考试科目名称 数据结构 （A1 卷）

得分	

1、填空题。（每小题 **2** 分，本题满分 **20** 分）

(1) C++语言中，数组是按行优先顺序存储的，假设定义了一个二维数组 A[20][30]，每个元素占两个字节，其起始地址为 2140，则二维数组 A 的最后一个数据元素的地址为__2140+2*(30*20-1) = 3338(3338, 3339)__。

(2) 若 A，B 是两个单链表，链表长度分别为 n 和 m，其元素值递增有序，将 A 和 B 归并成一个按元素值递增有序的单链表，并要求辅助空间为 O(1)，则实现该功能的算法的时间复杂度为__O(m+n)__。

(3) 快速排序的平均时间复杂度是__n*lg n__。

(4) 假设有一个包含 9 个元素的最小堆，存放在数组 A 中，则一定比 A[3]大的元素有__2 (A[7],A[8])__个；一定比 A[3]小的元素有__2 (A[0],A[1])__个。（元素从第 0 个位置开始存放）

(5) 广义表(((A)),(B,C), D, ((A), ((E,F)))) 的长度是__4__，深度是__4__。

(6) 有 10 个元素的有序表，采用折半查找，需要比较 4 次才可找到的元素个数为__3__。

(7) 当两个栈共享一存储区时，栈利用一维数组 A[n]表示，两栈顶指针为 top[0]与 top[1]，则栈满时的判断条件为__top[0]+1=top[1] 或者 top[0] = top[1]+1__。

(8) 假设计算斐波那契数的函数 Fib(long n)定义如下：

```
long Fib(long n){
    if(n<=1) return n;
    else return Fib(n-1)+Fib(n-2)
}
```

计算 Fib(5)时的递归调用树（即指明函数调用关系的树）的高度是__4__。假设叶子结点所在的高度为 0。

(9) 完全二叉树按照层次次序，自顶向下，同层从左到右顺序从 0 开始编号时，编号为 i 的结点的左子结点的编号为__2*i+1__。

(10) 假设用子女—兄弟链表方式表示森林，对应的二叉树的根结点是 p，那么森林的第三棵树的根结点在二叉树中对应的结点是：__p->rightchild->rightchild__。假

leftchild	data	rightchild

得分	

2、选择题。（每小题 **2** 分，本题满分 **20** 分）

(1) 如果能够在只知道指针 p 指向链表中任一结点，不知道头指针的情况下，将结点*p 从链表中删除，则这个链表结构应该是：（ B,C ）（多选题）

A. 单链表 B. 循环链表 C. 双向链表 D. 带头结点的单链表

(2) 以下哪种矩阵压缩存储后会失去随机存取的功能?（ A ）

A. 稀疏矩阵 B. 对称矩阵 C. 对角矩阵 D. 上三角矩阵

(3) 下面哪一方法可以判断出一个有向图是否有环（回路）：（ B ）(选 A，B 也对)

A. 广度优先遍历 B. 拓扑排序 C. 求最短路径 D.求关键路径

(4) n 个结点的线索二叉树(没有头结点)上含有的线索数为（ B ）

A. 2n　　B. n－l　　C. n＋l　　D. n

(5) 循环队列存储在数组 A[0..m]中，则入队时队尾指针 rear 的操作为（ D ）

A. rear=rear+1　　B. rear=(rear+1) mod (m-1)

C. rear=(rear+1) mod m　　D. rear=(rear+1)mod(m+1)

(6) 使用加权规则得到改进的 Union 操作 WeightedUnion，其目的是：（ B ）

A. 提高 Union 操作的时间性能

B. 提高 Find 操作的时间性能

C. 减少 Union 操作的空间存储

D. 减少 Find 操作的空间存储

(7) 使用 Kruscal 算法求解最小生成树时，为了设计效率较高的算法，数据结构方面可以选择：（ A ）

A. 利用最小堆存储边

B. 利用栈存储结点

C. 利用二维数组存储结点

D. 利用并查集存储边

(8) 已知一算术表达式的后缀形式为 ABC*+DE/-，其前缀形式为：（ D ）

A. -A+B*C/DE　　B. -A+B*CD/E　　C. -+*ABC/DE　　D. -+A*BC/DE

(9) n 个关键码排序，如果选用直接插入排序方法，则元素的移动次数在最坏情况下可以达到(B)。

A. n*n/2　　B. n*(n-1)/2　　C. n/2　　D. (n-1)/2

(10) 关键路径是 AOE 网络中 A C 。（多选）

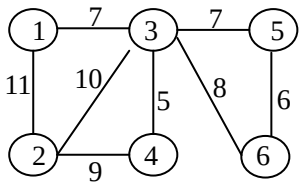
A. 从源点到汇点的最长路径　　B. 从源点到汇点的最短路径

C. 所有活动都是关键活动的路径　　D. 最短回路

得分	

3、简答题。（每小题 **5** 分，本题满分 **20** 分）

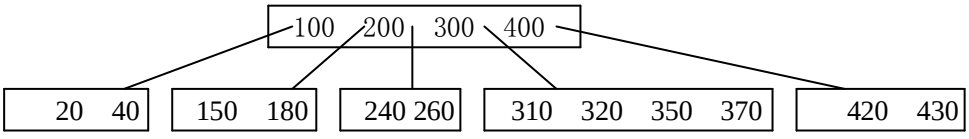
(1) 对如下无向图，按照 Dijkstra 算法，写出从顶点 1 到其它各个顶点的最短路径和最短路径长度。

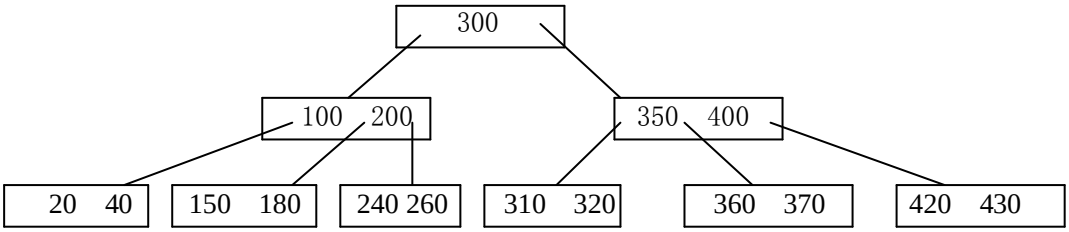


结点编号	1	2	3	4	5	6
路径长度	0	11	7	12	14	15

最短路径　1　1-2　1-3　1-3-4　1-3-5　1-3-6

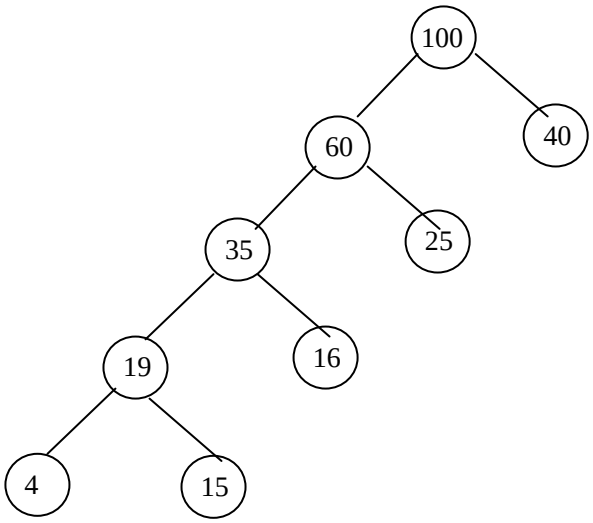
(2) 请画出在如下图所示的 5 阶 B 树中插入一个关键码 360 后得到的 B 树。





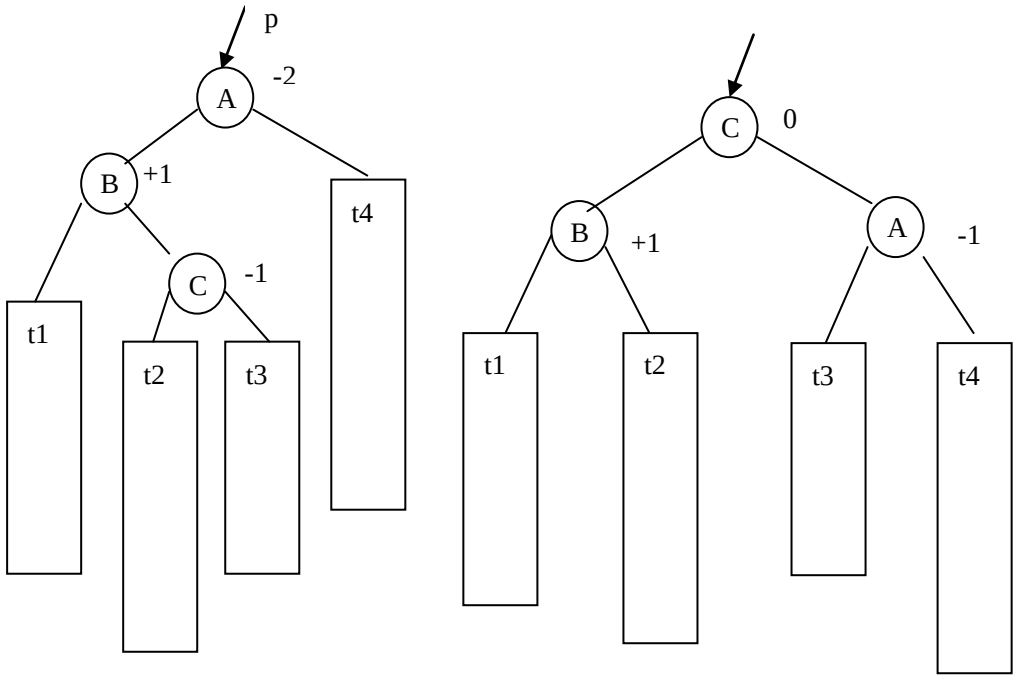
(3) 假设有权值集合{16,40,15,4,25}，给出相应的 huffman 树。假设某类信息由符号 a,b,c,d,e, 组成，而上面的权值分别是符号 a,b,c,d,e 的出现频率。请给出各个符号的 Huffman 编码。

Huffman 编码：
a: 001
b: 1
c: 0001
d: 0000
e: 01



注意：因为左右子树不同，
所以编码可以有多种，但是只要
其长度分别是 3，1，4，4，2；且
相互之间不形成前缀关系就
是正确的。

(4)在 AVL 树的插入操作中，假设插入一个结点后，当前节点 p 的平衡因子是-2，其左子结点的平衡因子是+1，左子结点的右子结点的平衡因子是-1。如图所示，请给出旋转调整之后的结构。



得分	

4、已知输入关键码序列为（10,90,20,60,78,35,42,31,15），地址区间为 0～11。

(1) 请设计一个散列函数，把上述关键码散到 0～11 中。画出散列表，冲突用线性探测法解决。（5 分）

散列函数为：

f(x) = x % 12　允许其它的散列函数

0	1	2	3	4	5	6	7	8	9	10	11
60/1	31/7		15/1			90/1	78/2	20/1	42/4	10/1	35/1

(2) 搜索元素 31 需要比较的次数是多少？（2 分）

7　（7-->8->9->10->11->0->1（成功））

(3) 计算在等概率情况下查找成功的平均查找长度 ASLsucc。（3 分）

(1+7+1+1+2+1+4+1+1) / 9 = 19/9

得分	

5、 程序设计题。（每小题 **15** 分，本题满分 **30** 分）

1. 设计一个算法，根据一棵二叉树的前序序列和中序序列，构造出这棵二叉树。 二叉树的结点都用字符表示。前序序列和中序序列都是字符串。二叉树的结点定义如下：

```
struct binTreeNode
{char data;
 binTreeNode *leftChild, *rightChild;
}
```

解：

```
TreeNode * tree(char *preorder, *midorder)
{
return
    TreeRecursive(preorder, 0, strlen(preorder)-1, midorder, 0, strlen(midorder)-1);
}

TreeNode * TreeREcursive(char *pre, int preSt, int preEnd,
                                    char* mid, int midSt, int midEnd)
{
    if (preEnd < preSt)
        return NULL;
    char  rt = pre[preSt];
    for(int j = midSt; j<=midEnd; j++)
        if(mid[j] == rt) break;
    if(j>midEnd){cout<<"Wrong input";    return NULL;}
    TreeNode root = new binTreeNode( );
    root->data = rt;
    int lLen = j - midSt;
    root->leftChild = treeRecursive(pre, preSt+1, preSt+lLen - 1,
```

```
                                        mid, midSt,midSt+lLen-1);
    root->rightChild=treeRecursive(pre, preSt+lLen+1, preEnd,
                                        mid, midSt+lLen+1, midEnd);
    return root;
}
```

要点在于根据 preOrder 的第一个字符，在 midOrder 中找到左右子树的分界；然后递归调用生成左右子树；做到这一点，就可以给出大部分分数。

可能有很多细节上不一样的地方，比如这里的 preEnd 指向的字符在相应的树中；但是有些人可能是 preEnd 指向下一个位置。或者传递参数时直接使用字符串传递。都是可以的。

如果使用别的办法，参考其效率，酌情给分。只要效率过得去，也可以得满分。但是纯粹枚举则得分不高。

2. 设计非递归算法实现图的深度优先遍历。（图用邻接表表示，已经定义了一个顺序栈 stack[top]，top 为栈顶指针，使用 visit(node)来表示对顶点 node 的访问。）

图的邻接表结构定义如下：

```
struct Edge   {
   int dest;
   Edge   *link; //下一条边链指针
}
struct Vertex {
  int   data;
  Edge   *adj; //边链表的头指针
   }
class Graph   {
   private:
   Vertex *Nodetable; //顶点表
   int cnt
}
```

解：

```
Graph : :  DFS(int v)//从 v 开始搜索；
{
    bool      visited[MAXVert];      访问标记
    Edge      nextEdge[MAXVert];     下一条边所指结点

    stack[0] = v;
    nextEdge[0] = graph.Nodetable[v].adj;
    top = 0;
    visited(stack[top]);     错了，visit(stack[top]);
```

```
        visited[stack[top]] = true;
        while(top >= 0)
        {
            while(nextEdge[top] && visited[nextEdge[top]->dest])
            //寻找下一个尚未访问的邻接节点
                nextEdge[top] = nextEdge[top]->link;
            if(nextEdge[top] != NULL)
            {
                int   nextVert = nextEdge[top]->dest;
                visite(nextVert);   //访问下一个邻接结点；保证了被压入栈中的顶点都被访问；
                visited[nextVert] = true;

                stack[top+1] = nextVert; //压栈，进入下一个结点；
                nextEdge[top+1] = Nodetable[nextVert].adj;

                nextEdge[top] = nextEdge[top]->link;   此步重要
                top ++;
            }
            else  top --;
        }
}
```

另一个数据结构试卷

得分	

1、填空题。（每小题 2 分，本题满分 **20** 分）

(1) 假设用一个一维数组 B 来按行存放一个对称矩阵 A 的下三角部分，那么访问 A 的第 i 行第 j 列元素的语句是：_____。(下标都从 0 开始)

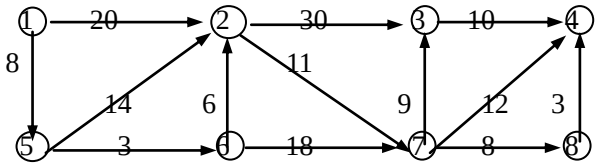
(2) 在单链表指针 P 所指结点后插入指针为 s 的结点操作是：_____。

(3) 线性表 $L=(a_0,a_1,a_2,\ldots a_{n-1})$用数组表示，假设删除表中任一元素的概率相同，则删除一个元素所需的平均移动次数为 _____。

(4) 在指针 L 指向一带头结点的双向循环链表，则判断该表中只有一个元素结点的条件是：_____ (设结点结构为 | lLink | data | rLink |)

(5)使用邻接矩阵表示图时，遍历一个顶点的所有相邻顶点的时间复杂度是：_____。(假设图的顶点个数为 n, 边的个数为 e)

(6) 已知带权有向图如下，使用 Dijkstra 算法求解从顶点 1 到达各顶点的最短距离时，该算法将按照_____（列出顶点顺序）的顺序给出各个顶点的距离。



(7) 假设一组关键码为（20，41，22，17，19，56，35），则用堆排序的方法建立的初始最

大堆为_____。

(8) KMP 模式匹配算法的时间复杂度是：_____。

(9) 设循环队列存放在数组 A[0..m-1]中，若用牺牲一个单元的办法区分队列满和队列空(设队尾指针为 rear, 队头指针为 first)，则判断队满的条件是：_____。.

(10) 对 n 个元素进行排序，如果用直接选择排序，所需的关键码比较次数最少为_____，如果用直接插入排序，则所需的关键码比较次数最少为_____。

得分	

2、选择题。（每小题 2 分，本题满分 **20** 分）

（1）假设使用转换后的二叉树来表示森林，那么对该二叉树的前序遍历对应于对森林的_____遍历。

A 先根次序遍历　B 后根次序遍历　C 广度优先遍历　D 以上都不是

(2) 假设我们将一个表达式表示为一棵二叉树。比如 a+b*c 对应的树的根为+，其左子树是一个叶子结点 a，右子树的根为*，左右子结点分别为 b 和 c。显然每个内部结点代表了一个运算。假设一个并行 CPU 有很多个 ALU 可以同时执行计算任务，且完成所有计算需要的时间都是一个时间单位。那么完成一个表达式计算的最短时间是_____。

A 树的高度　B 树的宽度　C 树的内部结点个数　D 树的分支数

(3) 假设在快速排序算法中总是选择被排序子序列的最后一个元素作为基准。那么这个算法的最坏情况出现在_____。

A 被排序的初始序列已经排好序时

B 被排序的初始序列是逆序时

C 被排序的初始序列呈现中间大，并逐次向两边减小的情况

D 以上都不是

(4) 散列方法中，线性探查法的“堆积”问题是指_____。

A 不同探查序列的关键码占据了可利用的空桶，使得寻找某一关键码需要经历很长的探查序列。

B 不同探查序列的关键码占据了可利用的空桶，使得插入某个关键码时找不到空桶。

C 不同探查序列的关键码占据了可利用的空桶，导致寻找某个关键码时出现错误。

D 不同探查序列的关键码占据了可利用的空桶，为了保证算法能够寻找到正确的关键码，不得不将装载因子限制在某个阀值之下。

(5) 在中序线索二叉树（使用 lTag，rTag）中，一个结点的中序后继是_____。（多选）

A 如果其 rTag 为 0，则后继是其 rightChild 所指结点的最左后代；

B 如果其 rTag 为 0，则后继是其 rightChild 所指结点；

C 如果其 rTag 为 1，则后继是其 rightChild 所指结点的最左后代；

D 如果其 rTag 为 1，则后继是其 rightChild 所指结点。

(6) 假设有一棵二叉树的结点前序排列是 1、2、3、4、5。下面的____序列不可能是这棵二叉树的中序排列。

A　3、2、1、4、5

B　2、3、1、4、5

C　2、1、4、5、3

D　2、1、5、3、4

(7) 有 11 个结点的 AVL 树的最大高度为_____。(假设叶结点的高度为 1，树的高度为根结点的高度)

A 3　　B 4　　C 5　　D 6

(8) 下面的排序算法中，时间复杂度不是 $O(n\log_2 n)$的算法是_____。（多选）

A 折半插入排序　　B 堆排序　　C 快速排序　　D 基数排序

(9) 使用 Prim 算法求解最小生成树时，使得算法效率较高的图的表示方式是_____。

A 邻接矩阵表示　　B 邻接表表示　　C 以上表示方法都一样

(10) 在 B 树的删除操作中，最坏情况下可能需要读写磁盘_____次。（假设内存工作区足够大，但操作之前各个结点都存放在磁盘上，B 树的高度为 h）

A　2h-2　　B 2h-1　　C 3h-1　　D 3h-2 次

得分	

3 、解答题。（每小题 5 分，本题满分 20 分）

(1) 使用一个 32 位整数以位向量集合的方式来表示 0 到 31 之间的整数的子集。请写出打印集合 s 中所有元素的代码。

(2) 假设一棵二叉树 T 的中序遍历序列和层次遍历（按层次递增顺序排列，同一层次自左向右）序列分别是 ABCDGEF 和 CAEBDFG。请写出该二叉树的后序遍历序列。

(3) 已知图的邻接矩阵为:

	V1	V2	V3	V4	V5	V6
V1	0	1	1	0	0	0
V2	0	0	1	1	0	0
V3	0	0	0	0	1	0
V4	0	0	1	0	0	1
V5	0	0	0	0	0	0
V6	0	0	0	0	1	0

请写出全部拓扑排序序列。

（4）对图 3－1 所示的 AOE 网络，求出每个活动的最早开始时间 e()和最迟开始时间 l()，并确定哪些活动是关键活动。

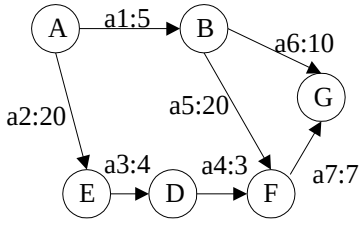


图3－1

得分	

4、以下算法实现了循环链表的逆转。

```
template<class Type>
void CList<Type>::Inverse ( ) {
CListNode<Type> *p,*q,*r;
if   (first->link= =first)   return;
p=first->link;q=p->link;
p->link=first;
while (_____)
{ r=q->link;q->link=p;
   p=q;q=r;
}
_____;
}
```

（1）请在算法的横线上填入适当的语句，每处只填一个语句或一个表达式；（6 分）

（2）请用大“O”表示法表示算法的时间复杂度和空间复杂度，假设链表长度为 n。（4 分）

得分	

5、以下给出一个排序算法，其中 n 是数组 A[ ]中元素总数。

```
template<class Type>
void sort (Type A[ ], int n) {
   int i = 1;
   while ( i<n-i+1)
   {min=max=i;
    for (j=i+1;j<=n-i+1;++j)
      { if   (A[j]<A[min]) min=j;
        else if (A[j]>A[max]) max=j;
      }
     if (min!=i)   Swap(A[min], A[j]); // Swap()函数的功能是交换两个数
     if (max!=n-i+1)
          { if (max= =i)   Swap(A[min], A[n-i+1]);
            else Swap(A[max], A[n-i+1]);
            }
      i++;
   }
}
```

(1) 阅读此算法，说明此算法的基本思想。（2 分）

(2) 对于下面给出的整数数组，请写出前 4 趟 **while** 结束时数组中数据的变化。（每行仅写出一个 **while** 循环的变化）（4 分）

步	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
	34	23	89	26	3	45	18	102	41
1									
2									
3									
4									

（3）请写出此排序算法最好情况下数据的移动次数。(2 分)

(4) 请用大“O”表示法表示算法的时间复杂度。（2 分）

得分	

6、 程序设计题。（每小题 10 分，本题满分 20 分）

好题

(1) 二叉树 T 以二叉链表的形式存储，设计算法返回二叉树 T 的先序序列的最后一个结点的指针（要求不用递归）。

二叉链表的结点结构为：

leftchild	data	rightchild

好题

(2) 已知一个有向图的邻接表表示，给出一个算法高效地计算出这个图的逆邻接表（即顶点 i 的边链表是所有进入该顶点的边）。要求算法的时间复杂度是 O(n+e)，其中 n 是图的顶点数，e 是图的边数。

答案:::::::::::::::::::::::

1、（1）B[i*(i+1)/2+j]

(2)s->link=p->link;p->link=s;

(3)(n-1)/2

(4)(L->rLink->rLink= =L || L->lLink->lLink= = L) && L->rLink!=L;

（写出 L->rLink->rLink= =L && L->rLink!=L;

或者 L->lLink->lLink= = L && L->rLink!=L;

之一即可）

(5)O(n)

(6)5 6 2 7 8 3 4

(7)56 41 20 17 19 22 35

(8)O(LengthP+LengthT)

(9)rear+1= =first;

(10)n(n-1)/2 ; n-1

2.

(1) A

(2) A

(3) B

(4) A

(5)A D （bc 也对？因为 rTag 为 0/1 时到底记录哪个值？）

(6)D

(7）C

(8）A D

(9）B

(10) D

2.

(1) long bits[]={ 0x00000001,0x00000002,0x00000004,0x00000008,
0x00000010,0x00000020,0x00000040,0x00000080
0x00000100,0x00000200,0x00000400,0x00000800,
0x00001000,0x00002000,0x00004000,0x00008000,
0x00010000,0x00020000,0x00040000,0x00080000,
0x00100000,0x00200000,0x00400000,0x00800000,
0x01000000,0x02000000,0x04000000,0x08000000,
0x10000000,0x20000000,0x40000000,0x80000000}

for (i=0;i<32;i++)

if s & bits[i] cout<<i;

也可以使用移位的方法测试第 i 位上是否为 1，如果为 1 则输出。

(2)BAGDFEC

(3) 1 2 4 3 6 5

1 2 4 6 3 5

(4)	A	B	E	D	F	G
Ve	0	5	20	24	27	34
Vl	0	7	20	24	27	34

	a1	a2	a3	a4	a5	a6	a7
e()	0	0	20	24	5	5	27
l()	2	0	20	24	7	24	27

关键活动为 a2 a3 a4 a7

4、（1）template<class Type>void List<Type>::Inverse ( ) {

ListNode<Type> *head=new ListNode<Type>( );//创建表头结点，其 link 域默认为
//NULL；

```
if (first->link= =head)    return;
p=first->link;q=p->link;
p->link=first;
while (q!=first)
{ r=q->link;q->link=p;
  p=q;q=r;
}
first->link=p;
}
```

(2) O(n) O(1)

5、(1)冒泡排序的改进算法，每趟排序都选出最小的和最大的将其放在前边和后边的位置上

（2）

步	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
	23	89	26	3	45	18	102	41
1	3	89	26	23	45	18	41	102
2	3	18	26	23	45	41	89	102
3	3	18	23	26	41	45	89	102
4	3	18	23	26	41	45	89	102

(3) 0

(4) $O(n^2)$

6、(1) while (t!=NULL) t->left != NULL || t->right != NULL

```
{ p=t;
  t=t->rightchild;
 if (t= =NULL) t=p->leftchild;
}
return t
```

(2)

```
for(int i = 0; i<numVertices; i++)
{
    Edge * p = NodeTable[i].adj;
    while(p!=NULL)
    {
        Edge *q = new Edge(i,p->weight);
        q->link = NodeTable[p->dest];
        NodeTable[p->dest].reverseAdj = q;
    }
}
```

感觉代码不够

评分标准：如果使用其它名字来表示结点数量（numVertices）,邻接表头指针（adj,） 逆邻接表头指针（reverseAdj）等，不扣分。

只要基本思想正确，小的格式错误不扣分。比如 Edge 的构造函数可以使用其它写法。只要能够表明构造得到的边结点的 dest 是当前结点号，就不扣分。

如果学生能够给出其它算法，确定正确后可得满分。如部分正确则酌情给分。