数据结构知识点汇总

第一章 绪论

第一部分

1. 数据结构三要素：逻辑结构、存储结构和数据的运算。
2. 数据的物理结构主要包括顺序存储结构和链式存储结构两种情况。
3. 数据的逻辑结构是对数据之间关系的描述，主要有线性结构和非线性结构两大类。(2015 年 820 填空题 1)
4. 线性结构主要包括以下几种数据结构线性表的顺序和链式结构、栈和队列、串、数组和广义表。
5. 数据结构是相互之间存在一种或多种特定关系的数据元素的集合。
6. 算法特性：有穷性、确定性、可行性、输入和输出。
7. 一个“好”的算法应考虑达到以下目标：正确性、可读性、健壮性和效率与低存储量需求。(2014 年 820 填空题 1)
8. 算法是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每一条指令表示一个或多个操作。
9. 常用复杂度大小：$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)<O(n!)<O(n^n)$。
10. 程序设计是数据结构的选用和算法设计的组合。(2006 年 820 填空题 3)

第二章 线性表

第一部分

1. 线性表是具有相同数据类型的 n（n≥0）个数据元素的有限序列。
2. 线性表是一种逻辑结构，顺序表和链表是指存储结构。
3. 读取数组给定下标的数据元素的操作，称为取值操作；存储或修改数组给定下标的数据元素的操作，称为赋值操作。(2010 年 820 填空题 1)
4. 循环队列的判满、判空方法。

牺牲一个存储单元法：

```
队满条件：(Q.rear + 1)%MaxSize == Q.front
队空条件：Q.front == Q.rear
队列中元素个数：(Q.rear - Q.front + MaxSize) % MaxSize
```

计数器法：

```
队满条件：Q.size == MaxSize
队空条件：Q.size == 0
```

标志位法：

```
队满条件：tag == 1
队空条件：tag == 0
若因插入导致 Q.front == Q.rear，置 tag = 1，
若因删除导致 Q.front == Q.rear，置 tag=0。
```

5. 顺序表采用的是随机存取方式，线性链表采用的是顺序存取方式。(2016 年 820 填空题 1)
6. 若希望从链表的任何一个结点出发都能访问到表的其他结点，应采用循环单链表或循环双链表。(2008 年 820 填空题 7)

第二部分

1. 线性表有哪两种存储结构？在这两种存储结构中元素之间的逻辑关系分别是

通过什么决定的？（2004 年 820 简答题 4）

有顺序和链式两种存储结构，顺序结构中元素之间的逻辑关系由物理存储位置决定，链式结构中元素之间的逻辑关系由链指针决定。

2. 若表的总数基本稳定，且很少进行插入和删除，但要求以最快的速度存取表中的元素，这时应采取哪种存储表示？为什么？

采用顺序表。若表的总数基本稳定，且很少进行插入和删除，则顺序表可以充分发挥它的存取速度快、存储利用率高的优点。

3. 简述单链表中设置头结点的作用。

a. 由于开始结点的位置被存放在头结点的指针域中，所以在链表的第一个位置上的操作和在表其它位置上的操作一致，无须进行特殊处理。

b. 无论链表是否为空，其头指针是指向头结点的非空指针（空表中头结点的指针域为空），因此空表和非空表的处理也就统一了。

4. 如果有 n 个表同时并存，并且在处理过程中各表的长度会动态发生变化，表的总数也可能自动改变，在此情况下，应选用哪种存储表示？为什么？

采用链表。如果采用顺序表，在多个表并存的情况下，使用表浮动技术在同一存储空间内定义多个顺序表，初始时把整个空间均等地分配给每个表，在问题求解的过程中，一旦发现某个表有放满并溢出的情况，必须移动其它表以扩充溢出表的空间，导致不断把大片空间移来移去，不但时间耗费很大，而且操作复杂，容易出错。如果表的总数还要变化，操作起来就更困难。如果采用链表就没有这些问题，各个表自行扩充，各自操作。

5. 为什么在单循环链表中设置尾指针比设置头指针更好？

尾指针是指向终端结点的指针，用它来表示单循环链表可以使查找链表的开始结点和终端结点都很方便。设置一个带头结点的单循环链表，其尾指针是 rear，则开始结点和终端结点分别为指针 rear 所指结点的后继结点的后继结点和指针 rear 所指结点，即 rear->next->next 和 rear，查找时间均为 O(1)。若用头指针来表示该链表，则查找开始结点为 O(1)，终端结点为 O(n)。

6. 简述递归过程的关键点。

a. 反复用与原问题相似但更简单的新问题来表示较复杂的原问题，直到问题可解。

b. 不能产生自己调用自己的无穷序列，即必须有递归调用出口。

7. 描述堆栈和递归的关系。

递归过程是一种调用自身的函数，在调用的过程中存在转入子程序的过程。在每次转入子程序前需要保护现场，则将相应参数和中间结果压入系统堆栈，而在子程序返回的时候，需要恢复现场，则将之前压栈的数据从系统堆栈弹出，因此，递归过程存在隐含的堆栈操作，而且子程序的调用过程满足堆栈先进后出的特性。

第三章 栈和队列

第一部分

1. 判定循环队列的满与空，有三种方法，它们是计数器法、标志位法和牺牲一个存储单元法。（2012 年 820 填空题 1）

2. 广义表难以用顺序存储结构，适合编写递归算法的广义表的存储结构是表头表尾链。（2006 年 820 填空题 5）

3. 对广义表进行操作，结果总是表的基本操作是取表尾操作。（2010 年 820 填空题 6）

4. 实现二叉树按层次遍历算法时，最适合的数据结构是队列。（2010 年 820 填空题 4）

5. 在 n*n 的对称矩阵中，采用只存储下三角部分，只需 n(n+1) / 2 个存储单元。（2008 年 820

填空题 6)

6. 广义表 A(b,A)的长度为 2，深度为无穷。(2005 年 820 填空题 4)

7. 线性表、栈和队列都是线性结构，可以在线性表的任何位置插入和删除元素；而栈只能在栈顶插入和删除元素；对于队列只能在队尾插入元素、在队头删除元素。(2004 年 820 填空题 1)

8. 取出广义表 L=(x,(x,y,z,a))中原子 y 的函数是 Head(Tail(Head(Tail(L))))。(2004 年 820 填空题 3)

9. 递归过程实现时使用的数据结构是栈。层次遍历二叉树时使用的数据结构是队列。(2002 年 820 填空题 3)

10. 已知广义表 A=(a,b)， B=(A,A)， C=(a,(b,A),B),tail(head(tail(C)))的结果是(A)。(2002 年 820 填空题 7)

第二部分

1. 栈的概念。

栈是限定仅在一端进行插入或删除操作的线性表，允许进行插入或删除的一端称为栈顶，另一端称为栈底。

当 n 个编号元素以某种顺序进栈，并且可在任意时刻出栈，所获得的编号元素排列的数目 N 恰好满足 Catalan 函数的计算，即

$$N = \frac{1}{n+1} C_{2n}^{n}$$

2. 队列的概念。

队列是一种仅允许在表的一端进行插入，而在表的另一端进行删除的线性表，允许插入的一端称为队尾，允许删除的一端称为队头。

3. 栈和队列各有什么特点，什么情况下用到栈，什么情况下用到队列？

栈的主要特点是“后进先出”，栈的应用十分广泛，通常将递归算法转换成非递归算法时需要使用栈，栈的应用有括号匹配、算术表达式求值和迷宫问题求解等。

队列的主要特点是“先进先出”，队列的应用也十分广泛，特别在操作系统资源分配和排队论中大量地使用队列，还有队列在层次遍历中的应用。

4. 队列是一个表头和表尾，既能插入又能删除的线性表。该说法是否正确？为什么？

不正确。队列是一个限制只能在表头删除和只能在表尾插入的线性表，上述说法只说明了队列有表头和表尾，而插入和删除位置没有具体限定。

5. 对数组 Am×n，它主要有哪两种顺序存储结构？在这两种存储方式中元素 aij 的存储地址 loc(i,j)的值是多少？（注：每个元素占一个存储单元，loc(1,1)=1）

按行优先存储，loc(i,j)=(i-1)*n+j。

按列优先存储，loc(i,j)=(j-1)*m+i。

6. 稀疏矩阵的三元组表的说明中，nu,mu,tu,data 存放什么内容？

nu 存稀疏矩阵行数，mu 存稀疏矩阵列数，tu 存非零元个数，data 存放非零元素。

第四章 树与二叉树

第一部分

1. 遍历二叉树实质上是对一个非线性结构进行线性化操作。(2014 年 820 填空题 2)

2. 线索二叉树标志域含义：

ltag == 0，lchild 域指向结点的左孩子；ltag == 1，lchild 域指向结点的前驱结点。

rtag == 0，rchild 域指向结点的右孩子；rtag == 1，rchild 域指向结点的后继结点。

3. N_h 表示深度为 h 的平衡树中含有的最少结点数，则 $N_0=0, N_1=1, N_2=2$，并且有 $N_h=N_{h-1}+N_{h-2}+1$。
4. 对二叉排序树中序遍历可以得到线性有序序列。(2013 年 820 填空题 6)
5. 为了保持二叉排序树的高效查找效率，在插入结点时常需要作处理。
6. 由一棵二叉树的后序序列和中序序列可唯一确定这棵二叉树。(2007 年 820 填空题 1)
7. 二叉树结点数 n 与边数 e 的关系为 n-e=1。(2007 年 820 填空题 2)
8. 树是以结点的分支定义层次结构，表示数据元素之间一对多的关系。(2006 年 820 填空题 2)
9. 将一棵树转换成二叉树后，根结点没有右子树。(2005 年 820 填空题 2)
10. AVL 树不一定是完全二叉树；完全二叉树一定是 AVL 树。(2005 年 820 填空题 5)
11. 含有 3 个结点的不同的二叉树有 5 棵。(2005 年 820 填空题 7)
12. 给定 n 个值构造哈夫曼树，经过 n-1 次合并才能得到最终的哈夫曼树。(2004 年 820 填空题 2)
13. 有 n 个结点的 d 度树，若用 d 个链域的多重链表表示，则只有 n-1 个非空链表。(2002 年 820 填空题 6)

第二部分

1. 树的路径长度与树的带权路径长度有什么区别？

树的路径长度是根到每一个结点的路径长度之和；树的带权路径长度为树中所有叶结点的权值与路径长度的乘积的总和。两者的区别为：

a. 计算的范围：前者涉及所有结点，后者仅考虑叶结点。
b. 计算的方法：前者仅是路径长度之和，后者是权值与路径长度乘积之和。

2. 简述树与线性表结构上的不同点。

树中结点可有多个后继，但仅有一个前驱，而线性表中一个结点的前驱和后继均最多一个。树中元素的关系是一对多，而线性表是一对一。

3. 简述树与它的二叉树表示(孩子—兄弟表示)间的关系？

a. 根是同一结点。
b. 二叉树中结点的左孩子域指向树中它的第一个孩子。
c. 二叉树中结点的右孩子域指向树中它的下一个兄弟。

4. 二叉树性质。

a. $n_0 = n_2 + 1$；
b. 二叉树的第 i 层上最多有 2^{i-1}(i≥1)个结点；
c. 高度为 k 的二叉树最多有 2^k-1(k≥1)个结点；
d. 有 n 个结点的完全二叉树，各结点从上到下、从左到右依次编号（1-n），则有：
　　如果 i!=1，则 i 结点的双亲为 $\lfloor i/2 \rfloor$；
　　如果 2i≤n，则 i 结点左孩子为 2i；否则无左孩子；
　　如果 2i+1≤n，则 i 结点右孩子为 2i+1；否则无右孩子；
e. 具有 n 个结点的完全二叉树的高度为 $\lceil \log_2(n+1) \rceil$ 或 $\lfloor \log_2 n \rfloor + 1$；
f. Catalan 函数：给定 n 个结点，能构成 h(n)种不同的二叉树：

$$\mathbf{h(n)} = \frac{\mathbf{C_{2n}^{n}}}{\mathbf{n+1}}$$

。

5. 回答满足后序序列和前序序列正好相同的二叉树的树型和正好相反的二叉树的树型各是什么？(2003 年 820 简答题 1)

相同：只有一个根节点的二叉树。
相反：任意结点均无左孩子的二叉树（仅有右孩子的二叉树）
　　或任意结点均无右孩子的二叉树（仅有左孩子的二叉树）

综合：高度等于结点数

6. 什么样的二叉树，对它采用任何次序的遍历，结果都相同？（2004 年 820 简答题 3）

仅有根结点的二叉树或空二叉树。

7. 根据中序、先序、后序遍历二叉树的特点，将根结点、叶结点、叶结点或无左子树结点、叶结点或无右子树结点填入下表空白处。

	第一个被访问的结点	最后一个被访问的结点
先序遍历二叉树	根结点	叶结点
中序遍历二叉树	叶结点或无左子树结点	叶结点或无右子树结点
后序遍历二叉树	叶结点	根结点

8. 简述在中序线索树中，求结点 P 中元素在中序遍历中的前驱元素存储地址 Q 的主要步骤。

（1）当 P->ltag == 1，则 Q = P->lchild，结束

（2）Q = P->lchild

（3）当 Q->rtag == 1，则结束。

（4）Q = Q->rchild，转（3）

9. 简述在中序线索二叉树中找结点 N 的后继结点的主要思想。

（1）若 N->rtag == 1，则表示 P 的右孩子域指向后继结点，N 的后继结点为 N->rchild，结束，否则执行（2）。

（2）Q = P->rchild，即转向 P 的右孩子。

（3）当 Q->ltag == 1，则表示 Q 的左孩子域指向该结点的前驱结点，即 N 结点，所以 N 的后继结点为该结点，结束。

（4）Q = Q->lchild，转（3）。

10. 设 HT 为哈夫曼树，叶结点 a 的路径长度 La 比叶结点 b 的路径长度 Lb 长，试证明：叶结点 b 的权值 Wb 不小于叶结点 a 的权值 Wa。

反证法，设 Wb<Wa，HT 的带权路径长度 WPL1=Wa*La+Wb*Lb+Others。Others 为除 a,b 外其它叶结点的权值与路径长度乘积之和。现交换叶结点 a 和 b，则新树 NewT 的带权路径长度 WPL2=Wa*Lb+Wb*La+Others。WPL1-WPL2=Wa*La+Wb*Lb-(Wa*Lb+Wb*La)=Wa*(La-Lb)-Wb*(La-Lb)=(Wa-Wb)*(La-Lb)>0 则 WPL1 不是带权路径最小的树，与 HT 为哈夫曼树矛盾，故 Wb≥Wa。

11. 试证明若按中序遍历给定二叉树，能得到结点有序序列，则该二叉树是二叉排序树。

反证法，假设按中序遍历给定二叉树，得到结点有序序列，而该二叉树又不是二叉排序树。设 D 表示某结点值，L 表示该结点的左子树，R 表示该结点的右子树，MIN(X)和 MAX(X)分别表示 X 子树的最小、最大值，即存在这样的子树：使不等式 MAX(L)<D<MIN(R)不成立，但是按中序遍历给定二叉树的任意子树的顺序一定是 LDR，又因得到的是结点的有序序列，所以对任意子树 MAX(L)<D<MIN(R)一定成立，假设矛盾。故该二叉树一定是二叉排序树。

12. 当查找表有既能较快查找又能适应动态变化的需求时，选用什么查找方法最适合？并简述其理由。

选用二叉排序树。当较平衡时，二叉排序树的查找性能接近于二分查找，插入删除元素只需修改指针，性能也好。

13. 已知一颗二叉排序树 BST 和中序遍历算法 inorder，如何能得到从大到小的结点序列。

方法一：修改中序遍历算法为 RNL，即先递归遍历右子树，输出根结点，然后递归遍历左子树。

方法二：将二叉排序树的所有左右子树交换，然后用中序遍历算法遍历，所输出的结点序列就是从大到小的有序序列。

14. 设二叉排序树 T 中各结点关键字互不相同，x 指向 T 的叶结点且是双亲 y 的右孩子，试证明 y->key 是 T 中小于 x->key 的所有关键字的最大者。(2005 年 820 简答题 1)

因 x 是其双亲 y 的右孩子，按中序遍历二叉排序树 T 可以得到有序序列：

…, y->key, x->key, … 由此可得 y->key 是 T 中小于 x->key 的所有关键字的最大者。

15. 试证明若按中序遍历给定二叉树，能得到结点有序序列，则该二叉树是二叉排序树。(2005 年 820 简答题 3)

反证法，假设按中序遍历给定二叉树，得到结点有序序列，而该二叉树又不是二叉排序树。即存在这样的子树：使得不等式 MAX(L)<D<MIN(R)不成立，但是按中序遍历给定二叉树的任意子树的顺序一定是 LDR，又因得到的是结点有序序列，所以对任意子树 MAX(L)<D<MIN(R)一定成立，假设矛盾。故该二叉树一定是二叉排序树。

16. 简述递归过程的关键点。(2005 年 820 简答题 4)

1)反复用与原问题相似但更简单的新问题来表示较复杂的原问题，直到问题可解；

2)不能产生自己调用自己的无穷序列，即必须有递归调用出口。

17. 将二叉树中所有结点按后序遍历顺序排列。并在每个结点中附加一个 0 到 3 之间的整数，以表示结点的孩子状态。该整数为 0 时，表示相应的结点为叶结点；为 1 时，表示相应的结点只有一个左儿子；为 2 时，表示相应的结点只有一个右儿子；为 3 时，表示相应的结点有左右两个儿子。请回答如何找结点 i 的右孩子?(2005 年 820 简答题 5)

若结点 i 有右儿子，它一定排在结点 i 的前一个，即 i-1 为其右儿子。

第五章 图

第一部分

1. 图不可以是空图，即顶点集不能为空，但边集可以为空。
2. 设图中顶点数为 n，则其生成树有 n-1 条边；若图的边数大于 n-1，则一定是有环(回路)图；若图的边数小于 n-1，则一定是非连通图。(2004 年 820 填空题 5)
3. 连通分量是无向图中的极大连通子图。(2010 年 820 填空题 3)
4. 要保证连通具有 10 个顶点的无向图，至少需要 37 条边。(9 个顶点构成完全图，再加一条边连通第 10 个顶点)
5. 在无权的无向图 G 的邻接矩阵 A 中，若(v_j, v_i)属于图 G 的边集合，则对应元素 A[i][j]等于 1。(2012 年 820 填空题 3)
6. 图的遍历算法复杂度:

DFS 空间复杂度：O(v) 时间复杂度：邻接矩阵 $O(v^2)$ 邻接表 O(v+e)；

BFS 空间复杂度：O(v) 时间复杂度：邻接矩阵 $O(v^2)$ 邻接表 O(v+e)。

7. 对有 n 个顶点、e 条边且使用邻接表存储的有向图进行广度优先遍历，其算法复杂度是 O(n+e)。
8. 图的遍历结果所依赖的因素有顶点编号、存储结构和所执行的算法。
9. 普里姆(Prim)算法时间复杂度 $O(v^2)$，适用于稠密图。

10. 克鲁斯卡尔(Kruskal)算法时间复杂度与排序算法 sort 有关，适合于稀疏图。
11. 迪杰斯特拉(Dijkstra)算法时间复杂度 $O(n^2)$。
12. 弗洛伊德(Floyd)算法时间复杂度 $O(n^3)$。
13. Floyd 最短路径算法中，$A^{(K)}[i,j]$表示从顶点 V_i 到顶点 V_j 中间顶点序号不大于 k 的最短路径长度。(2004 年 820 填空题 4)
14. 在有向图的邻接矩阵中，若主对角线以下的元素均为零，则该图的拓扑有序序列是存在的。(2005 年 820 填空题 3)
15. 在 AOE 网中，从源点到汇点所经历的边的权值之和最小的路径，称为最短路径；从源点到汇点所经历的边的权值之和最大的路径，称为关键路径。(2008 年 820 填空题 8)
16. AOV 网是一种有向无回路的图。
17. 对 DAG 图表示的整个工程和系统，人们最关心的是两个方面的问题:工程能否顺利进行、估算整个工程完成所必须的最短时间。
18. 一个有向图的邻接表和逆邻接表中结点的个数相同。(2013 年 820 填空题 2)
19. 一个有向无环图的拓扑排序序列不/不一定是唯一的。(2013 年 820 填空题 7)
20. 若能得到拓扑有序序列，则有向图无回路。(2010 年 820 填空题 2)
21. 迪杰斯特拉(Dijkstra)算法是求带权有向图中某个源点到其余各顶点(单源)的最短路径,是按路径长度的递增次序产生最短路径的。(2008 年 820 填空题 4)
22. 若希望得到树高较矮的生成树，则采用图的 BFS 广度优先搜索遍历算法。(2007 年 820 填空题 4)
23. 4 个顶点的无向完全图一共有个生成树。(2008 年 820 填空题 5)
24. 设网中的顶点数为 n，边的条数为 e，则普里姆(Prim)最小生成树算法的时间复杂度为 $O(n^2)$，适合于边稠密的网，克鲁斯卡尔(Kruskal)最小生成树算法的时间复杂度为 O(eloge)。(2006 年 820 填空题 1)
25. 有向图用邻接矩阵表示，删除所有从第 i 个结点出发的边的方法是将第 i 行全置为 0。(2002 年 820 填空题 4)

第二部分

1. 对有向图和无向图，分别描述如何判定图中是否存在回路。(2001 年 820 简答题 3)

对有向图，可以用拓扑排序算法来判定图中是否存在回路，当输出顶点数小于顶点数时，图中存在回路，否则，图中无回路。或者，使用深度优先遍历，按退出 DFS 过程的先后顺序记录下的顶点是逆拓扑序列。若在执行 DFS(v)未退出前出现顶点 u 到 v 的回边，则说明存在包含顶点 u 和顶点 v 的环。

对无向图，若边数大于等于顶点数时，则图中存在回路，否则，图中无回路。

2. 什么叫无向图的连通分量和生成树？

无向图的极大连通子图称为连通分量；图的极小连通子图称为生成树。

3. 对 n 个顶点的无向图，采用邻接表表示时，如何判别下列有关问题？(2004 年 820 简答题 6)

（1）图中有多少条边？
（2）任意两个顶点 i 和 j 是否有边相连？
（3）任意一个顶点的度是多少？

（1）图中的边数=邻接表链表结点总数的一半
（2）任意两顶点间是否有边相连，可看其中一个顶点的邻接表，若链表中的 adjvex 域有另一顶点位置的结点，则表示有边相连。
（3）任意一个顶点的度等于该顶点的链表中的结点个数。

4. 简述具有 n 个顶点、带权的无向图的邻接矩阵的特点。

（1）矩阵有 n 行 n 列

（2）矩阵是对称的

（3）当 vi 到 vj 有边时，ga.cost(i,j)为数值，否则为∞。

5. 对弗洛伊德（Floyd）每一对顶点之间的最短路径算法试回答：算法的时间复杂度；$A^{(k)}[i,j]$的含义；如何用该算法来判断图是否有回路。（2006 年 820 简答题 1）

算法的时间复杂度为 $O(n^3)$。

$A^{(k)}[i,j]$表示从 vi 到 vj 的中间顶点的序号不大于 k 的最短路径长度。

若主对角线元素有非无穷大值，即有回路。

6. 回答 AOV 网和 AOE 网能解决的主要问题。

AOV 网：

判定工程的可行性。有回路，整个工程就无法结束。

确定各项活动在整个工程执行中的先后顺序。

AOE 网：（

估算工程的最短工期（从源点到汇点至少需要多少时间）。

找出哪些活动是影响整个工程进展的关键。

7. 什么样的连通图其最小生成树是唯一的？

具有 n 个顶点，n-1 条边的连通图其生成树是唯一的；或者每条边的权值均不相同的连通图其最小生成树是唯一的。

8. 对一个连通网，用迪杰斯特拉算法求得指定顶点到其它 n-1 个顶点之间的最短路径，由这些最短路径边构成的生成树也一定是一颗最小生成树。上述说法是否正确？并举例说明。

不对。得到的生成树不一定是最小生成树。如：

v1	0	1	3	5
v2	1	0	∞	∞
v3	3	∞	0	4
v4	5	∞	4	0

以 v1 为源点用最短路径求得的生成树是{(v1,v2),(v1,v3),(v1,v4)}，生成树代价为 1+3+5=9；而用最小生成树算法求得的最小生成树为{(v1,v2),(v1,v3),(v3,v4)}，生成树代价为 1+3+4=8，所以上述说法是错误的。

9. 缩短某关键活动的持续时间是否总能缩短整个工程的工期？为什么？

不一定能。因为任何一项活动持续时间的改变都可能会导致关键路径的改变，所以只有在不改变关键路径的情况下，缩短关键活动的持续时间才有效；在有多条关键路径时，缩短不是所有关键路径共有的关键活动的持续时间，并不能缩短整个工期。

10. 什么样的 DAG 的拓扑排序结果是唯一的？

初始化无前驱顶点的顶点只有一个，每一次输出顶点后只有一个顶点无前驱。

11. 说明关键路径计算的思想。

关键路径是 AOE 网上从开始点到完成点路径长度最长的路径。假设开始点是 v1，从 v1 到 vi 的最长路径长度叫做事件 vi 的最早发生时间。这个时间决定了所有以 vi 为尾的弧所表示的活动的最早开始时间。用 e(i)表示活动 ai 的最早开始时间，再定义一个活动的最迟开始时间 l(i)，这是在不推迟整个工程完成的前提下，活动 ai 最迟必须开始进行的时间。两者之差 l(i)-e(i)意味着完成活动 ai 的时间余量。l(i)=e(i)的活动是关键活动，关键路径上的所有活动都是关键活动，计算关键路径就是求出所有活动的

最早开始时间 e(i)和最迟开始时间 l(i)，所有 l(i)=e(i)的活动构成了关键路径。

12. 下面有一种称为“破圈法”的求解最小生成树的方法：所谓“破圈法”就是“任取一圈，去掉圈上权最大的边”，反复执行这一步骤，知道没有圈为止。试判断这种方法是否正确？如果正确，请说明理由；如果不正确，举出反例。（注：圈就是回路。）

这种方法是正确的。由于经过“破圈法”后，最终没有回路，故一定可以构造出一颗生成树。下面证明这颗生成树是最小生成树。记“破圈法”生成的树为 T，假设 T 不是最小生成树，则必然存在最小生成树 T0，使得它与 T 的公共边尽可能地多，则将 T0 与 T 取并集，得到一个图，此图中必然将存在回路，由于“破圈法”的定义就是从回路中去除权最大的边，此时生成的 T 的权必然是最小的，这与原假设 T 不是最小生成树矛盾，从而 T 是最小生成树。

13. DFS 和 BFS 遍历各采用什么样的数据结构来暂存顶点？当要求连通图的生成树的高度最小，应采用何种遍历？（2004 年 820 简答题 1）

DFS 遍历采用栈来暂存顶点。BFS 采用队列来暂存顶点。当要求连通图的生成树的高度最小时，应采用 BFS 遍历。

第六章 查找

第一部分

1. 对查找表除进行查找操作外，可能还要进行向表中插入数据元素，或删除表中数据元素的表，称为动态查找表。（2010 年 820 填空题 5）

2. 设顺序线性表的长度为 30，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 6.5。（二分查找块，顺序查找块内元素）

3. Hash 函数构造方法：

（1）直接地址法（线性函数）

（2）数字分析法

（3）平方取中法

（4）除留余数法（一般 p 选择小于或者等于表长的最大素数）

（5）折叠法（移位叠加、间界叠加）

4. 处理冲突方法：

（1）开放地址法

1）线性探测法

2）平方探测法

3）再散列法

4）伪随机序列法

（2）链地址法

5. 填入哈希表中的元素个数与哈希表的长度的比值，称为哈希表的装填因子。（2008 年 820 填空题 3）

6. 为了能有效地应用 HASH 查找技术，必须解决的两个问题是构造一个好的 HASH 函数和确定解决冲突的方法。

7. 散列表中解决冲突的两种方法是开放地址法和链地址法。

8. 设某散列表的长度为 100，散列函数 H(k)=k%P，则 P 通常情况下最好选择 97。（比 P 小的最大素数）

9. 在各种查找算法中，平均查找长度与关键字个数 n 无关的方法是哈希查找方法。（2007 年 820 填空题 3）

10.折半查找要求数据元素有序，存储方式采用顺序存储。(2005 年 820 填空题 6)

第二部分

1.简述折半查找的优缺点。

优点：能高效率（$\log_2 n$ 次比较）进行查找。

缺点：不利于插入、删除（动态查找）。

2.简述哈希表查找方法。

哈希表查找方法不是通过关键字比较，而是通过对关键字进行某种计算，来实现关键字到存储地址的转换。

3.试分析线性探测法和二次探测法解决哈希地址冲突时，可能存在的不足。

线性探测法：容易产生“堆积”问题，这是由于当连续出现若干个同义词后，会产生连续多个地址被占用，而导致随后映射到这些地址的关键词由于前面的同义词堆积而产生冲突，尽管随后的这些关键词并没有同义词。

二次探测法：不能充分利用空间，它不能探测到哈希表上的所有单元，但至少能探测到一半单元。

4.试解释开放定址公式 $H_i=(H(key)+d_i)$ MOD m 中，H_i，$H(key)$，d_i 和 m 的含义。

H_i 为处理冲突过程中得到的地址序列，最后不发生冲突的地址为 key 在表中的地址；$H(key)$ 为哈希函数；d_i 为增量序列；m 为哈希表表长。

5.衡量 HASH 函数好坏的主要标准是什么？

（1）地址分布均匀（随机性好） （2）地址冲突少

第七章 排序

第一部分

1.任何一个借助“比较”进行排序的算法，在最坏情况下所需进行的比较次数至少为 $\lceil \log_2(n!) \rceil$ 。

2.借助于“比较”进行排序的算法，在最坏情况下能达到的最好的时间复杂度为 $O(n\log_2 n)$。

3.排序分类。

插入类排序：直接插入排序、折半插入排序、希尔排序

交换类排序：冒泡排序、快速排序

选择类排序：简单选择排序、堆排序

归并类排序：二路归并排序

基数类排序：基数排序

4.每次直接或通过基准元素间比较两个元素，若出现逆序排列就交换它们的位置，这种排序方法叫做交换排序。

5.堆排序关键两步为建立初始堆和筛选(调整为堆)。(2006 年 820 填空题 4)

6.实现基数排序算法时，最适合的数据结构是链队列。(2010 年 820 填空题 7)

7.经过一趟排序，能保证一个元素到达最终位置，有交换类(冒泡排序、快速排序)和选择类(简单选择排序、堆排序)排序。

8.排序方法的元素比较次数与原始序列无关的有简单选择排序和折半插入排序。

9.排序方法的排序趟数与原始序列有关的有交换类（冒泡排序、快速排序）排序。

10.基数排序算法不需要进行记录关键字间的比较。(2015 年 820 填空题 10)

第二部分

1. 什么是内部排序？什么是外部排序？

内部排序：排序期间，全部记录都存放在内存的排序方法。

外部排序：排序期间，全部记录已不能同时存放在内存；排序过程中，需要记录在内、外存之间移动。

2. 选择排序的速度与原始记录的顺序是无关的，试简述其原因。

采用 for 循环，固定了执行次数。设记录数为 n，选择排序需要进行 n-1 次遍历，第 i 次选择排序遍历剩下 n-i 个元素（执行次数固定，不因原始记录顺序变化而改变）每一次选择剩下元素中最小（最大）元素放入有序区，经过 n-1 次遍历将前 n-1 个最小（最大）元素排列有序，剩下一个元素自然有序，所以选择排序的速度与原始记录的顺序无关。

3. 简述快速排序算法思想。

在待排序的 n 个元素中任取一个元素（通常取第一个元素）作为基准，把该元素放入适当的位置后，数据序列被此元素划分成两部分，所有关键字比该元素关键字小的元素放置在前一部分，所有关键字比它大的元素放置在后一部分，并把该元素排在这两部分的中间（称该元素归位），这个过程称作一趟快速排序，之后对所有划分出来的两部分分别重复上述过程，直至每部分内只有一个元素或为空为止。

4. 简述堆排序算法思想。

首先将存放在 L[1…n]中的 n 个元素建成初始堆，由于堆本身的特点（以大顶堆为例），堆顶元素就是最大值。输出堆顶元素后，通常将堆底元素送入堆顶，此时根结点已不满足大顶堆的性质，堆被破坏，将堆顶元素向下调整使其继续保持大顶堆的性质，再输出堆顶元素。如此重复，直到堆中仅剩下一个元素为止。

5. 简述插入排序算法思想。

每趟将一个待排序的元素作为关键字，按照其关键字大小插入到已经排好的部分序列的适当位置上，直到排序完成。

6. 简述简单选择排序算法思想。

假设排序表为L[1…n],第 i 趟排序即从L[i…n]中选择关键字最小的元素与L(i)，交换，每一趟排序可以确定一个元素的最终位置，这样经过 n-1 趟排序就可以使整个排序表有序。

7. 简述希尔排序算法思想。

先将待排序表分割成若干个形如 L[i,i+d,i+2d,…,i+kd]的“特殊”子表，分别进行直接插入排序，当整个表中元素已呈“基本有序”时，再对全体记录进行一次直接插入排序。

8. 简述堆排序算法思想。

堆是 n 个元素的序列，先选一个关键字最大（最小）的记录，然后与序列中的最后一个记录交换，之后将序列中前 n-1 个记录重新调整为堆，再将堆顶记录和当前堆序列的最后一个记录交换，如此反复直至排序结束。

优点：堆排序只需要一个记录大小供交换使用的辅助空间，调整堆时子女仅和双亲比较。

9. 简述归并排序算法思想。

“归并”的含义是将两个或两个以上的有序表组合成一个新的有序表。假定待排序表含有 n 个记录，则可以看成是 n 个有序的子表，每个子表长度为 1，然后两两归并，得到 ⌈n/2⌉ 个长度为 2 或 1 的有序表；再两两归并，如此重复直到合并成一个长度为 n 的有序表为止，这种排序方法称为二路归并排序。

10. 简述基数排序算法思想。

基数排序是一种很特别的排序方法，它不是基于比较进行排序的，而是采用多关键字排序思想，借助“分配”和“收集”两种操作对单逻辑关键字进行排序。基数排序又分为最高位优先（MSD）排序和最低位优先（LSD）排序。以 r 为基数的最低位优先基数排序的过程：假设线性表由结点序列 a0,a1,…,an-1 构成，每个结点 aj 的关键字由 d 元组（k_j^{d-1}，k_j^{d-2}，…，k_j^1，k_j^0）组成，其中 $0 \leqslant k_j^i \leqslant r-1$（$0 \leqslant j < n, 0 \leqslant i \leqslant d-1$）。在排序过程中，使用 r 个队列 Q0，Q1，…，Qr-1。排序过程如下：

对 i=0，1，…，d-1，依次做一次“分配”和“收集”。

分配：开始时，把 Q0，Q1，…，Qr-1 各个队列置成空队列，然后依次考察线性表中的每一个结点 aj（j=0，1，…，n-1），如果 aj 的关键字 $k_j^i=k$，就把 aj 放进 Qk 队列中。

收集：把 Q0，Q1，…，Qr-1 各个队列中的结点依次首尾相接，得到新的结点序列，从而组成新的线性表。

11. 排序算法的比较。

算法种类	时间复杂度			空间复杂度	是否稳定
	最好情况	平均情况	最坏情况		
直接插入排序	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	是
冒泡排序	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	是
简单选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	否
希尔排序		和增量有关	$O(n^2)$	$O(1)$	否
快速排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$	$O(\log_2 n)$	否
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$	否
2-路归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	是
基数排序	$O(d(n+r))$	$O(d(n+r))$	$O(d(n+r))$	$O(r)$	是

12. 海量数据分布在 100 台电脑中，想个办法高效统计出所有数据的前 10 个最大关键字数据，并分析时间复杂度。

先分别求出每台电脑的前 10 个最大的关键字数据，再根据这 100 台电脑的前 10 个最大关键字数据，共 1000 个数据求出前 10 个最大关键字数据即可。具体分析如下：

（1）先求出每台电脑的前 10 个最大关键字数据。由于只需要求出部分数据，因此不需要对 n 个数据全部排序，采用部分排序算法即可，比如简单选择排序、堆排序、桶排序。

（2）求 n 个数据的前 k（这里 k=10）个最大关键字数据，当 k<<n 时，最佳的方法是将后面的 n-k 个数据依次与前面的 k 个数据的最小值比较，如果比最小值还小，则扔掉该数据，继续比较下一个数据，否则扔掉更小的数据，把这个新数加入，直到余下的 n-k 个数据都处理完。由于每次需要前 k 个元素调整为小顶堆（时间复杂度为 O(klogk)），余下元素依次与小顶堆根结点比较，比根结点小则扔掉，比根结点大则用当前值替换根结点并调整为小顶堆，时间复杂度为 O(logk)。所以总的最坏时间复杂度为:

O(klogk)+O((n-k)logk)=O(nlogk)

（3）再从 m（这里 m=100）台电脑的共 km（这里 km=1000）个数据中选择所有数据的前最大的 k 个，采用（1）类似的方法即可求出。即将一台电脑的小顶堆作为初始小顶堆，余下 m-1 台电脑的最大的 k 个元素依次与小顶堆的根结点比较，大于根结点则替换根结点并调整为小顶堆，直到余下的数据都处理完成，时间复杂度为 O((m-1)klogk)

（4）综合上面 3 步，最终选择小顶堆能够最快统计出所有数据的前 k（k=10）个最大关键字数据，总的最坏时间复杂度为：

O(nlogk)+O((m-1)klogk)=O((n+mk-k)logk)

如果 k<<km<<n，则 T(n)=O(n)。

13. 若输入数据存储在带头结点的双向循环链表中，下面各种排序算法是否仍然适用？为什么？

快速排序、直接插入排序、简单选择排序、堆排序。

快速排序适用。因为可以快速定位到第一个元素和最后一个元素结点，然后通过 1 个指针从头部向后移动，另外一个指针从尾部向前移动，逐一与基准元素进行比较并能够通过修改指针完成结点交换操作。

插入排序适用。因为可以方便地找到前驱后继和通过修改指针完成结点交换操作。

选择排序适用。因为只需要移动指针遍历链表并通过修改指针完成结点交换。

堆排序不适用。因为双向循环链表无法方便地找到完全二叉树的双亲与孩子结点。