考试题型

- 单项选择：5题，共10分
- 填空：5题，共10分
- 简答题：5题，共45分
- 算法阅读：15分
- 算法设计：20分

考试要求：闭卷

第1章 概论

- DS描述“按一定逻辑关系组织的数据元素的表达及有关操作
- 数据逻辑构造：线性构造、树形构造和图形构造
- 数据存储构造：顺序措施、链接措施、索引措施、散列措施
- 抽象数据类型ADT
- 算法4个特征：通用性、有效性、拟定性、有穷性
- 算法分析：T(n)、S(n)算法分析的有关概念；最差、最佳与平均情况的意义

ADT的定义

- 三元组表达　ADT=(D, R, P)
- ADT 抽象数据类型名 {
  - 数据对象D:<数据对象的定义>
  - 数据关系R :<数据关系的定义>
  - 基本操作P:<基本操作的定义>
  - } ADT 抽象数据类型名
- 用C++类模板的申明表达ADT

数据的抽象

算法的抽象

ADT定义类模板

- 类模板代表一类类，不代表详细的类
- 类模板的定义格式：

```
template<class Type> //类型参数Type,使用时用详细数据类型替代
class className{
    private:
        Type dataList;
        …
    public:
        methodName( );
        …
};
```

- C++引入模板概念，是想突出数据的逻辑构造而忽视其详细的数据类型

申明、定义和使用 C++类模板(2)

类模板：描述了一组有关的类，它们只能经过类型来区别

1、类模板申明：仅提供了类的名称和类的模板参数

```
template <class Elem> //类模板 Array 可接受任何类型作为参数
class Array { Elem* data;
              int size;              //申明类模板Array的类数据
              public: Array( int sz ) ;        //函数组员
              int GetSize( ) ;              };
```

2、函数组员定义

```
template <class Elem> Array<Elem>::Array( int sz ) {
                        size = sz; data = new Elem[size];}
template <class Elem> int Array<Elem>::GetSize( ){
                        return size; }
```

3、类模板的使用方法

```
Array<int> int_array(100); //Array接受int作参数,
                    //int_array为长100的int型数组对象
```

常见上限g(n)的种类(用于比较各算法优劣)

- $O(1)<O(\log n)<O(n)<O(n\log n)<O(n^2)$

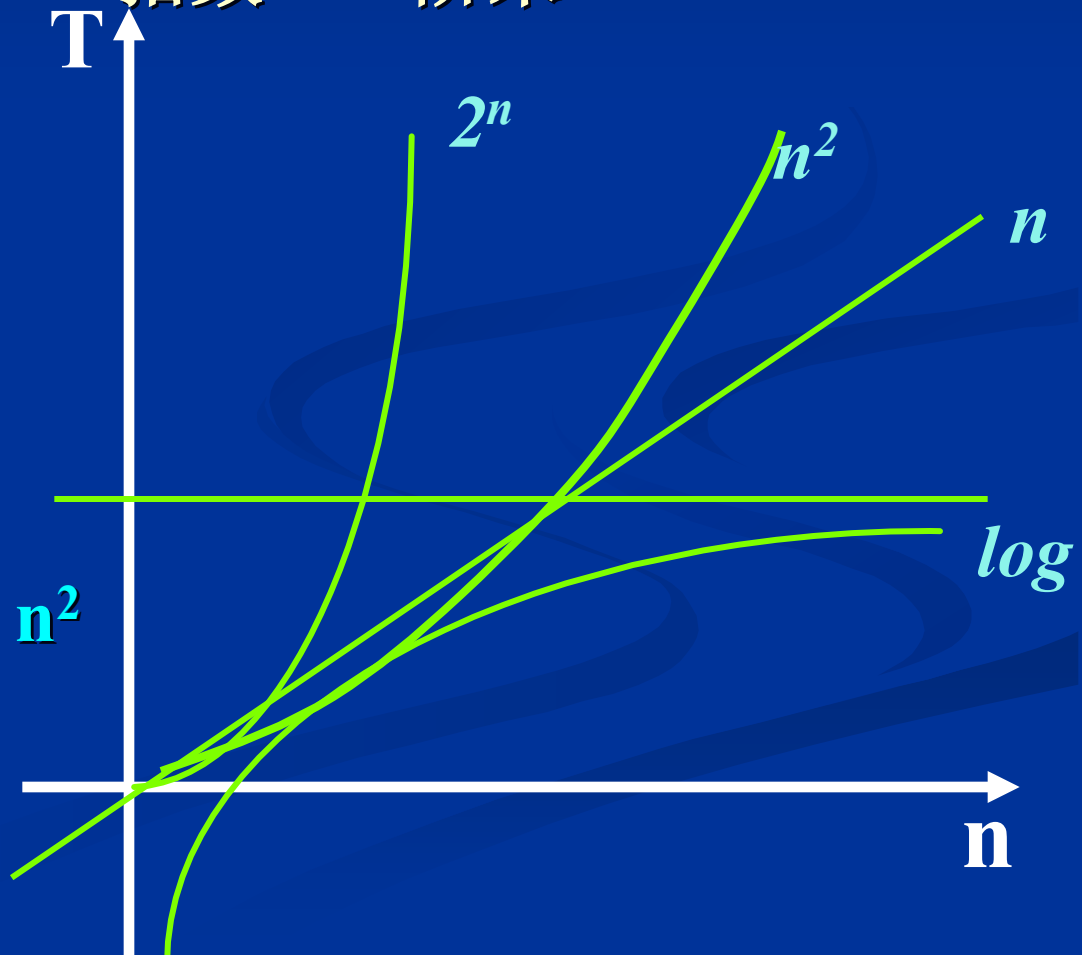
  常数阶　对数　　线性　对数乘积　　平方

  $<O(n^3)<\ldots.<O(2^n)<O(n!)$

  立方　　指数　阶乘

- 常数：g(n) = 1
- 对数：g(n) = logn
- 线性增长：g(n) = n
- 二阶增长：g(n) = n^2
- g(n) = nlog(n),

  n <= nlog(n)增长率 <= n^2
- 指数增长：g(n) = a^n



题型举例

算法指的是（　　）。

A. 计算机程序　　B. 处理问题的计算措施

C. 排序算法　　D. 处理问题的有限运算序列

将长度为 n 的单链表接在长度为 m 的单链表之后的算法时间复杂度为（　　）。

- A. O(n)　　B. O(1)

第2章　线性表

- 清楚线性表定义、了解类定义及抽象数据类型中定义的各个基本操作含义
- 存储形式：顺序存储构造、链式存储构造
- 顺序存储构造的特点：

  1. 逻辑构造与物理构造一致；
  2. 属于随机存取方式

  缺陷：插入\删除元素需要移动，平均约二分之一的元素，限制了长度变化
- 链式存储构造的特点：

  1. 逻辑构造与物理构造不一致；
  2. 属于顺序存取方式

2.1.2 线性表的存储构造

- 逻辑构造→存储空间 的映射
  - 数据→存储单元 建立映射
  - 关系→存储单元之间关系 建立映射
- 线性表2类存储构造

1. 顺序存储（定长的一维数组构造、向量型顺序存储构造）
   - 为整个元素动态分配连续空间
   - 特点：逻辑相邻物理也相邻
2. 链式存储（变长的线性表存储构造）
   - 按需分配(插入:分配一种结点/删除:回收一结点)
   - 特点：逻辑相邻物理不一定相邻

链表—单向、循环、双向

- 不特殊阐明，均带头结点(防止对空表的特殊处理)
- 算法：（时间复杂性）
  - ✓在有序表中插入/删除结点
  - ✓给定元素位置，插入/删除相应结点

  注意：
  - ✓对循环链表操作时，尾部的判断
  - ✓双向链表的插入/删除结点
  - ✓删除结点一定要释放空间

2.4 线性表实现措施的比较

- 顺序表优点
  - 存储紧凑（逻辑构造由存储相对位置体现，没使用指针，不用花费附加开销 ）
  - 随机访问（给定下标, 常量时间可定位）
- 链表优点
  - 不限定长度（无需事先了解线性表的长度，允许线性表的长度有很大变化 ）
  - 不必移动，仅需改指针即可插删（能够适应经常插入删除内部元素的情况 ）
- 合用
  - 不能拟定长度上限、频繁插删：用链式存储构造
  - 按位置频繁进行读取、数据域占用空间不大于指针域：用顺序存储构造

题型举例

有序的顺序表与无序的顺序表相比，其操作优势是（　　）。

A. 删除快　　B. 插入快

C. 生成快　　D. 查找快。

若对线性表进行的主要操作是按下标存取，且极少插入和删除，则应该采用的存储构造是_____；若需要频繁地对线性表进行插入和删除时，则应该采用的存储构造是_____。

第3章　栈与队列

栈与队列的定义、ADT定义及基本操作。

栈的应用：会跟踪递归函数执行过程

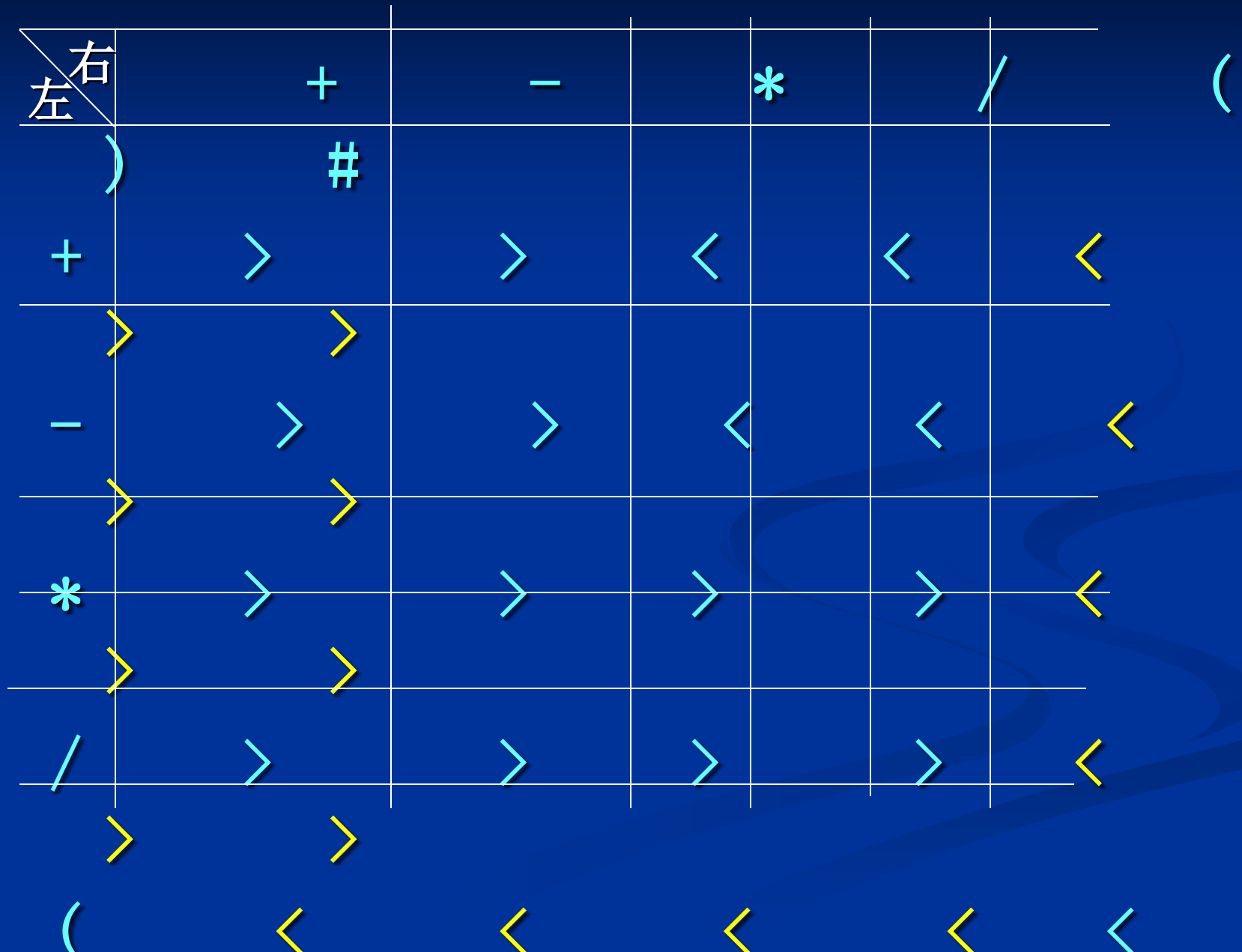
深度（纵向）环游

体现式求值(中缀→后缀→求值)

队列的应用：按层环游

注意：熟悉ADT的操作形式，会直接调用抽象数据类型中定义的操作

算符优先关系表



a/(b-c)+d*e#➔ abc-/de*+

输入	动作	栈内容	■ 后缀体现式	■ 相应算法环节
a	输出a	#	a	1
/	/>#，入栈	#/		4.2
(	入栈	#/（		2
b	输出b		ab	1
-	->(，入栈	#/（-		4.2
c	输出c		abc	1
)	-出栈输出,直至(	#/（	abc-	3.2
	)==(,出栈不输出	#/		3.2
+	+</,/出栈输出	#	abc-/	4.1
	+>#，入栈	#+		4.2
d	输出d		abc-/d	1
*	*>+，入栈	#+*		4.2
e	输出e		abc-/de	1

后缀体现式求值

- 循环：依次分析输入序列：
  - 1. 输入是操作数：则入栈；
  - 2. 输入是运算符：则两次出栈，取操作数2和操作数1，按照运算符进行计算。将成果入栈。
- 反复，直至遇到结束符“=”
- 此时栈顶值就是输入体现式的值。

第 4 章 字符串(了解)

- 基本概念
- 存储构造（顺序、原则类）
- 基本操作的含义

第5章　二叉树

- 定义、性质、存储构造（相应的类定义）
- 满二叉树、完全二叉树及扩充二叉树
- 抽象数据类型定义中的基本操作含义
- 深度环游（递归与非递归），广度环游
- 二叉搜索树插入、删除（改善）与检索算法；必须能够跟踪执行过程；求ASL。
- 堆概念、建堆、筛选、插、删的有关算法(过程)
- Huffman树构造及Huffman编码

深度优先环游二叉树（递归实现）

```
1.  template<class T>
2.  void BinaryTree<T>::PreOrder
    (BinaryTreeNode<T>* root)  {
3.          if(root!=NULL){
4.          Visit(root->value( ));          //前序
5.          DepthOrder(root->leftchild());  //访问左子树
6.                  Visit( root ) //中序
7.          DepthOrder(root->rightchild());//访问右子树
8.                  Visit( root ) //后序
9.          }
10. }
```

生成二叉搜索树

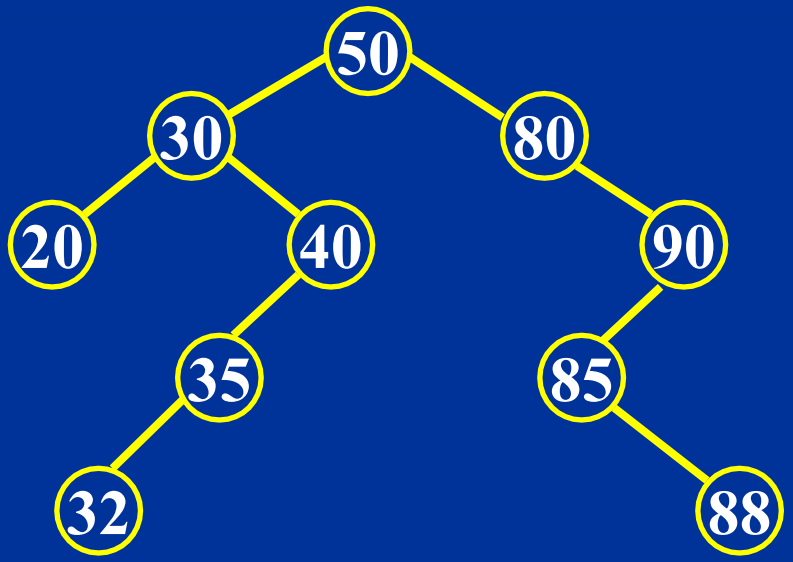
45,53,12,37,3,100,61,24,90,78

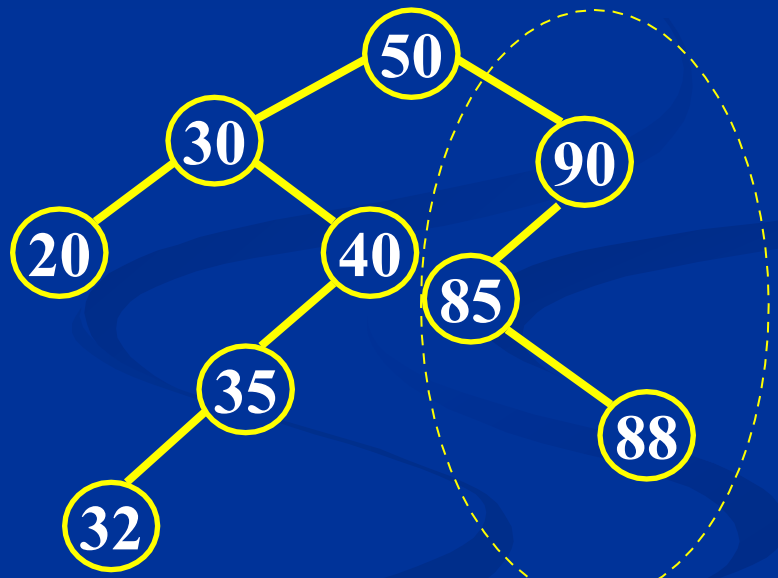
二叉搜索树的删除

- 在二叉搜索树里删除结点时，不是把以这个结点为根的子树都删除掉，只是删除一种结点，而且还要保持二叉搜索树的性质。
- 删除过程分为两种情况：
  - 没有左子树
  - 有左子树

没有左子树

若结点p没有左子树，则用右子树的根替代被删除的结点p。





有左子树（改善）

若p(50)有左子树，则在左子树里找中序环游的最终一种结点s(40)，删除s，用s的左子树替代s，用s替代被删p

这种措施能够降低二叉树的高度。