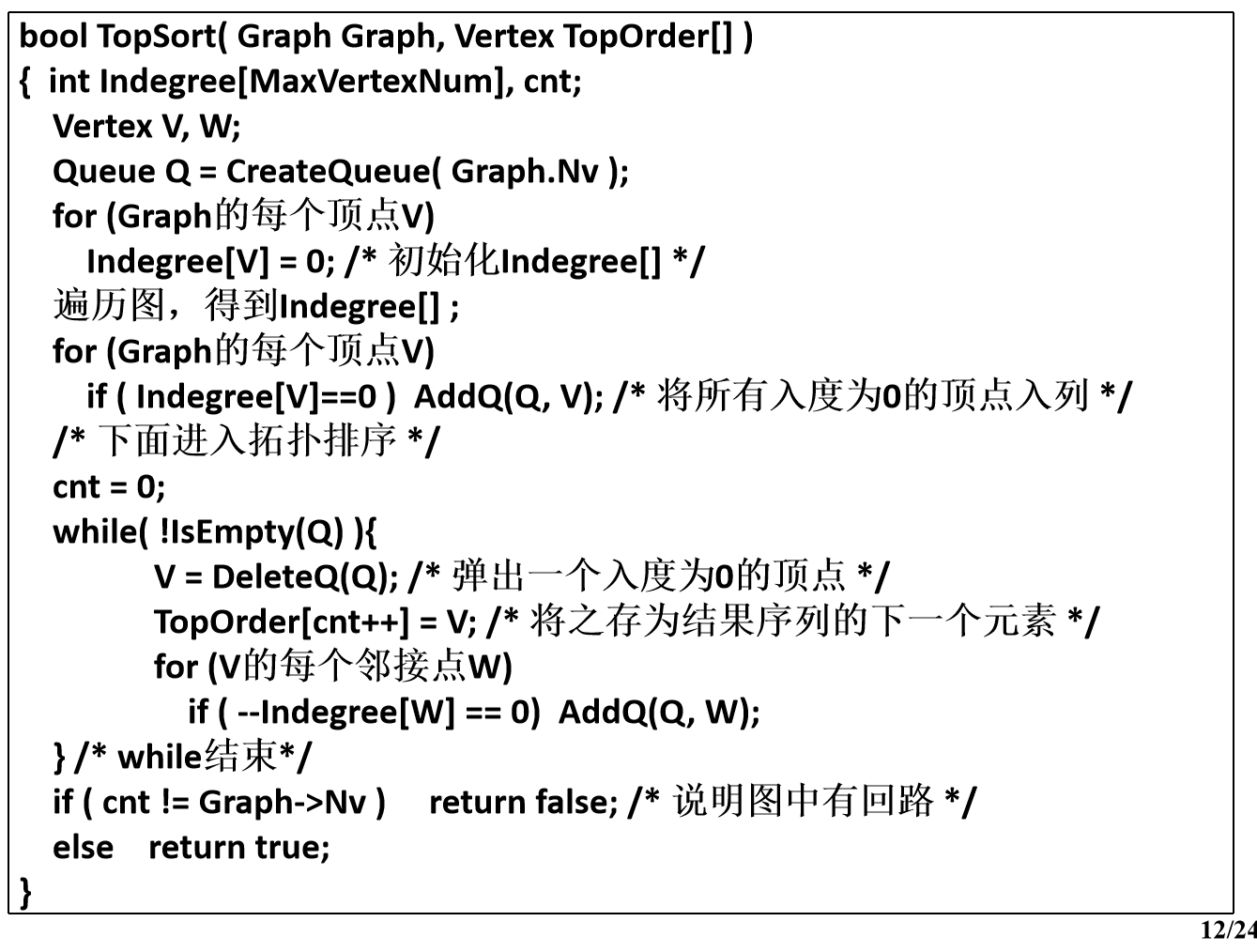
if(count < G.vexnum )

return ERROR;

else

return OK;

}



7.6最短路径（掌握迪杰斯特拉算法的求解过程以及算法，按照书上表格可以，按照ＰＰＴ上的过程也可以）

void Dijkstra( MGraph G, int v0, PathMatrix &P, ShortPathTable &D ) {

//用迪杰斯特拉算法求有向网G的v0顶点到其余顶点v的最短路径P[v]及其带权路径长度D[v]

//用p[v][w]为true，则w是从v0到v当前求得最短路径上的点

//final[v]为true当且仅当已求得从v0到v的最短路径

for( v = 0; v < G.vexnum; ++v){

final[v] = FALSE;

D[v] = G.arcs[v0][v];

for(w = 0; w < G.vexnum; ++w)

P[v][w] = FALSE;

if(D[v] < INFINITY){

P[v][v0] = TRUE;

P[v][v] = TRUE:

}

}

D[v0] = 0;

final[v0] = TRUE;

//开始主循环，每次求得v0到某个顶点v的最短路径，并加v到S集

for( i = 1; i < G.vexnum; ++i ){ //其余G.vexnum-1个顶点

min = INFINITY;

for( w = 0; w < G.vexnum; ++w ){

if( !final[w] )

if(D[w] < min)

{ v = w;

min = D[w];

}

}

final[v] = TRUE;

for( w = 0; w < G.vexnum; ++w){

if(!final[w] && (min + G.arcs[v][w] < D[w])){

D[w] min + G.arc[v][w];

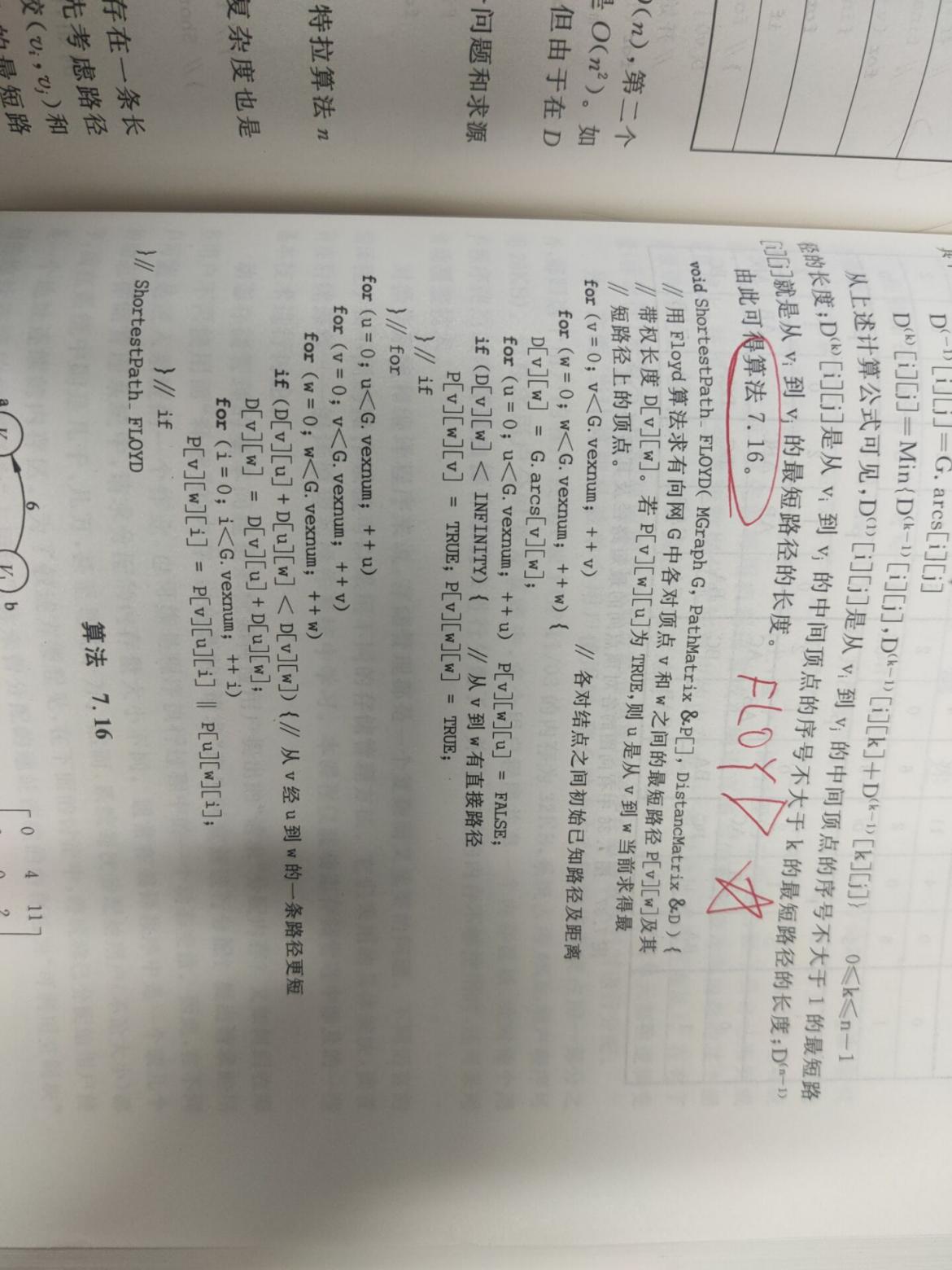
P[w] = P[v];

P[w][w] = TRUE;

}

}

}



课后习题

1. 查找

9.1 静态查找表（顺序查找的算法，监视哨的作用，折半查找）

int Search\_Seq (SSTable ST, KeyTable key){

ST.elem[0].key = key;

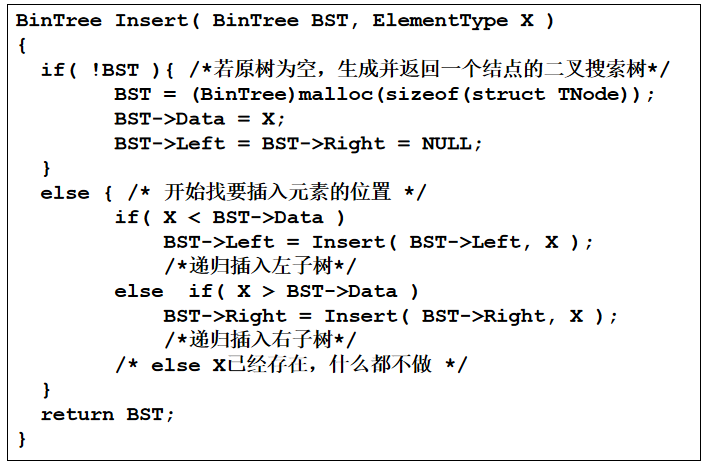
for(i = ST.length; ST.elem[i].key != key; --i);

return i;

}

9.2动态查找表

9.2.2二叉排序树（定义，性质，插入过程及算法，删除过程，删除不要求掌握算法）



9.2.3平衡二叉树（定义，判断一棵树是否是平衡二叉树，如果不平衡，如何调整，不掌握算法）

9.2.4 B\_树（定义，性质，插入和删除过程，不要求掌握算法）

9.3哈希表（定义，特点，复杂度与什么相关，处理冲突的方法，除留余数法建立查找表）

课后习题

1. 排序

10.1概述（排序的基本概念，稳定性）

10.2插入排序（直接插入排序，希尔排序，排序过程，不要求算法）

10.3 快速排序（起泡排序，快速排序（掌握算法））

int Partition ( SqList &L, int low, int high){

L.r[0] = l.r[low];

while( low < high ) {

while(low < high && L.r[high].key > pivotkey )

--high;

L.[low] = L.[high];

while(low < high && L.r[low] < pivotkey )

++low;

L.r[high] = L.r[low];

}

L.r[low] = L.r[0];

return low;

}

void QSort(SqList &L, int low, int high){

if(low < high)

mid = QSort(L,low,high);

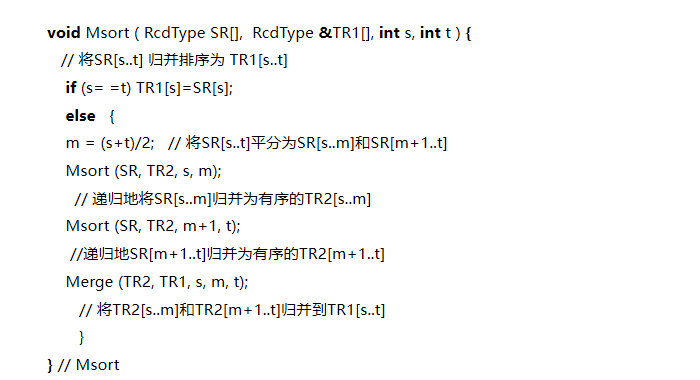
QSort(L,low,mid - 1);

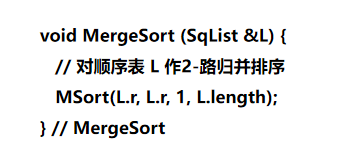
QSort(L,mid + 1, high);

}

10.4 选择排序（简单选择排序，堆排序，不要求算法）

10.5 归并排序（归并排序的过程，算法结合第２章理解）





10.6基数排序（基数排序，链式基数排序的过程，不要求算法）

10.7各种排序方法讨论（很重要，参照PPT）

课后习题