# 17-18DSA期末考答案

本答案由群笨猫自制，题目本身可能没什么参考意义，主要是给大家体验一下期末考卷子大概的题型，而且如果做出来跟笨猫的不一样，很可能是笨猫有问题，记得在群里说一下。

## 选择题

选择题一般是15-20分，前面的简单题1分，后面的困难题2分。

1.C　按逻辑来分就是线性和非线性，前者包含队列、栈等，后者包含树、图等
2.A　由于要存取一个指定序号的元素，所以顺序表最方便
3.A　显然是栈，做错了的私聊我，实验二给你扣5分
4.C　访问直接找序号即可，增加/删除还得移动一下，所以是O(n)
5.A
6.A　考虑一下连通图的定义，也就是从图任何一点能到其它点，所以我们把一个六边形砍掉一条边，依然是连通图，5条边就足够联通了
7.B　mid=（l+r）/2，第一次是56（5），第二次是23（2），第三次是34（3），第四次是45（4），对于长度为n的顺序表进行二分查找，最多需要$log_2 n$+1次（向下取整）
8.A　快排是先找个枢纽，然后把元素分别扔到枢纽的左右，按字典顺序来看，以ff为枢纽，B的bb不对，C的gc和da不对,D的eb不对，只能选A
9.A　注意到这题说的是有向图，所以500条边只对应500个非零元素，如果是无向图的话可以是1000
10.B　从最内层开始看，先把d/e转为de/，然后c+d/e就是cde/+，b*（c+d/e）就是bcde/+*，最后是abcde/+*+
11.C　这种题有两个思路，一是直接画个符合条件的树，然后数一下就行，反正单选答案唯一；二是利用树的性质来列方程计算，树中所有节点的度之和=节点总数-1，这题总的度是10，所以一共有11个节点，11-5=6
12.D　按顺序画个完全二叉树就知道了
13.D　没什么好说的，记得看看第八章ppt最后那几页，有汇总各种排序的时间复杂度和引入的辅助空间复杂度
14.A　拓扑排序要注意选取一个点的所有前置之后，才能选取该点

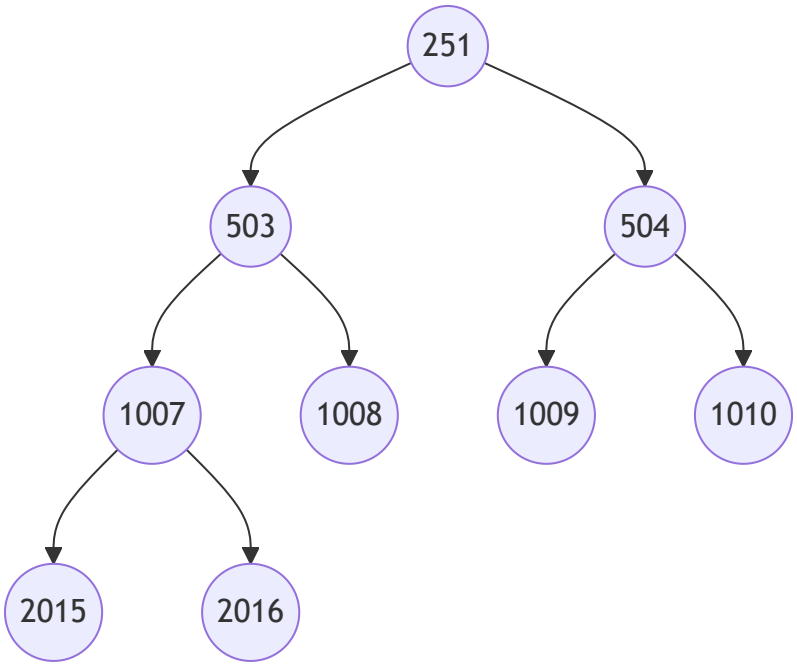
## 填空题

填空题一般是15-20分，前面的简单题每空1分，后面的困难题每空2分。
1.由于有头结点，所以链表为空的判断条件是L->next==NULL

2.希尔、快排、堆排都不稳定，基数排序是O（n），所以符合条件的只有归并排序。稳定性的定义：维持关键字相等的记录的先后顺序不变
3.13，注意空格也占空间，也别忘了字符串最后有个'\0'
4.Kruskal主要从边的角度考虑，所以更适合稀疏图
5.显然是O（n），你要写$\Theta(n)$也行
6.注意到这是个三维数组，按行优先是看最后一个下标，列优先是第一个下标，按照行优先的角度做这题，A[1][2][1]的元素是1x4x5+2x5+1=21，所以第21个元素是A[1][2][1],186/21=6，所以每个元素占6字节；342/6=57,57=2x4x5+3x5+2，所以是A[2][3][2]
7.从头开始检索替换，所以结果是"XYbXYba"
8.不能拓扑排序，说明图有环
9.先序是根左右，后序是左右根，m和n是兄弟（疾旋鼬）节点，所以如果后序时m在n后，说明n是左孩子，m是右孩子，换成先序的话m也在n后面
10.参考上面的公式，$log_2 9$+1=4
11.这题想考察的知识点是有什么方法可以区分循环队列的队满和队空，ppt上给的思路有：少用一个元素空间；设置长度length；设置full；设置empty。这四种选一个来操作即可，由于有length了，可以把n个元素空间全用上，不需要专门空一个了。

# 应用题

应用题一般是40-50分，每题分值不等
1.
(1)T是完全二叉树，1+2+...+1024=2047,2017介于2047和1023之间，所以T的最底层有1024-2047+2017=994个节点，S(2047)显然在右子树上
(2)S左侧有994-512-1=481个节点（注意别忘了减1）
(3)第三题这个讲法看起来比较奇怪，仔细一品，实际上是说画出一个包含S的有4层的子树。我用mermaid画一下




2.这题就纯作业题了

(1)线性探测法就是1234这样试，没什么好说的

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| HT | 11 | 22 | 46 | 13 | 01 | 70 | | | 41 | 31 | 30 |

(2)成功是针对数来的，删掉22之后（这里的意思应该是直接在第一题的HT里删就行了，而不是把22删掉之后重新排一个HT出来），13查2次，01查5次，70查2次，30查3次，所以ASL是15/8=1.875

(3)不成功是针对表的位置来的，ASL=(2+1+5+4+3+2+1+1+5+4+3)/11=31/11

3.

(1)第一次最大堆：95,85,49,72,34,40,43,58,65,20

把95选走之后，剩下的元素重新建最大堆：85,72,49,65,34,40,43,58,20

(2)2路归并排序，那就是两两分组，第一趟的结果为：34,85,43,72,40,95,49,58,20,65

第二趟的结果为：34, 43, 72, 85, 40, 49, 58, 95, 20, 65

(3)增量为3的希尔排序，那就是(34,72,49,20),(85,95,58),(43,40,65)这三组内部排，结果为：20, 58, 40, 34, 85, 43, 49, 95, 65, 72

4.一共n+（n-1）=2n-1个非零元素，刚好能扔进去。(0,n-1)对应0，(1,n-2)对应1，(1,n-1)对应2，(2,n-3)对应3，(2,n-2)对应4...显然(a,b)中a的系数为3，b的系数为1，接下来凑凑就行

$$k=\begin{cases}0 & i=0,j=n-1\\ 3i+j-n & otherwise\end{cases}$$

5.迪杰斯特拉算是常考点了，S表示这个点是否已找到最短路径，D表示当前的最短路径是多少，P则存

储路径的具体形式。分析一下，一开始逮捕的是v3，然后是v4，接着是v2，也就是说此时只有这三个节点的最短路径是被找到的，具体的表格结果如下：

| S | | D | | P | | | |
|---|---|---|---|---|---|---|---|
| 1 | | 0 | | - | | | |
| 0 | | 20 | | 0 | 3 | 4 | 1 |
| 1 | | 19 | | 0 | 3 | 4 | 2 |
| 1 | | 10 | | 0 | 3 | | |
| 1 | | 17 | | 0 | 3 | 4 | |
| 0 | | 25 | | 0 | 3 | 4 | 5 |

# 算法设计

算法设计估计是20分，一共2题，一题10分的样子，感觉算法设计题反而是最简单的

1.

(1)

```
List ListInit(int A[], int n) {
    List pHead = new List, p=pHead;
    for (int i = 0; i < n; ++i) {
        p->next = new List;
        p = p->next;
        p->val = A[i];
    }
    p->next = NULL;
    return pHead;
}
```

(2)

```
List reverse(List pHead) {
    List head = new List;
    head->next = NULL;
    while (pHead->next) {
        List current = new List;
        current->val = pHead->next->val;
        current->next = head->next;
        head->next = current;
        pHead = pHead->next;
    }
    return head;
}
```

2.

(1)二叉检索树：左<中<右

```
bool checkBST(bTree *pBTree) {
    if (pBTree->lchild && (pBTree->lchild->data >= pBTree->data || !check
 BST(pBTree->lchild))) return false;
    if (pBTree->rchild && (pBTree->rchild->data <= pBTree->data || !check
 BST(pBTree->rchild))) return false;
    return true;
 }
```

(2)等概率条件下，那就是总深度/节点数就行

```
int countDepthTotal(bTree *pBTree, int depth) {
    int ret = depth;
    if (pBTree->lchild) ret += countDepthTotal(pBTree->lchild, depth +
1);
    if (pBTree->rchild) ret += countDepthTotal(pBTree->rchild, depth +
1);
    return ret;
 }
 int countNodeTotal(bTree *pBTree) {
    int ret = 1;
    if (pBTree->lchild) ret += countNodeTotal(pBTree->lchild);
    if (pBTree->rchild) ret += countNodeTotal(pBTree->rchild);
    return ret;
 }
 double calcASL(bTree *pBTree) {
    return (double)countDepthTotal(pBTree,1) / countNodeTotal(pBTree)
 }
```