

线性表

线性表的定义。（重要）

线性表的类型：（非常重要）

顺序、链式。

顺序表的基础操作：（非常重要）

创建、查找、排序、插入、删除。

链式表的基础操作：（非常重要）

创建、查找、排序、插入（第一个节点之前、最后一个节点之后）、删除（第一个节点、最后一个节点）。

两种类型线性表的优缺点。（非常重要）

循环链表及其操作：（非常重要）

创建、插入、删除、查找，回顾猴子选大王。

双向链表及其操作：（非常重要）

创建、插入、删除、查找，插入删除与单向链表有什么不同。

特殊矩阵的压缩存储。（不重要）

稀疏矩阵的三元表示。（不重要）

矩阵、广义表和串

一、矩阵

1. 传统存储方法：二维数组 **(重要)**
2. 压缩存储方法 **(不重要)**
 1. 对角矩阵的压缩存储
 2. 稀疏矩阵的三元组表表示
 3. 系数矩阵的十字链表表示

二、广义表 (不重要)

1. 广义表的基本概念
2. 广义表的链式存储结构
3. 多元多项式的广义表表示

三、串

1. 串的基本概念 **(重要)**
2. 串的存储结构：顺序存储、链式存储 **(不重要)**
3. 串的相关算法：
 1. 串的相等关系判断 **(重要)**
 2. 串的插入 **(重要)**
 3. 串的模式匹配：朴素字符串匹配算法、KMP算法 **(不重要)**

栈

(仅代表助教个人经验)

3.1 栈的基本概念

重要等级：非常重要

考试可能题型：选填空题

- 栈的定义
- 栈的基本操作
 - 插入删除
 - 判断空栈满栈
 - 查看栈顶元素

3.2 栈的顺序储存结构

重要等级：非常重要

考试可能题型：编程题

```
//类型定义
#define MAXSIZE 1000
ElemType STACK[MAXSIZE];
int Top;
//初始化堆栈
void initStack()
{
    Top = -1;
}
//测试堆栈是否为空
int isEmpty()
{
    return Top == -1;
}
//测试堆栈是否已满
int isFull( )
{
    return Top == MAXSIZE - 1;
}
//进栈算法
void push( ElemType s[ ], ElemType item ){
    if( isFull() )
        Error("Full Stack!");
    else
        s[++Top] = item;
}
//出栈算法
ElemType pop( ElemType s[ ] )
{
    if( isEmpty() )
        Error("Empty Stack!");
    else
```

```
        return s[Top--];  
    }  
    //代码来自PPT
```

- 多栈共享连续空间问题（不太重要）

3.3 栈的链式储存结构

重要等级：重要

考试可能题型：编程题（个人感觉用顺序储存结构比较简单）

代码部分详见PPT

队列

（仅代表助教个人经验）

基本概念：

重要程度：很重要

考查类型：选填

- 队的定义：先进先出
- 队的操作：
 - 建队
 - 进队、出队
 - 队是否为空/满

队列的顺序存储结构

重要程度：一般

考查类型：编程

队列的链式存储结构

重要程度：很重要

考查类型：编程

队的操作：

1. 初始化队列

```
void initQueue()
{
    Front=NULL;
    Rear=NULL;
}
```

2. 判断队列是否为空

```
int isEmpty()
{
    return Front==NULL;
}
```

队空,返回1
否则,返回0

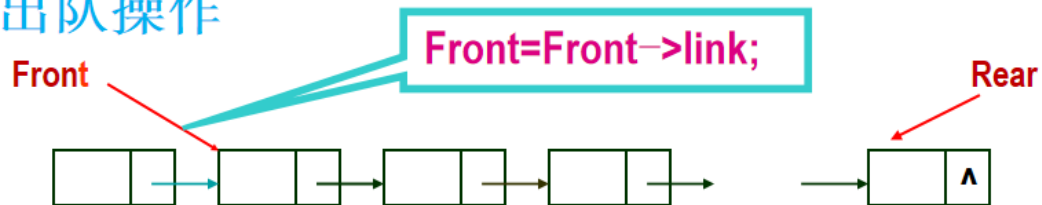
截图(Alt + A)

```

void enLQueue(ElemType item )
{
    QNodeptr p;
    if((p=(QNodeptr)malloc(sizeof(QNode))) ==NULL) /* 申请链结点 */
        Error("No memory! ");
    p->data=item;
    p->link=NULL;
    if(Front==NULL)
        Front=p;                /* 插入空队的情况 */
    else
        Rear->link=p;
    Rear=p;                      /* 插入非空队的情况 */
}

```

4. 出队操作



```

ElemType deLQueue()
{
    QNodeptr p;
    ElemType item;
    if(isEmpty())
        Error("Empty queue!"); /* 队为空，删除失败 */
    else{
        p=Front;
        Front=Front->link;
        item=p->data;
        free(p);
        return item;          /* 队非空，删除成功 */
    }
}

```

难点：循环队列，需要注意边界情况

数据结构知识点——树

(仅代表助教个人经验)

1. 树的基本概念：（比较重要，要求掌握）

- 度、结点的度、叶节点、分支结点、树的层次、树的深度/高度、森林

2. 树的存储方式：（掌握并会编写代码）

- 顺序存储结构（数组）
- 链式存储结构（居多）

3. 二叉树：（非常重要）

- 定义、满二叉树、完全二叉树

- 二叉树的性质：

1. 具有n个结点的非空二叉树共有 $n-1$ 个分支。
2. 非空二叉树的第i层最多有 2^{i-1} 个结点($i \geq 1$)。
3. 深度为h的非空二叉树最多有 $2^h - 1$ 个结点。
4. 若非空二叉树有 n_0 个叶结点,有 n_2 个度为2的结点,则 $n_0 = n_2 + 1$
5. 具有n个结点的非空完全二叉树的深度为

$$h = \lfloor \log_2 n \rfloor + 1$$

6. 完全二叉树中编号对应的数学关系（注意从0还是1开始）

- 根据给出的二叉树几条性质，会计算其他量

如给出一个具有767个结点的完全二叉树，求其叶子结点个数为（）

4. 树的遍历：（非常重要）

- 前序遍历、中序遍历、后序遍历（dfs）
- 按层次遍历（bfs）

查找

(仅代表助教个人经验)

3.1 基本概念

选填可能会考概念，可以面向作业题复习，建议复习以下几个概念，以免到时候会错题意。

- 属性/记录
- 查找表/(主)关键字
- 连续/链接/索引/散列组织形式

3.2 基本操作

- **查找**，参见第六次作业的第一题，十分基础十分重要！！
- 插入/删除/修改操作，可以看看链表和线性表那块。
- 排序(强烈建议复习qsort和cmp写法)。

3.3 查找方式

关于B-树，B+树，Trie树等ppt上带星号的知识点可以面向选填复习(毕竟就三道题，不会出太刁钻的点)。查找性能上考试不会为难大家，只要不出死循环就ok，大家选择自己最不容易写错的那种查找方法写~

- 顺序查找
 - 一种是线性结构，一种是链表结构。
 - 第一种for循环注意下标别笔误，第二种注意尾指针以及查不到情形时的处理。
- 折半查找
 - 递归和非递归都要会，考试如果非要用建议非递归，不容易错。
 - 注意不等号的带不带等号的细节。
- 索引查找
 - 选填概率比较大，编程到第六次作业第一题难度就ok。
- 二叉查找树
 - 建议面向选填复习，关于B+树，B-树可以好好看看结构体内各个字段的意义。
- 散列查找
 - 建议面向选填复习，感觉编程时候现写真的挺费劲。
 - 理解散列冲突及其解决方法(特别是链地址法)。

排序

(仅代表助教个人经验)

写在前面的话

- 排序在选填题和编程题中均会出现，但编程题通常是作为辅助算法而不是核心算法
- 至少要会写一种稳定排序算法，但最好是能熟练使用 `qsort` 函数，不要重复造轮子
- 注意排序的边界条件，比较容易出 bug
- 推荐一个算法可视化的网站：<https://visualgo.net/zh>

排序的基本概念

- 重要等级：重要
- 相关概念
 - 分类
 - 内排序：插入、选择、冒泡、希尔、堆排、二路归并、快排等
 - 外排序：多路归并等
 - 性能
 - 时间复杂度：比较次数、交换次数
 - 空间复杂度：额外内存消耗
 - 稳定性：相等元素的相对顺序是否改变

内排序

插入排序

- 重要等级：重要
- 算法核心：类似扑克牌理牌的过程，自左向右将元素依次向左插入合适的位置
- 性能：时间复杂度 $O(n^2)$ 、空间复杂度 $O(1)$ 、稳定

选择排序

- 重要等级：重要
- 算法核心：每次选择出最小的元素，并与首个元素交换
- 性能：时间复杂度 $O(n^2)$ 、空间复杂度 $O(1)$ 、不稳定

冒泡排序

- 重要等级：重要
- 算法核心：相邻元素两两比较，逆序则交换，自左向右重复该过程
- 性能：时间复杂度 $O(n^2)$ 、空间复杂度 $O(1)$ 、稳定

上面三种排序算法注意不要混淆，个人觉得选择排序较好写，比较符合人类直觉

希尔排序

- 重要等级：一般
- 算法核心：在冒泡排序的基础上优化，比较的是间隔为 gap 的元素，并不断缩小 gap 范围
- 性能：时间复杂度 $O(n\log n) \sim O(n^2)$ 、空间复杂度 $O(1)$ 、不稳定

堆排序

- 重要等级：一般
- 算法核心
 - 建一个堆，每次取出堆顶元素
 - 堆相关算法：建立、插入、删除堆顶
- 性能：时间复杂度 $O(n\log n)$ 、空间复杂度 $O(1)$ 、不稳定

二路归并排序

- 重要等级：一般
- 算法核心：将数组拆成两半，分别对左边一半和右边一半进行归并排序（递归），随后合并两个子数组
- 性能：时间复杂度 $O(n\log n)$ 、空间复杂度 $O(n)$ 、稳定

快速排序

- 重要等级：一般
- 算法核心：选取一个分界元素，将小于它的元素排在左侧，大于它的元素排在右侧，再分别对左右两个子数组重复该过程（递归）
- 性能：时间复杂度 $O(n\log n)$ 、空间复杂度 $O(\log n)$ 、不稳定

qsort 函数

- 重要等级：非常重要
- 编程题能用就用，不要重复造轮子
- 易错点
 - 数组的起始下标为 0 还是为 1？是 `qsort(ar, ...)` 还是 `qsort(ar + 1, ...)`？
 - 数组的元素个数是最开始输入的 n？还是动态维护的值？
 - 数组的元素大小：`sizeof(ar[0])` 或 `sizeof(struct Element)`
 - cmp 函数
 - int 比较时不要直接相减，否则可能溢出
 - double 比较时考虑浮点精度
 - 指针类型的转换是否正确
- 如何使 qsort 具有稳定性？将元素的起始下标作为该元素数据的一部分，并在 cmp 函数中将其作为最后一个关键字

最后放一页课程 ppt 中的总结内容

排序方法	平均情况	最好情况	最坏情况	辅助空间	稳定性
冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
简单选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
直接插入排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
希尔排序	$O(n\log n) \sim O(n^2)$	$O(n^{1.3})$	$O(n^2)$	$O(1)$	不稳定
堆排序	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(1)$	不稳定
归并排序	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n)$	稳定
快速排序	$O(n\log n)$	$O(n\log n)$	$O(n^2)$	$O(\log n) \sim O(n)$	不稳定

从算法性质来看：

- ◆ 简单算法：冒泡、选择、插入
- ◆ 改进算法：谢尔、堆、归并、快速

排序算法总结

从时间复杂度来看：

- ◆ 平均情况：后3种改进算法 > 谢尔 (远) > 简单算法
- ◆ 最好情况：冒泡和插入排序要更好一些
- ◆ 最坏情况：堆和归并排序要好于快速排序及简单排序

从空间复杂度来看：

- ◆ 归并排序有额外空间要求，快速排序也有相应空间要求，堆排序则基本没有。

从稳定性来看：

- ◆ 除了简单排序，归并排序不仅速度快，而且还稳定



(仅代表助教个人经验)

考试题型：选填空题

6.1 图的基本概念

重要等级：重要

- 图的定义
- 图的分类
 - 无向图
 - 有向图
 - 网
- 名词术语
 - 顶点的度
 - 出度
 - 入度
 - 路径和路径长度
 - 子图
 - 图的连通
 - 生成树

这部分主要是概念内容，大家利用ppt结合作业当中的题目把概念理清楚即可

图的存储方法

重要等级：非常重要

- 邻接矩阵存储方法
- 邻接表储存方法

这两种方法使用较多，大家只要认真完成了作业应该理解难度不大，另外需要理清楚哪种适合稀疏图那种适合稠密图

- 三元组储存方法
使用较少

图的遍历

重要等级：非常重要

- 深度优先遍历 DFS
- 广度优先遍历 BFS

图的遍历算法是图最基本的算法之一，希望大家能够使用代码实现，这两种方法在树中也经常出现，也可能结合树的题目出现在编程题中，选填空题中也可能出现

- 5 图的深度优先遍历类似于二叉树的_____。
- A. 前序遍历 B. 中序遍历 C. 后序遍历 D. 层次遍历

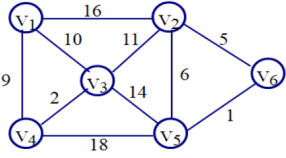
最小生成树

重要等级：非常重要

- Prim算法
- Kruskal 算法

求最小生成树也是比较重要的算法，建议大家都能实现一下，起码实现的步骤需要了解，考试之中也是经常出现选填空题

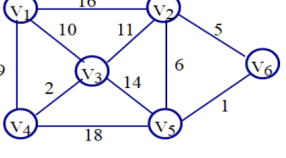
5



1.00

对于上图所示的无向连通图，若采用普里姆（Prim）算法求其最小生成树，假设第一个选择加入最小生成树的顶点为V1，则最后一条加入最小生成树的边的权值为_____。

6



1.00

对于上图所示的无向连通图，若采用克鲁斯卡尔（Kruskal）算法求其最小生成树，则最后选择加入最小生成树的边的权值为_____。

最短路径问题

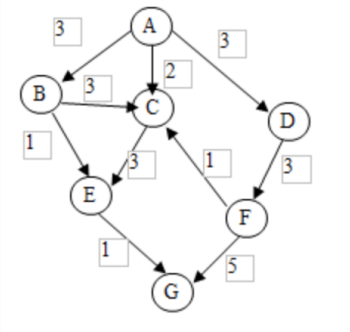
重要等级：非常重要

- Dijkstra算法

最短加权路径的主要实现方法，在前几年考试中常有出现，建议大家最好写代码实现一下，理解会更深，作业题中也有比较好的例题。

9

用迪杰斯特拉算法计算下图中A到G的最短路径为_____。（输出序列中不要有空格、标点符号等，保持大写，输出样例：ABCDE FG）



AOV 网和拓扑排序

- 拓补排序
重要等级：重要
- AOV网
重要等级：较不重要

拓补排序能够理解概念，完成作业中的题目即可，有能力的同学可以尝试使用c语言实现，理解会更深

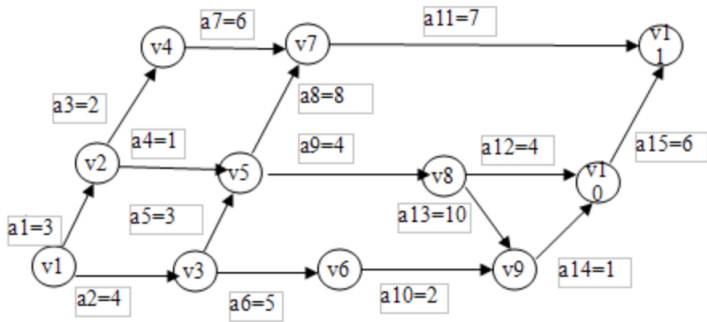
- 8 已知某有向图 $G=(V,E)$ ，其中 $V=\{v1,v2,v3,v4,v5,v6\}$ ， $E=\{<v1,v2>, <v1,v4>, <v2,v6>, <v3,v1>, <v3,v4>, <v4,v5>, <v5,v2>, <v5,v6>$ 1.00
 $>\}$ ， G 的拓补序列是_____。（输出序列中不要有空格、标点符号等，保持小写，输出样例：v1v2v3v4v5v6）

AOE网和关键路径

重要等级：较不重要

AOV网、AOE网这部分的内容考的相对较少，能够理解作业中选填空题应该没有问题

- 10 手工计算如下图所示的AOE网中的关键路径为_____（输出序列中不要有空格、标点符号等，输出样例：a1a2a15a10）。



网络流量问题

重要等级：较不重要

该部分内容考的很少，但是建议大家尽量了解一下