**linked list**

**linear list**

**9.对于一个随机算法，其最坏情况下的运行时间等于运行时间期望值的常数倍。**

**T      F**

**虽然我们希望如此，因为这会使算法比较稳定，但大多数算法的运行时间期望值是最优和最差的情况的折中。**

**(neuDS)数据的物理结构是指数据在计算机中的实际存储形式。**

**T      F**

**28.存在一棵总共有2016个结点的二叉树，其中有16个结点只有一个孩子。**

**T      F**

**假设二叉树的结点拥有0个，1个，2个孩子的结点分别为N0，N1，N2。首先，需要了解，二叉树中度为2的结点数量比叶节点少一，那么N0+N1+N2=2016⇒N1+2N2+1=2016⇒N1+2N2=2015N0+N1+N2=2016⇒N1+2N2+1=2016⇒N1+2N2=2015，看来N1必须是奇数。**

**4.The best "worst-case time complexity" for any algorithm that sorts by comparisons only is:**

**A.O(logN)  
    B.O(N)  
    C.O(NlogN)  
    D.O(N2)**

**对于比较排序算法，堆排序和归并排序的最坏时间复杂度都是O(NlogN)。**

**7.Use simple insertion sort to sort 10 numbers from non-decreasing to non-increasing, the possible numbers of comparisons and movements are:**

**A.100, 100  
    B.100, 54  
    C.54, 63  
    D.45, 44**

**插入排序的比较次数为(n−1)+(n−2)+...+1=n(n−1)2(n−1)+(n−2)+...+1=n(n−1)2，交换次数一定比比较次数低，所以选择D。**

**12.The recurrent equations for the time complexities of programs P1 and P2 are:**

* **P1: T(1)=1, T(N)=T(N/2)+1;**
* **P2: T(1)=1, T(N)=2T(N/2)+1;**

**Then the correct conclusion about their time complexities is:**

**A.they are both O(logN)  
    B.O(logN) for P1, and O(N) for P2  
    C.they are both O(N)  
    D.O(logN) for P1, and O(NlogN) for P2**

**P1的加号左边经过logN次递归等式计算恒等于1，而右边每次递归等式都加1，经过logN次计算最后为logN，两边取大是logN；  
  
P2加号的左边经过logN次递归等式计算是N，右边每次递归等式都加1，经过logN次计算最后为logN，两边取大总体为N。**

**The recurrent equations for the time complexities of programs P1 and P2 are:**

* **P1: *T*(1)=1,*T*(*N*)=*T*(*N*/3)+1**
* **P2: *T*(1)=1,*T*(*N*)=3*T*(*N*/3)+1**

**Then the correct conclusion about their time complexities is:**

**(3分)**

****

**A.**

**they are both *O*(log*N*)**

****

**B.**

***O*(log*N*) for P1, *O*(*N*) for P2**

****

**C.**

**they are both *O*(*N*)**

****

**D.**

***O*(log*N*) for P1, *O*(*N*log*N*) for P2**

**线索二叉树**

**我们把这棵二叉树进行中序遍历后，将所有的空指针域中的rchild，改为指向它的后继结点。**

**如图6-10-3，我们把这棵二叉树进行中序遍历后，将所有的空指针域中的lchild，改为指向它的前驱结点。**

**.Given a 12×12 symmetric matrix M. If the upper triangular entries mi,j (1≤i≤j≤12) of M are stored row by row in a 1-dimentional array N (in C programming language). What is the index of m6,6 in N?**

**A.50  
    B.51  
    C.55  
    D.66**

**上三角每一行的第一个元素为mi,i，所以m6,6为第12+11+10+9+8+1=51个元素，因为数组下标从0开始，所以最后得到的下标为50。**

**.In-order traversal of a binary tree can be done iteratively. Given the stack operation sequence as the following:**

**复制代码**

**push(1), push(2), push(3), pop(), push(4), pop(), pop(), push(5), pop(), pop(), push(6), pop()**

**Which one of the following statements is TRUE?**

**A.3 and 5 are siblings  
    B.1 is the parent of 5  
    C.6 is the root  
    D.None of the above**

**这道题本质上讲的是中序遍历的非递归实现，根据给出的stack序列，能够构建出一个二叉树**

**中缀转后缀**

**从左到右开始扫描中缀表达式**

**遇到数字， 直接输出**

**遇到运算符**

**a.若为“(” 直接入栈**

**b.若为“)” 将符号栈中的元素依次出栈并输出, 直到 “(“, “(“只出栈, 不输出**

**c.若为其他符号, 将符号栈中的元素依次出栈并输出, 直到遇到比当前符号优先级更低的符号或者”(“。 将当前符号入栈。**

**扫描完后, 将栈中剩余符号依次输出**

**中缀转前缀**

**读取到加号，由于绿栈栈顶是乘号，优先级高，所以乘号弹出并压入红栈，再把加号压入绿栈**

**深度：把根的深度定为0，最下面的一点点数**

**高度：把底层的设为0，最高的一层层数**

**但是两个肯定都是一样的**

**2.5 完满二叉树(Full Binary Tree)**

**A Full Binary Tree (FBT) is a tree in which every node other than the leaves has two children.**

**换句话说，所有非叶子结点的度都是2。（只要你有孩子，你就必然是有两个孩子。）**

**2.4 完全二叉树(Complete Binary Tree)**

**A Complete Binary Tree （CBT) is a binary tree in which every level,**

**except possibly the last, is completely filled, and all nodes**

**are as far left as possible.**

**换句话说，完全二叉树从根结点到倒数第二层满足完美二叉树，最后一层可以不完全填充，其叶子结点都靠左对齐。**

**2.3 完美二叉树(Perfect Binary Tree)**

**A Perfect Binary Tree(PBT) is a tree with all leaf nodes at the same depth.   
All internal nodes have degree 2.**

**一个深度为k(>=-1)且有2^(k+1) - 1个结点的二叉树称为完美二叉树。 (注： 国内的数据结构教材大多翻译为"满二叉树")**

**A.判断欧拉通路是否存在的方法**

**有向图：图连通，有一个顶点出度大入度1，有一个顶点入度大出度1，其余都是出度=入度。**

**无向图：图连通，只有两个顶点是奇数度，其余都是偶数度的。**

**B.判断欧拉回路是否存在的方法**

**有向图：图连通，所有的顶点出度=入度。**

**无向图：图连通，所有顶点都是偶数度。**

**“Circular Queue” is defined to be a queue implemented by a circularly linked list or a circular array. “循环队列”被定义为由循环链表或循环数组实现的队列。 F**

**The sum of the degrees of all the vertices in a connected graph must be an even number. 连通图中所有顶点的度数之和必须是偶数。 T**

**In a connected graph, the number of edges must be greater than the number of vertices minus 1. 在连通图中，边的个数必须大于顶点的个数减1 F**

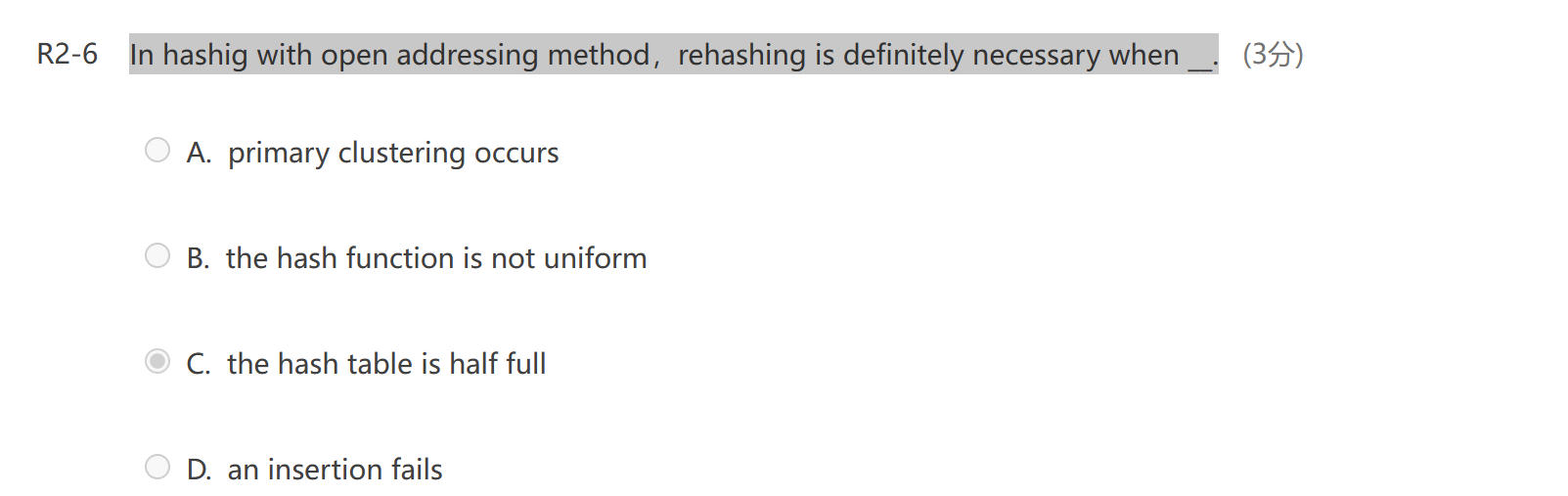
**If a graph is represented by adjacency lists, then the space taken depends only on the number of vertices, not the number of edges. 如果一个图是用邻接表表示的，那么所占用的空间只取决于顶点的数量，而不是边的数量。 F**

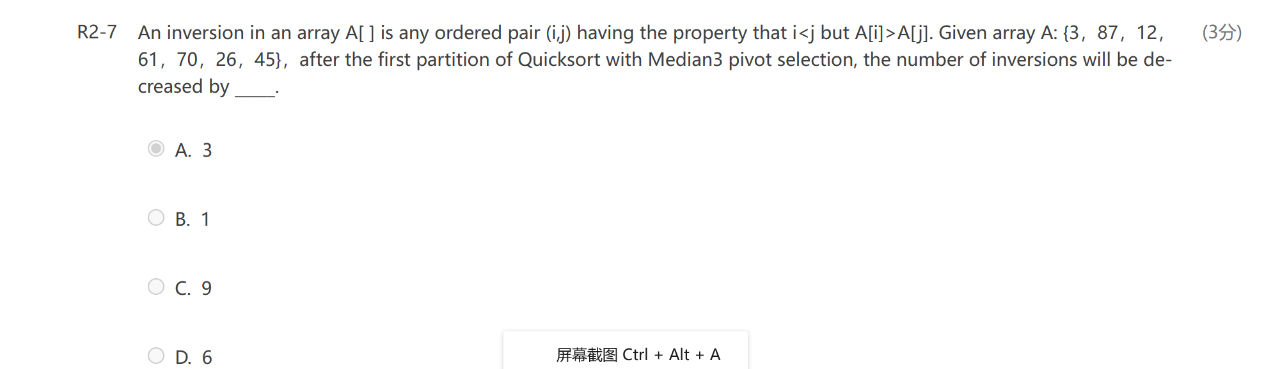
**If a graph is represented by an adjacency matrix, then the space taken depends only on the number of vertices, not the number of edges. 如果一个图是用邻接矩阵表示的，那么所占用的空间只取决于顶点的数量，而不是边的数量。 T**

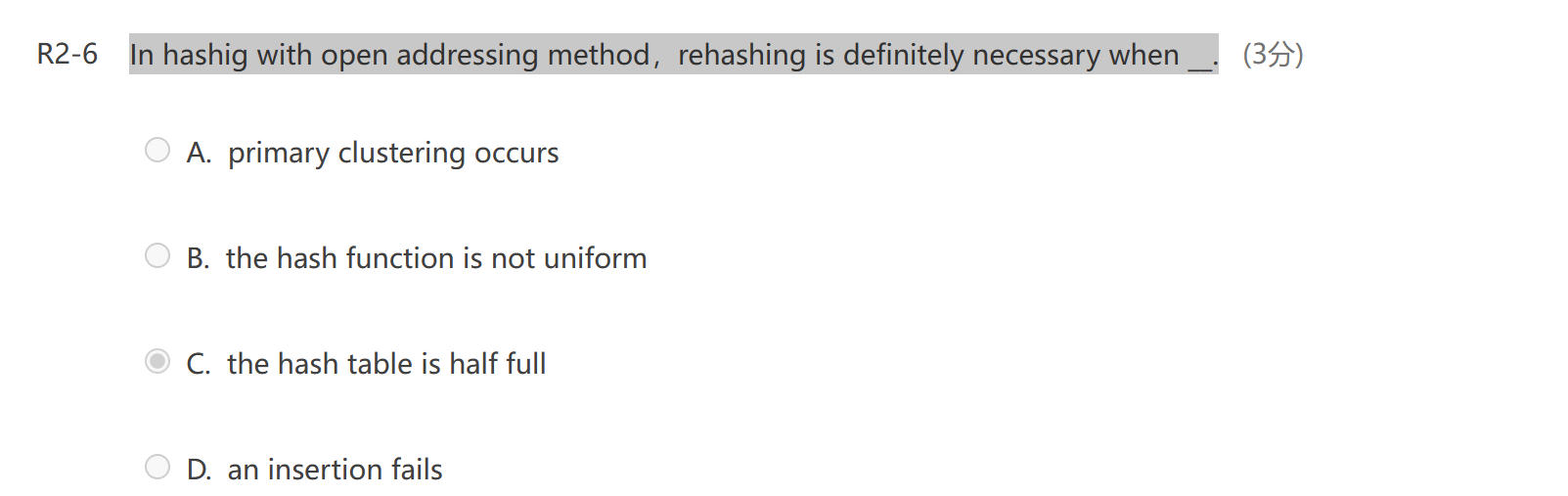
**In a directed graph, the sum of the in-degrees and out-degrees of all the vertices is twice the total number of edges. 在有向图中，所有顶点的入度和出度之和是总边数的两倍。 T**

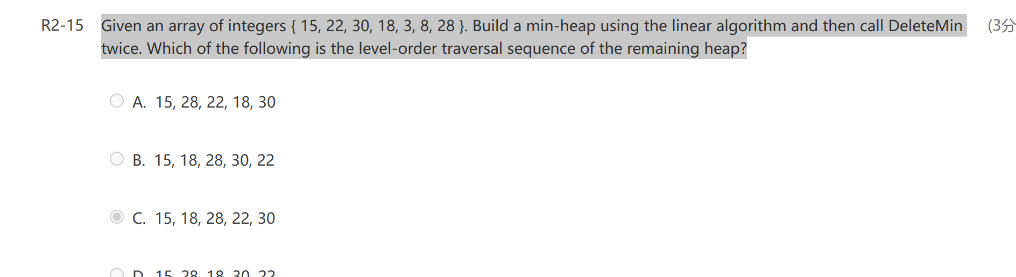
**The best “worst-case time complexity” for any algorithm that sorts by comparisons only must be O(NlogN). 对于任何只通过比较排序的算法来说，最好的“最坏情况时间复杂度”必须是O(NlogN)。 T**

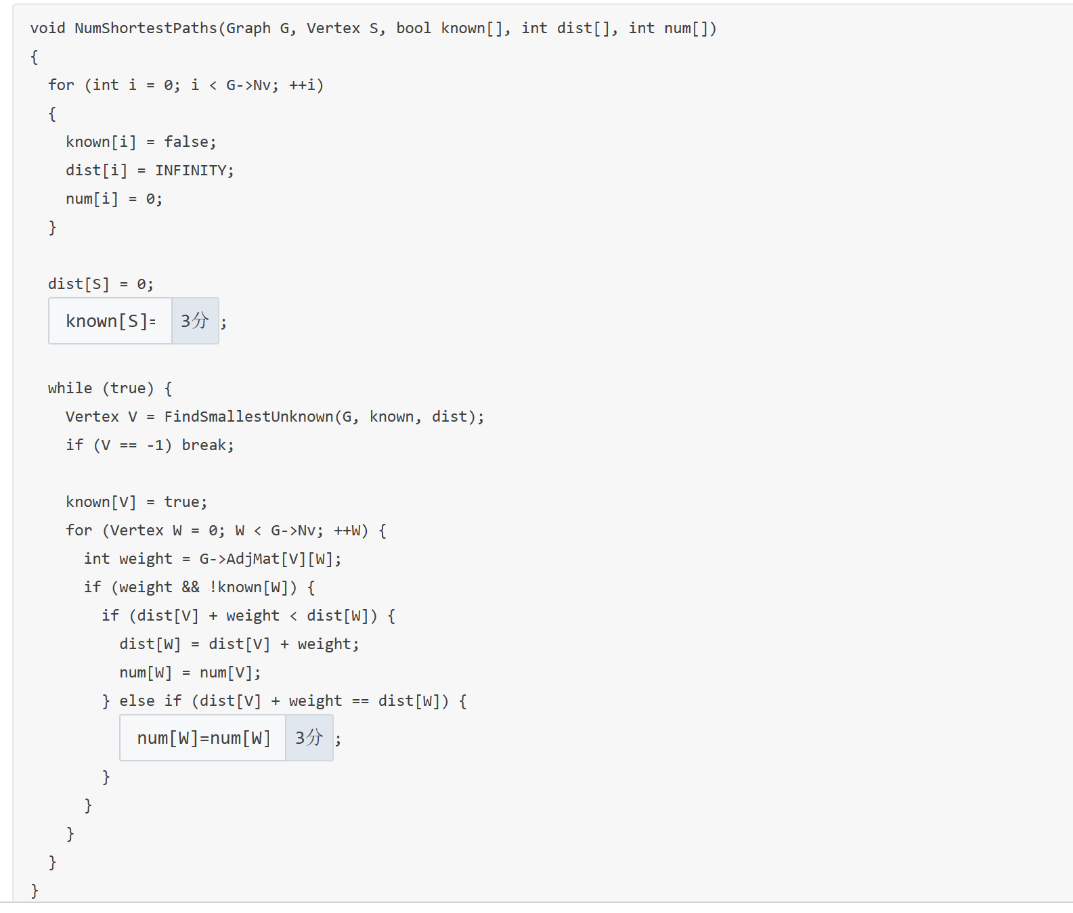
**Shell sort is stable. 希尔排序是稳定的 F**

****

****

****

****

****

**To sort N records by quick sort, the worst-case time complexity is Ω(NlogN). 对N条记录进行快速排序，最坏情况下时间复杂度为Ω(NlogN)。 T（表示最优复杂度）**

**If any NP-complete problem can be solved in polynomial time, then all the problems in NP can be solved in polynomial time. 如果任何NP完备问题都能在多项式时间内求解，那么所有NP问题都能在多项式时间内求解。 T**

**NP-complete问题：属于NP问题，且属于NP-hard问题。**

**NP-hard problems and NP-complete problems are the subsets of NP problems. NP困难问题和NP完全问题是NP问题的子集。 F**

**To prove the correctness of a greedy algorithm, we must prove that an optimal solution to the original problem always makes the greedy choice, so that the greedy choice is always safe. 为了证明贪婪算法的正确性，我们必须证明原问题的一个最优解总是做出贪婪选择，从而使贪婪选择总是安全的。 F**

**A binary tree that is not full cannot correspond to an optimal prefix code. 非满的二叉树不能对应于最优前缀代码。 T**

**For a connected graph, if there are exactly two vertices having odd degree, we can find an Euler tour that visits every vertex exactly once by starting from one of its odd-degree vertices. 对于连通图，如果恰好有两个顶点为奇数度，我们可以找到一个欧拉遍历，从其中一个奇数度顶点出发，恰好访问每个顶点一次。 F**

**对于无向图,若所有的点的度数均为偶数,那么存在欧拉回路.若有且只有两个点的度数为奇数,那么存在一条欧拉路径,且入口和出口正好是这两个点.**

**对于有向图,若所有的点的出度均等于入度,则存在欧拉回路.若有且仅有两个点的入度不等于出度,且点A的出度减入度=1,点B的入度减出度=1,那么存在一条欧拉路径,且起点为点A,终点为点B.**

**推论补充：欧拉路径（一笔画，如果起点终点相同，叫欧拉回路） 有欧拉环游（经过每条边恰好一次）的叫欧拉图**

**无向连通图 G 是欧拉图，当且仅当 G 不含奇数度结点( G 的所有结点度数为偶数)；**

**无向连通图 G 含有欧拉通路，当且仅当 G 有零个或两个奇数度的结点；**

**有向连通图 D 是欧拉图，当且仅当该图 D 中每个结点的入度=出度；**

**有向连通图 D 含有欧拉通路，当且仅当该图D 中除两个结点外，其余每个结点的入度=出度，且此两点满足 deg-(u)-deg+(v)=±1 。（起始点s的入度=出度-1，结束点t的出度=入度-1 或两个点的入度=出度）；**

**一个非平凡连通图是欧拉图当且仅当它的每条边属于奇数个环；**

**如果图G是欧拉图且 H = G-uv，则 H 有奇数个 u,v-迹仅在最后访问 v ；同时，在这一序列的 u,v-迹中，不是路径的迹的条数是偶数。**

**For a connected graph, if there are exactly two vertices having odd degree, we can find an Euler tour that visits every vertex exactly once by starting from one of its odd-degree vertices. F**

**The average run time and the extra space of Heapsort for sorting n elements are O(nlogn) and O(1), respectively. 对n个元素进行堆排序的平均运行时间和额外空间分别为O(nlogn)和O(1)。 T**

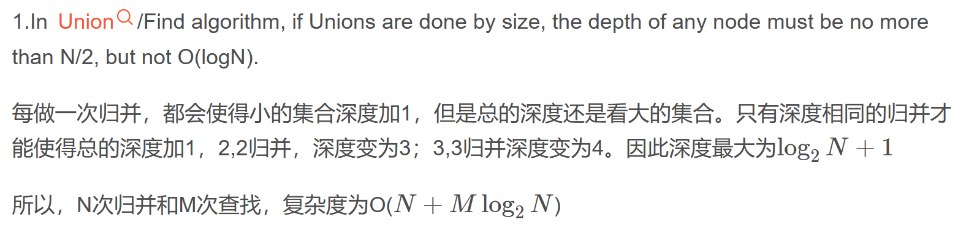
**The time comlexity of Binary Search will be the same no matter we store the elements in an array or a linked list. 无论我们将元素存储在数组还是链表中，二分搜索的时间复杂度都是相同的。 F（选择排序是一样的）**

**For the extra space taken by an internal sorting algorithm, we have: heap sort < quick sort < merge sort. 对于内部排序算法占用的额外空间，我们有:堆排序<快速排序<归并排序。 T**

**The number of leaf nodes in a binary tree is only related to the number of 2-degree nodes , i.e it has nothing to do with the number of 1-degree nodes. 二叉树的叶节点数只与2度节点数相关，与1度节点数无关。 T**

**During the sorting, processing every element which is not yet at its final position is called a “run”. To sort a list of integers using quick sort, it may reduce the total number of recursions by processing the large partion first in each run. 在排序过程中，处理尚未到达最终位置的每个元素称为“运行”。 为了使用快速排序对整数列表进行排序，它可以在每次运行中先处理大分区，从而减少递归的总数。 F**

**In Union/Find algorithm, if Unions are done by size, the depth of any node must be no more than N/2, but not O(logN). F**



**For a graph, if each vertex has an even degree or only two vertexes have odd degree, we can find a cycle that visits every edge exactly once . 对于一个图，如果每个顶点都是偶数度，或者只有两个顶点是奇数度，我们可以找到一个循环，它恰好访问每条边一次。F**

**For a graph, if each vertex has an even degree, we can find an Euler circuit that visits every vertex exactly once. 对于一个图，如果每个顶点都是偶数度，我们可以找到一个欧拉回路，它恰好访问每个顶点一次。 F**

 the average time complexity of the travesal of a binary tree with n nodes is O(n). n个节点的二叉树的平均遍历时间复杂度为O(n)。 F

**In the Activity Selection problem, consider any non-empty set of activities S, and let am be an activity in S with the latest start time. Then am must be included in some maximum-size subset of mutually compatible activities of S. 在活动选择问题中，考虑活动S的任何非空集合，设am为S中的一个活动，其最晚开始时间为。 那么am必须包含在S相互兼容的活动的某个最大大小子集中 T**

**If there are less than 20 inversions in an integer array, then Insertion Sort will be the best method among Quick Sort, Heap Sort and Insertion Sort. 如果在一个整数数组中有少于20个逆序，那么插入排序将是快速排序、堆排序和插入排序中最好的方法。 T**

**Prim’s algorithm is to grow the minimum spanning tree by adding one edge, and thus an associated vertex, to the tree in each stage. 普里姆的算法是通过在每一阶段向树中添加一条边，从而增加一个相关的顶点来增长最小生成树。 T**

**Kruskal’s algorithm is to maintain a forest and to merge two trees into one at each stage. Kruskal的算法是维护一个森林，并在每个阶段将两棵树合并成一棵树。 T**

An algorithm to check for balancing symbols in an expression uses a stack to store the symbols. 检查表达式中符号是否平衡的算法使栈（不是队列）来存储符号。T

**The best case time complexity of sorting algorithms based only on comparisons is at least O(NlogN). 在最好的情况下，只基于比较的排序算法的时间复杂度至少为O(NlogN)。 F（想想冒泡）**

**The best “worst-case time complexity” for any algorithm that sorts by comparisons only must be O(NlogN). 对于任何只通过比较排序的算法来说，最好的“最坏情况时间复杂度”必须是O(NlogN)。 T**

**An AVL tree that all the balance factors of non-leaf nodes are 0 must be a perfect binary tree. 所有非叶节点的平衡因子均为0的AVL树必定是一棵完美二叉树。T**

**In hashing, functions “insert” and “find” have the same time complexity. 在哈希中，函数“insert”和“find”具有相同的时间复杂度。 T**

**If quadratic probing is used to resolve collisions, then a new insertion must be successful if the size of the hash table is a prime. 如果使用二次探测来解决冲突，那么如果哈希表的大小是素数，则新的插入必须成功。 F**

哈希表的平均查找长度ASL = 总查找长度／关键字集合大小

注：总查找长度为所有关键字查找次数之和

哈希表的装填因子α = 填入表中的元素个数／哈希表的长度

To build a heap from N records, the best time complexity is: 从N条记录构建一个堆，最佳的时间复杂度是:  
B.O(N)

If a graph has N vertices, then it must have at most \_\_\_\_ edges. 如果一个图有N个顶点，那么它最多有\_\_\_\_条边。  
B.N(N−1)/2

**Among the algorithms that sort by comparisons only, which method has its worst-case time complexity bounded above by O(NlogN)? 在只进行比较排序的算法中，哪种方法的最坏情况时间复杂度在O(NlogN)以上?**

**B.merge sort**

To sort N records iteratively by merge sort, the number of runs is: 对N条记录进行归并排序，执行次数为:  
A.O(logN)

To sort N records by merge sort, the space complexity is: 对N条记录进行归并排序，空间复杂度为:  
B.O(N)

In a hash table, “synonyms”(同义词) means: 在一个哈希表,“同义词”(同义词)的意思是:  
B.two elements sharing the same hash value两个元素共享相同的哈希值

The average search time of searching a hash table with N elements is: 搜索包含N个元素的哈希表的平均搜索时间为:  
A.cannot be determined

**If a stack is used instead of a queue for the topological sort algorithm, does a different ordering result? 如果在拓扑排序算法中使用堆栈而不是队列，排序结果会不同吗?**

**C.Yes, sometimes.**

**What is the major difference among lists, stacks, and queues? 列表、堆栈和队列之间的主要区别是什么?**

**B.Stacks and queues are lists with insertion/deletion constraints**

**If a graph with n vertices and e edges is represented by adjacency lists, then the time complexity of the topological sort must be: 如果一个有n个顶点和e条边的图用邻接表表示，则拓扑排序的时间复杂度必须为:**

**B.O(n+e)**

To sort data files of size 10TB, the proper method is: 对于大小为10TB的数据文件，合适的排序方法是:  
C.merge sort

**Rehashing is required when an insertion to a hash table fails. Which of the following is NOT necessary to rehashing? 当插入哈希表失败时，需要重新哈希。 下面哪个选项对于重新哈希是不必要的?**

**A.change the collision resolution strategy .改变冲突解决策略**

The two structures suitable for representing a sparse matrix are: 适用于表示稀疏矩阵的两种结构是:  
A.triplets and multilists

**We can tell that there must be a lot of redundant calculations during the exhaustive search for the matrix multiplication problem, because the search work load is the Catalan number, yet there are only \_\_\_ different sub-problems M ij**

**我们可以看出，在穷举搜索矩阵乘法问题的过程中一定有大量的冗余计算，因为搜索工作量是加泰罗尼亚数，而只有\_\_\_个不同的子问题M ij**

**B.O(N^2)**

Ο，读音：big-oh；表示上界，**小于等于**。

Ω，读音：big omega、欧米伽；表示下界，**大于等于**。

Θ，读音：theta、西塔；既是上界也是下界，称为确界，等于。

**The average search time of searching a hash table with N elements is(2分)**

1. **O(1)**
2. **O(logN)**
3. **O(N)**
4. **cannot be determined**

线性插入：当load factor（此概念在上一章介绍）为0.5时，平均每次插入（等同于非成功寻找）需要位移2.5次，平均每次成功寻找需要位移1.5次。将load factor保证在0.5以下，那么时间是比较理想的。

平方探测插入：只有当load factor小于0.5且Hash表大小为质数时，才能保证每次插入都成功

