**《数据结构与算法》作业**

|  |  |  |
| --- | --- | --- |
| **[习题2 表结构](#_习题2，表)** | **[习题3 树结构](#_习题3，树)** | **[习题4 图结构](#_习题5，图)** |
| **[习题5 查找](#_习题5_查找)** | **[习题6 排序](#_习题6_排序)** |  |

注：作业在每章结束后提交。

# 习题1 绪论

1-1 名词解释：数据结构。

1-2 数据结构的基本逻辑结构包括哪四种?

1-3 “为什么要学数据结构与算法”这个问题，一般可以从( )、应用需求和程序优化等几个方面来理解。

(A) 硬件速度

(B) 操作系统

(C) 课程定位

(D) 问题规模

1-4 算法包括哪五种特性?

1-5 简述算法及其时间复杂度。

1-6 在本课程的学习方法中，有一种“通过实验训练，提高构造性思维能力，掌握特定问题的解决方法”。这里的“构造性思维”是指( )，为待解问题设计一个合理的框架，从而使问题转化并得到解决。

(A) 依据结构化思想

(B) 利用具体问题的典型特征

(C) 为数据选择适当的存储结构

(D) 整理定义在存储结构之上的基本操作

# 习题2 表结构

2-1 如果长度为n的线性表采用顺序存储结构存储，则在第i (1≤i≤n+1)个位置插入一个新元素的算法的时间复杂度为( )。

(A) O(1)

(B) O(n)

(C) O(nlog2n)

(D) O(n2)

2-2 在一个有127个元素的顺序表中插入一个新元素，要求保持顺序表元素的原有(相对)顺序不变，则平均要移动( )个元素。

(A) 7

(B) 32

(C) 64

(D) 127

2-3 已知An×n为稀疏矩阵。试从时间和空间角度比较，采用二维数组和三元组顺序表两种存储结构计算∑aij的优缺点。

2-4 综合比较顺序表和链表。

2-5 解释链表的“头指针、头结点和首元素结点”三个概念。

2-6 设链表L→a→b→c→d，指针域为\*next。执行下列命令后，(    )。

p=L->next->next;

L->next->next=NULL;

q=L->next->next;

(A) p→b→c→d，q→a

(B) p→b→c→d，q→NULL

(C) p→c→d，q→a

(D) p→c→d，q→a→b

2-7 描述下列算法的主要功能是(    )。

① 构造头结点L，取q=L;

② 产生1个结点p;

③ q−>next=p;

④ 输入p−>data的值;

⑤ 取q=p;

⑥ 重复执行②至⑤n次;

⑦ p−>next=NULL;

(A) 通过输入n个数据元素构建链表L

(B) 采用前插法，在链表L中输入n个数据元素

(C) 通过产生n个结点构建链栈L，q为栈顶指针

(D) 在链队列L中输入n个数据元素，q为队尾指针

2-8 设两个循环链表的长度分别为n和m，则将这两个循环链表连接成一个循环链表，最好的时间复杂度为( )。

(A) O(1)

(B) O(n)

(C) O(m)

(D) O(min(n，m))

2-9 设push和pop分别表示进栈和出栈操作，输入序列为xyz，则经过栈操作(    )可以输出序列yzx。

(A) push, push, pop, push, pop, pop

(B) push, push, push, pop, pop, pop

(C) push, pop, push, pop, push, pop

(D) push, pop, push, push, pop, pop

2-10 设进栈序列为123，试给出所有可能的出栈序列。

2-11 如果进栈序列为123456，能否得到出栈序列435612和135426?

2-12 简述算法的功能(设数据元素类型为int)：

void proc(LinkQueue \*Q)

{

LinkStack S;

InitStack(S);

while(!EmptyQueue(Q) )

{

DeleteQueue(Q, d);

Push(S,d);

}

while(!EmptyStack(S) )

{

Pop(S, d);

InsertQueue(Q, d);

}

}

2-13 描述下列递归算法的功能。

int F(int m, int n)

{

if (n>m) return F(n, m);

else if (n==0) return m;

else return F(n, m%n);

}

2-14 编写递归算法：

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | 0，m=0且n≥0 |
|  | g(m, n)= |  |
|  |  | g(m-1, 2n)+n，m>0且n≥0 |

2-15 将下列递归过程改写为非递归过程。

void test(int &s)

{

int x;

scanf ("%d", &x);

if (x==0) s=0;

else

{

test(s);

s+=x;

}

}

2-16 按照格式要求给出调用F(3,'A','B','C')的运行结果：

void F(int n, char x, char y, char z)

{

if (n==1) printf("1 %c 🡪 %c\n", x, z);

else

{

F(n-1, x, z, y);

printf("%d %c 🡪 %c\n", n, x, z);

F(n-1, y, x, z);

}

}

# 习题3 树结构

3-1设二叉树T中度为1的结点11个，度为2的结点12个，则二叉树T共有(      )个叶子结点。

(A) 11

(B) 12

(C) 13

(D) 36

3-2 设树T的度为4，其中度为1，2，3和4的结点个数分别为4，2，1，1，则T中的叶子数为( )。

(A) 5

(B) 6

(C) 7

(D) 8

3-3 已知一棵度为k的树中，有n1个度为1的结点，n2个度为2的结点，…，nk个度为k的结点。试计算该树的叶子结点数。

3-4 证明：如果二叉树T的叶子结点数为n0，度为2的结点数为n2，则n0=n2+1。

3-5 对于任意非空二叉树，要设计出其后序遍历的非递归算法而不使用栈结构，最适合的方法是对该二叉树采用( )存储结构。

(A) 二叉链表

(B) 三叉链表

(C) 索引

(D) 顺序

3-6 一棵二叉树的叶子结点在其先序、中序和后序序列中的相对位置( )。

(A) 肯定发生变化

(B) 可能发生变化

(C) 不会发生变化

(D) 无法确定

3-7 设二叉树T按照二叉链表存储，则下列递归算法的主要功能是(      )。

int F(BiTree T)

{

if (!T) return 0;

x=F(T->Lchild);

y=F(T->Rchild);

if (y>x) x=y;

return x+1;

}

(A) 交换二叉树T的左右子树

(B) 计算二叉树T的高度

(C) 计算二叉树T的叶子结点数

(D) 先遍历左子树，再遍历右子树

3-8 已知二叉树T的先序序列为ABCDEF，中序序列为CBAEDF, 则T的后序序列为( )。

(A) CBEFDA

(B) FEDCBA

(C) CBEDFA

(D) 不确定

3-9 简述由先序序列和中序序列构造二叉树的基本操作方法。

3-10 已知二叉树的先序序列为ebadcfhgjik，中序序列为abcdefghijk，试画出该二叉树。

3-11 已知二叉树T的中序序列和后序序列分别为

(中序) 3, 7, 11, 14, 18, 22, 27, 35

(后序) 3, 11, 7, 14, 27, 35, 22, 18

试画出二叉树T。

3-12 已知二叉树T按照二叉链表存储，设计算法，计算T中叶子结点的数目。

3-13 已知二叉树T按照二叉链表存储，设计算法，交换T的左子树和右子树。

3-14 先序后继线索化算法是根据二叉链表建立先序后继线索二叉链表，其基本原则是在前驱空指针域中写入后继线索，即将右子树的( )指针写入左子树的最后一个叶子结点右指针域。

(A) 线索

(B) 根结点

(C) 前驱结点

(D) 后继结点

3-15 设计算法，在先序线索二叉树中，查找给定结点p在先序序列中的后继。

3-16对n (n≥2)个权值均不相同的字符构造哈夫曼树T，不正确的叙述( )。

(A) T一定是一棵完全二叉树

(B) T中一定没有度为1的结点

(C) T中两个权值最小的结点一定是兄弟结点

(D) T中任一分支结点的权值一定不小于下一层任一结点的权值

3-17设计一个求结点x在二叉树中的双亲结点算法。

# 习题4 图结构

4-1 设某个非连通无向图有25条边，问该图至少有( )个顶点。

(A) 7

(B) 8

(C) 9

(D) 10

4-2 设某无向图中有n个顶点e条边，则建立该图邻接表的时间复杂度为( )。

(A) O(n+e)

(B) O(n2)

(C) O(ne)

(D) O(n3)

4-3 带权有向图G用邻接矩阵R存储，则顶点i的入度等于R中(  )。

(A) 第i行非∞(或非0)的元素之和

(B) 第i列非∞(或非0)的元素之和

(C) 第i行非∞(或非0)的元素个数

(D) 第i列非∞(或非0)的元素个数

4-4下面关于无向图的存储结构叙述中，正确的是( )。

(A) 用邻接表存储图，占用的存储空间大小与图中边数有关，与顶点数无关

(B) 用邻接表存储图，占用的存储空间大小与图中边数和顶点数都有关

(C) 用邻接矩阵存储图，占用的存储空间大小与图中边数和顶点数都有关

(D) 用邻接矩阵存储图，占用的存储空间大小与图中边数有关，与顶点数无关

4-5 设图G=(V, E)，V={ a, b, c, d, e }，E={<a, b>, <a, c>, <b, d>, <c, e>, <d, c>, <e, d>}。

(1)是否存在从c到b的路径?

(2)计算ID(d)、OD(d)、TD(d)；

(3)画出各个强连通分量。

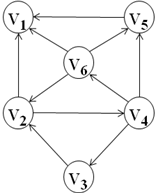
4-6 设计算法，由依次输入的顶点数目、狐的数目、各个顶点元素信息和各条狐信息建立有向图的邻接表。

4-7 请给出有向图的

(1) 每个顶点的入度和出度；

(2) 邻接矩阵；

(3) 邻接表。



4-8 设无向图G=(V，E)，V={a，b，c，d，e，f}，E={(a，b)，(a，e)，(a，c)，(b，e)，(c，f)，(f，d)，(e，d)}。从顶点a出发对图G进行深度优先搜索遍历，得到的顶点序列是( )。

(A) a b e c d f

(B) a c f e b d

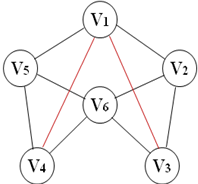
(C) a e b c f d

(D) a e d f c b

4-9 假设v1为出发点，优先考虑编号的顶点。试给出无向图的

(1)深度优先遍历的顶点序列和边序列；

(2)广度优先遍历的顶点序列和边序列。



4-10 概念解释：最小生成树。

4-11 设无向图G=(V, E)，V={a, b, c, d, e}，E={<a, b>, <a, c>, <a, d>, <b, c>, <c, e>, <d, e>}，G1=(V, E1)。如果G1是G的生成树，则错误的是( )。

(A) E1={<a, b>，<a, c>，<a, d>，<c, e>}

(B) E1={<a, b>，<a, c>，<c, e>，<d, e>}

(C) E1={<a, c>，<b, c>，<c, e>，<d, e>}

(D) E1={<a, d>，<b, c>，<c, d>，<d, e>}

4-12 判断一个有向图是否存在回路，除了可以利用深度优先遍历算法外，还可以利用( )。

(A) 广度优先遍历算法

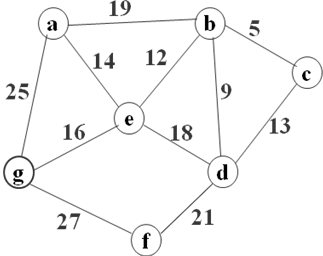
(B) 求最短路径的方法

(C) 拓扑排序方法

(D) 求关键路径的方法

4-13 设带权无向图G =(V, E)含有n个顶点m条边。试描述构造图G的最小生成树的克鲁斯卡尔(Kruskal)算法。

4-14 假设依据Prim算法产生无向网的最小生成树，出发顶点为a，则被选择的顶点序列是( )。



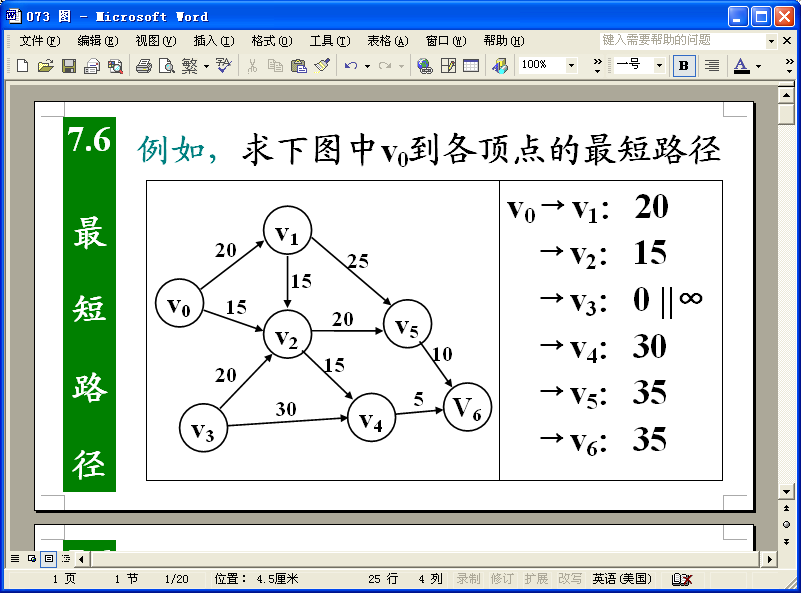
(A) a→b→c→d→e→f→g

(B) a→b→e→g→c→d→f

(C) a→e→d→b→c→f→g

(D) a→e→b→c→d→g→f

4-15 在有向图中，路径( )是从v0出发的一条最短路径。



(A) v0→v1→v5

(B) v0→v2→v3

(C) v0→v2→v4

(D) v0→v2→v5→v6

4-16 采用邻接表存储结构，设计一个算法，判别无向图G中指定的两个顶点之间是否存在一条长度为k的简单路径。

注：简单路径是指顶点序列中不含有重复的顶点。

4-17 设带权有向图G =(V, E)含有n个顶点、e条边，采用邻接矩阵Graph[n][n]作为存储结构。试设计算法Dijkstra(int V0，int n)，用于计算从源点V0到其它各顶点的最短路径。

4-18 设软件工程专业开设的主要课程如表所示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 代码 | 课程名称 | 先修课程 |  | 代码 | 课程名称 | 先修课程 |
| S1 | 高等数学 | 无 |  | S7 | 数据库系统 | S5 |
| S2 | 程序设计基础 | 无 |  | S8 | 编译技术 | S5 |
| S3 | 离散数学 | S1 |  | S9 | 算法分析 | S1, S5 |
| S4 | 计算机组成原理 | S2 |  | S10 | 软件工程导论 | S5 |
| S5 | 数据结构与算法 | S2, S3 |  | S11 | 计算机网络 | S4, S6 |
| S6 | 操作系统 | S4, S5 |  |  |  |  |

试根据先修课程要求绘制课程体系拓扑结构图(结点用课程代码表示)。

4-19 设含有6个顶点a, b, c, d, e, f的有向带权图G，其邻接矩阵如下：

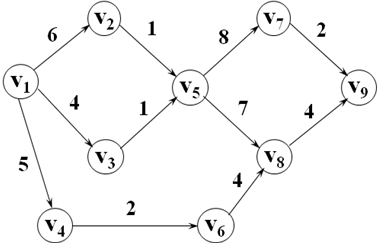
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ＼ | a | b | c | d | e | f |
| a | 0 | 4 | 6 | 0 | 0 | 0 |
| b | 0 | 0 | 5 | 0 | 0 | 0 |
| c | 0 | 0 | 0 | 4 | 3 | 0 |
| d | 0 | 0 | 0 | 0 | 0 | 3 |
| e | 0 | 0 | 0 | 0 | 0 | 3 |
| f | 0 | 0 | 0 | 0 | 0 | 0 |

要求：

(01)画出有向带权图G；

(2)求图G的关键路径，并计算关键路径长度。

4-20 设v1是源点、v9是汇点，则在有向图中，( )是一条关键路径。



(A) v1→v4→v6→v8→v9

(B) v1→v3→v5→v7→v9

(C) v1→v2→v5→v8→v9

(D) v1→v2→v5→v7→v9

# 习题5 查找

5-1 设顺序表的长度为30，平均分成5块，每块6个元素。如果采用分块查找，则其平均查找长度为(    )。（5+1）/2+（6+1）/2

(A) 5

(B) 5.7

(C) 6.5

(D) 8.2

5-2 将关键字2，4，6，8，10，12，14，16依次存放于一维数组A[0...7]中，如果采用折半查找方法查找关键字，在等概率情况下查找成功时的平均查找长度为( )。

(A) 21/8

(B) 7/2

(C) 4

(D) 9/2

5-3 简单描述静态查找和动态查找的区别。

5-4 设数组A中只存放正数和负数。试设计算法，将A中的负数调整到前半区间，正数调整到后半区间。分析算法的时间复杂度。

5-5 按照“逐点插入方法”建立一个二叉排序树，树的形状取决于( )。

(A) 数据序列的存储结构

(B) 数据元素的输入次序

(C) 序列中的数据元素的取值范围

(D) 使用的计算机的软、硬件条件

5-6 用利用逐点插入法建立序列(50, 72, 43, 85, 75, 20, 35, 45, 65, 30)对应的二叉排序树以后，查找元素35要在元素间进行( )次比较。

(A) 3

(B) 4

(C) 5

(D) 8

5-7 给定n个整数，设计算法实现：

(1) 构造一棵二叉排序树；

(2) 从小到大输出这n个数。

5-8 在平衡二叉树中，插入关键字46后得到一颗新的平衡二叉树。在新的平衡二叉树中，关键字37所在结点的左、右孩子结点中保存的关键字是( )。

25

18

53

37

69

(A) 18，46

(B) 25，46

(C) 25，53

(D) 25，69

5-9 用依次插入关键字的方法，为序列{ 5, 4, 2, 8, 6, 9 }构造一棵平衡二叉树(要求分别画出构造过程中的各棵不平衡二叉树)。

5-10 链地址法是Hash表的一种处理冲突的方法，它是将所有哈希地址相同的数据元素都存放在同一个链表中。关于链地址法的叙述，不正确的是( )。

(A) 平均查找长度较短

(B) 相关查找算法易于实现

(C) 链表的个数不能少于数据元素的个数

(D) 更适合于构造表前无法确定表长的情况

5-11 设哈希(Hash)函数H(k)=(3k)%11，用线性探测再散列法处理冲突，di=i。已知为关键字序列22，41，53，46，30，13，01，67构造哈希表如下：

哈希地址 0 1 2 3 4 5 6 7 8 9 10

关键字 22 41 30 01 53 46 13 67

则在等概率情况下查找成功时的平均查找长度是( )。1+1+2+2+1+1+2+6=16/8

(A) 2

(B) 24/11

(C) 3

(D) 3.5

5-12 有100个不同的关键字拟存放在哈希表L中。处理冲突的方法为线性探测再散列法，其平均查找长度为。试计算L的长度(一个素数)，要求在等概率情况下，查找成功时的平均查找长度不超过3。

素数表：101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167。

# 习题6 排序

6-1 如果顺序表中的大部分数据元素按关键字值递增有序，则采用( )算法进行升序排序，比较次数最少。

(A) 快速排序

(B) 归并排序

(C) 选择排序

(D) 插入排序

6-2 若一组记录的关键码为46, 79, 56, 38, 40, 84，则利用快速排序方法且以第一个记录为基准，得到的一次划分结果为( )。

(A) 38，40，46，56，79，84

(B) 40，38，46，79，56，84

(C) 40，38，46，56，79，84

(D) 40，38，46，84，56，79

6-3 对数据36，12，57，86，9，25进行排序，如果前三趟的排序结果如下：

第1趟：12，36，57，9，25，86

第2趟：12，36，9，25，57，86

第3趟：12，9，25，36，57，86

则采用的排序方法是( )。

(A) 插入排序

(B) 起泡排序

(C) 归并排序

(D) 快速排序

6-4设int r[9]={0, 25, 28, 13, 33, 56, 47, 19, 40}; 则调用F(r, 1, 8)之后，数组r[ ]中的数据元素存放顺序是(  )。

F(int r[ ], int s, int t)

{

for (int j=2\*s; j≤t; j\*=2)

{

if (r[j+1]<r[j]) ++j;

if (r[s]≤r[j]) break;

int x=r[s];

r[s]=r[j];

r[j]=x;

s = j;

}

}

(A) 0, 13, 19, 25, 28, 33, 40, 47, 56

(B) 56, 47, 40, 33, 28, 25, 19, 13, 0

(C) 0, 25, 28, 13, 33, 40, 47, 19, 56

(D) 0, 13, 28, 19, 33, 56, 47, 25, 40

6-5 在链式基数排序中，对关键字序列369, 367, 167, 239, 237, 138, 230, 139进行第1趟分配和收集后，得到的结果是( )。

(A) 167, 138, 139, 239, 237, 230, 369, 367

(B) 239, 237, 138, 230, 139, 369, 367, 167

(C) 230, 367, 167, 237, 138, 369, 239, 139

(D) 138, 139, 167, 230, 237, 239, 367, 369

6-6 设int r[7]={5，2，6，4，1，7，3}; 则执行for ( i=0; i<7; i++) r[r[i]-1]=r[i]; 命令之后，数组r[7]中的数据元素存放顺序是( )。

(A) 5，2，7，4，1，6，3

(B) 3，2，1，4，5，7，6

(C) 1，2，3，4，5，6，7

(D) A、B、C都不对

6-7 设计一种排序算法，对1000个[0, 10000]之间的各不相同的整数进行排序，要求比较次数和移动次数尽可能少。

6-8 设顺序表的结点结构为(Type Key; int Next)，其中，Key为关键字，Next为链表指针。试设计静态链表排序算法。

6-9 假设n个部门名称的基本数据存储在字符数组name[N][31]中，0≤n≤N≤20。试设计一个起泡排序算法，将n个部门名称按字典序重新排列顺序。

6-10 假设采用链表存储类型：

typedef struct RNode

{

int key; //数据域(也是关键字域)

struct RNode \*next; //指针域

} RNode, \*RList;

typedef RList R[N]; //链表类型, 常变量N≥n

又设R[1..n]是[10, 999]之间的随机整数。试设计一个链表基数排序算法，将R[n]中的数从小到大排序。排序结果仍存放在R[n]中。

6-11 在下列排序算法中，时间复杂度最好的是( )。

(A) 堆排序

(B) 插入排序

(C) 起泡排序

(D) 选择排序

6-12 根据建堆算法，将关键字序列5，7，10，8，6，4调整成一个大顶堆，最少的交换次数为( )。

(A) 1

(B) 2

(C) 3

(D) 4

6-13 给定关键字序列503, 87, 512, 61, 908, 170, 897, 275, 653, 426。

(1)以第一个关键字为枢轴，给出第1趟快速排序后的关键字序列;

(2)给出根据堆排序算法建立的小顶堆序列。

6-14 设计基于顺序表存储结构的树形选择排序算法。