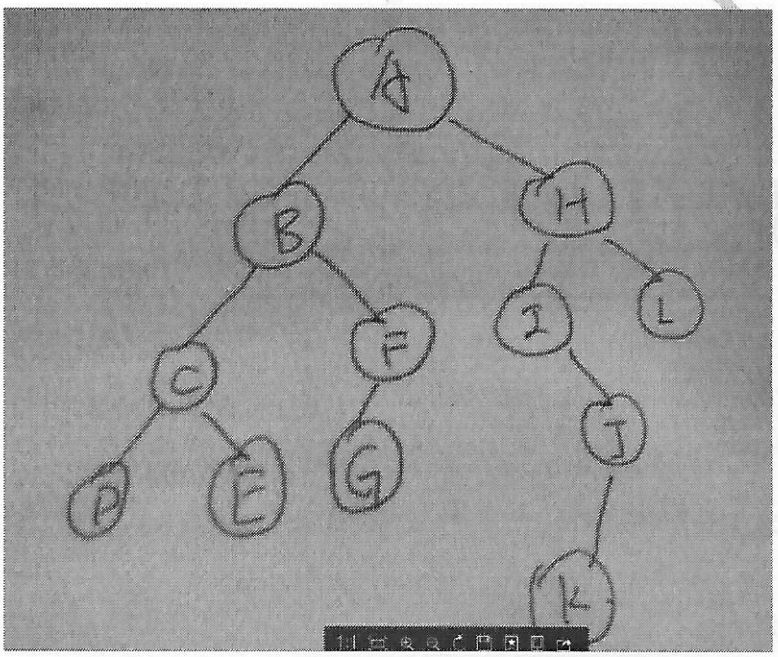
模拟试卷一答案

选择题
C B D A C

判断题

1. F 2.T 3.F 4.T 5.F 6.F 7.T 8.T 9.F 10 T
2.
填空题
1.L->next
2.s->next->prior=s;
3.先入先出，后入后出（FIFO,LILO）.
4.2i+j-2
5.tail(head(head(A)))
6.42
7.任一结点都无右子树
8.n-1
9.(10,16,13,14,15)
10.n-1

解答题
1. 已知一棵二叉树的先序、中序遍历序列如下，画出该二叉树。
先序：**ABCDEFGHIJKL**
中序：**DCEBGFAIKJHL**



2.求拓扑序列
1 5 2 7 3 4 6 7 8
1 5 2 7 6 3 4 7 8
1 5 2 7 6 3 7 4 8
3.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
65		28		43	31	30	33	57	22	98	11	77	100

平均查找长度:（1*8+2+4+5+3）/13=22/13

4. 对下面的数据表，写出采用快速排序算法排序的每一趟的结果。
14 11 22 3 5 25 76 61 100 44 34 120
5 11 3 14 22 25 34 61 44 76 100 120
3 5 11 14 22 25 34 61 44 76 100 120
3 5 11 14 22 25 34 44 61 76 100 120

算法设计
1 已知递增有序的带头结点额单链表表示一类集合，设计算法以判断集合 A 是否是 B 的子集。若 A 是 B 的子集，返回 TRUE，否则返回 FALSE。

```
bool check(node *&L, node* &L2)
{
    node *pa=L->next, *pb=L2->next;
    while (pa != NULL&&pb!=NULL)
    {
        if (pa->data == pb->data){
            pa = pa->next;
            pb = pb->next;
        }else if (pa->data > pb->data)
            pb = pb->next;
         else{
            return false;
        }
    }
    if (pa == NULL)
        return true;
    else
        return false;
}
```

2. 设计算法按先序次序遍历先序线索二叉树。要求采用非递归形式，且不用栈。

```
void preorder(node *t)
{
    node *p = t;
    while(p != NULL)
    {
        visit(p);
        P = presuc(p);
    }
}
node * presuc(node *p)
{
    If(p->itag == 0)
        Return(p->lchild);
    Else
        Return(p->rchild);
}
```

3. 已知数组 A[n]中的元素类型为整型。设计算法将数组 A 调整为按除 3 所得的余数的大小次序来排列，即所有余数为 0 的元素在最前面，余数为 1 的元素在中间，余数为 2 的元素在后面。

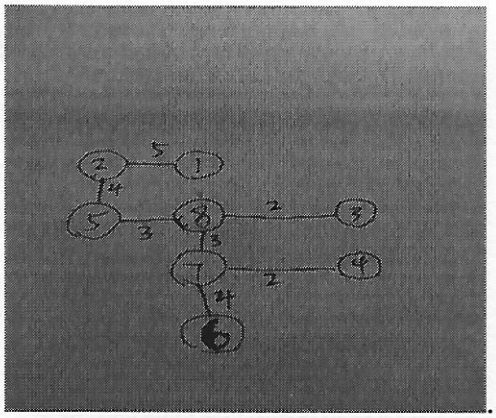
```
void sort(int a[],int n)
{
    int i = 1;
    do {
```

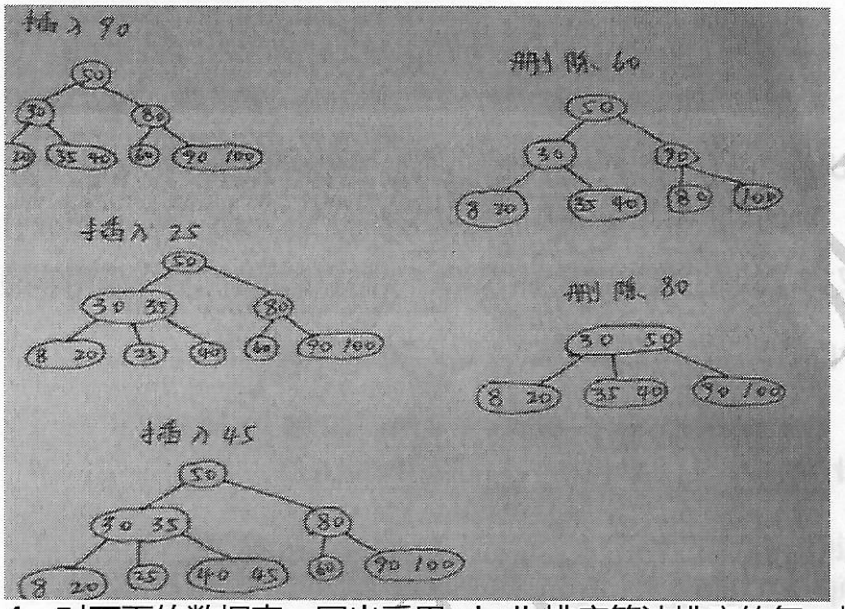
```
        bool checked = false;
        for ( int j = n-1; j >= i ; j--)
        {
            if (a[j] % 3 < a[j - 1] % 3)
            {
                int temp = a[j];
                a[j] = a[j - 1];
                a[j - 1] = temp;
                checked = true;
            }
        }
    i++;
    }while (checked && i <= n - 1)
}

4.
void outPreOrder(csNode*T,csNode*parents)
{
    if(T)
    {
            cout<<"("<<parents->data<<","<<T->data<<")";
            outPreOrder(T->firstChild,T);
            outPreOrder(T->nextbrother,T);
    }
}
5.
bool visited[maxnum];
void check(graph &g,int v)
{
        int connum = dfs_v(g, v, false);
        int connum_=dfs_v(g, v,true);
        if (connum_ > connum)
                cout << "是关节点" << endl;
        else cout << "不是关节点" << endl;
}
int dfs_v(graph &g, int v,bool b)
{
        int connum = 0;
        for (int id = 0; id < g.pointnum; id++)
                visited[id] = false;
        if(b)
                visited[v] = true;
        int v_ = firstadj(g, v);
        dfs_v(g, v_);
        connum++;
        for (int id = 1; id < g.pointnum; id++)
        {
                if (!visited[id])
                {
                        dfs_v(g, id);
                        connum++;
                }
        }
        return connum;
}
```

第二套

选择题
ACCBD
判断题
TTTFT　　TFFTT
填空题
1.log2^n
2.node *temp=P->next;p->next=temp->next;delete temp;
3.L->next!=L;L->next->next==L;
4.1118
5.tail(tail(head(head(A))))
6. 2^(k-2)
7.P->lchild==NULL&&p->rchild==NULL
8.n*(n-1)/2
9.5
10. nlog2^n
解答题：
1.算法 Print 及所引用的数组 A 的值如下，写出调用 Print(1)的运行结果（其中 n=15）。　　答案：DBHEAIFJCKGL
2.求图 3 的最小生成树





4. 对下面的数据表，写出采用 shell 排序算法排序的每一趟的结果，并标出数据移动的情况。

第一趟：1,11,8,14,15,2,5,66,100,22,34,20,44,76,125

移动情况：

A[7]->A[0],A[0]->A[7],A[9]->A[2],A[2]->A[9],A[10]->A[3],A[3]->A[10],
A[12]->A[5],A[5]->A[12],A[13]->A[6],A[6]->A[13],A[7]->A[14],A[0]->A[7
],A[14]->A[0]

第二趟：1,11,2,5,15,8,14,34,20,22,66,100,44,76,125

移动情况：

A[2]->A[5],A[5]->A[2],A[3]->A[6],A[6]->A[3]
A[7]->A[10],A[10]->A[7],A[8]->A[11],A[11]->A[8]

第三趟：1,2,5,8,11,14,15,20,22,34,44,66,76,100,125

移动情况：

A[1]->A[2],A[2]->A[1],A[2]->A[3],A[3]->A[2],A[4]->A[5],A[3]->A[4],A[5]-
>A[3]
A[5]->A[6],A[6]->A[5],A[7]->A[8],A[8]->A[7],A[8]->A[9],A[9]->A[8]
A[11]->A[12],A[10]->A[11],A[12]->A[10],A[12]->A[13],A[13]->A[12]

算法设计：

1.已知递增有序的两个单链表 A、B 分别存储了一个集合，设计算法实现求两个集合的并集的运算 A=AUB。

```
void together(node *L,node *L2)
{
   node *u=L,*u2=L2->next,*temp=NULL;
```

```
    while(u->next!=NULL&&u2!=NULL)
    {
       if(u->next->data>u2->data){
          temp=u->next;
          u->next=u2;
          u2=u2->next;
          u->next->next=temp;
       }
       else if(u->next->data==u2->data){
          u=u->next;
          u2=u2->next;
       }else {
       u=u->next;
       }
    }
    if(u->next==NULL)
       u->next=u2;
}
```

2.设计算法返回二叉树 T 的先序遍历序列中最后一个结点的指针，要求采用非递归形式，且不用栈。

```
node* DLR(node *L)
{
   node *temp=L;
   while(temp->lchild!=NULL&&temp->rchild!=NULL)
   {
      if(temp->rchild!=NULL)
         temp=temp->rchild;
      else
         temp=temp->lchild;
   }
   return temp;
}
```

3.设计算法输出给定哈夫曼树的哈夫曼编码

```
void hfm(node *L,char hfmCode[],int deep)
{
   if(L->lchild==NULL&&L->rchild==NULL){
       cout<<L->data<<":";
       for(int i=1;i<=deep;i++)
         cout<<hfmCode[i];
       cout<<endl;
   }else {
      hfmCode[deep]=0;
      hfm(L->lchild,hfmCode,deep+1);
      hfmCode[deep]=1;
      hfm(L->rchild,hfmCode,deep+1);
   }
}
```

4.对给定的有向图 G 及顶点 v0，设计算法以判断 G 是否是一棵以 v0 为根的有向树。若是有向树，返回 TRUE，否则，返回 FALSE。

```
 bool visited[];
bool isTree(Graph &G) {
        for (i = 0; i < G.VexNum; i++)
                visited[i] = False;
        int Vnum = 0;
        int Enum = 0;
        DFS(G, 1, Vnum, Enum, visited);
        if (Vnum == G.vexnum&&Enum == 2 * (G.vexnum - 1))
                return True;
        else
                return False;
}
void DFS(Graph &G, int v, int &Vnum, int &Enum, int visited[]) {
        visited[v] = True;
        Vnum++;
        int w = firstadj(G, v);
        while (w !  = -1) {
                Enum++;
                if (!visited[w])
                        DFS(G, w, Vnum, Enum, visited);
                w = nextadj(G, v, W);
        }
}
```

模拟试卷三

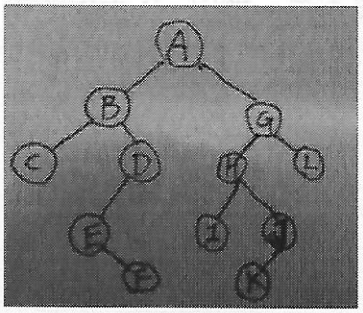
选择题：
DDDCB
判断题
√√×××
√×√×√
填空题
1.P->next->prior=p->prior;

- p->prior->next=p->next;
- delete p;

2.O(n^2)
3.S->next=p->next;

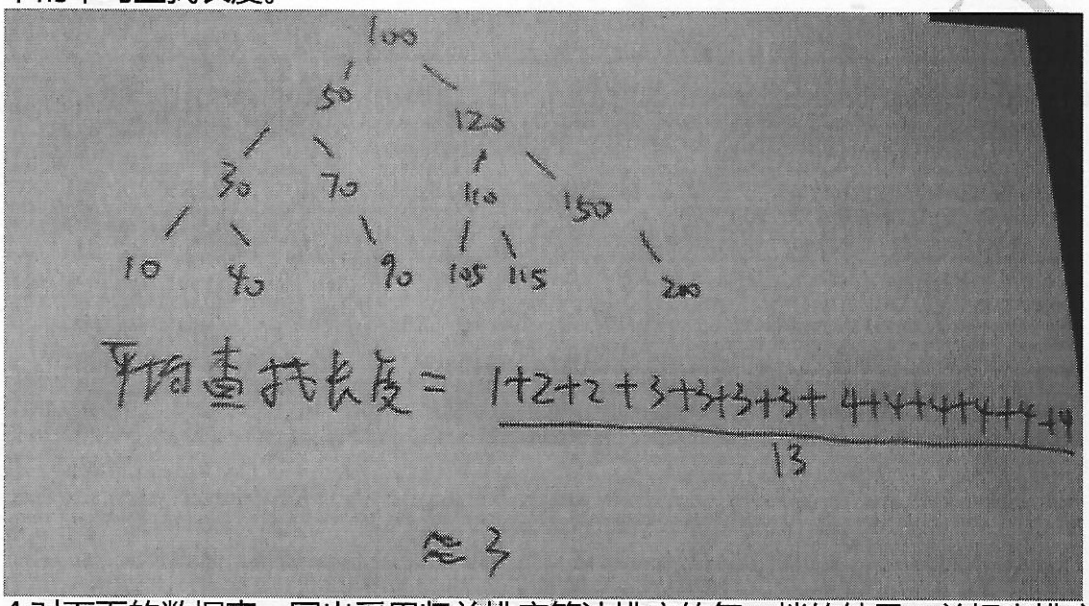
- p->next=s;

4.1150
5.(a b)
6.50
7.N+1
8.2(n-1)
9. 8,3,5,4
10.O(n^2)
四.解答题
1. 已知二叉树的先序、中序遍历序列如下，试构造出该二叉树。



2.略

3.以下面数据为输入序列构造二叉排序树，画出构造结果，并计算在等概率条件下的平均查找长度。



4.对下面的数据表，写出采用归并排序算法排序的每一趟的结果，并标出排序情况。

第一趟：11,125,22,34,15,44,66,76,8,100,14,20,2,5,1,80　获得 8 个长度为 2 的有序表

第二趟：11,22,34,125,15,44,66,76,8,14,20,100,1,2,5,80　获得 4 个长度为 4 的有序表

第三趟：11,15,22,34,44,66,76,125,1,2,5,8,14,20,80,100　获得 2 个长度为 8 的有序表

第四趟：1,2,5,8,11,14,15,20,22,34,44,66,76,80,100,125　获得 1 个长度为 16 的有序表

五.算法设计

1.设计算法实现运算 A=A-B∩C。

```
void f(seqList A,seqList B,seqList C)
{
   Node*ta=A;
   Node*tb=B;
   Node*tc=C;
   while(ta->next!=NULL)
   {
      if(ta->next->data<tb->next->data)
         ta=ta->next;
      else if(ta->next->data==tb->next->data)
      {
         while(tc->next->data<ta->next->data&&tc->next!=NULL)
```

```
            tc=tc->next;
        if(tc->next==NULL)
            break;
        if(tc->next!=NULL&&tc->next->data==tb->next->data)
        {
            Node*t=ta->next;
            ta->next=ta->next->next;
            delete t;
        }
    }
    else
    {
        while(tb->next!=NULL&&tb->next->data<ta->next->data)
            tb=tb->next;
        if(tb->next==NULL)
            break;
    }
  }
}
```

2.

```
int Search(elementType A[],int n,elementType x)
{
  for(int i=0;i<n-1;i++)
    if(x==A[i])
      rerurn i;
  return 0;
}
```

3.

```
void bfs(Graph G,cellType AdjMatrix[MaxVerNum][MaxVerNum])
{
  bool visited[MaxVerNum]={0};
  queue<int> q;
  q.push(1);
  while(!q.empty())
  {
    int t=q.front();
    q.pop();
    int t1=firstadj(G,t);
    while(t1)
    {
      visited[t1]=1;
      AdjMatrix[t][t1]=AdjMatrix[t1][t]=1;
      q.push(t1);
      t1=nextadj(G,t,t1);
    }
  }
}
```

4.

```
void PreOrderTraverse(BiNode *T,int x)
{
   if (T)
   {
      x++;
      cout <<"("<< T->data << "," << x <<")";
      PreOrderTraverse(T->lChild,x);
      PreOrderTraverse(T->rChild,x);
   }
}
```

第四套参考答案

选择题

CBBDD

判断题

TFFTT　　FTFFF

填空题

1.if(L->next!=NULL&&&L->next->next==NULL)

2.X->next=NULL;R->next=x;R=x;

3.先入后出，后入先出(FILO,LIFO)

4.1122

5.tail(tail(head(head(A))))

6.26

7.P->lTag==1&&p->rTag==1

8.1

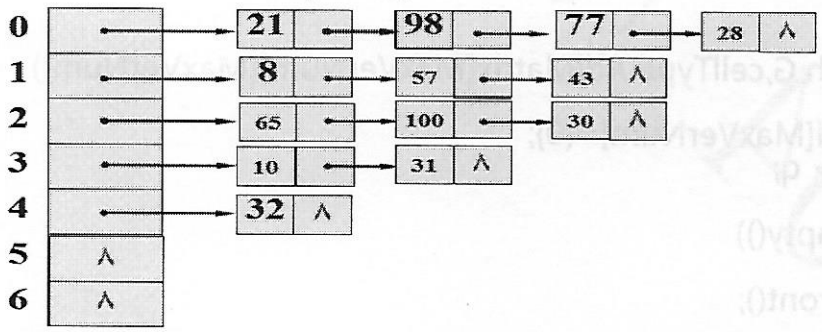
9.68

10.O(n^2)

解答题

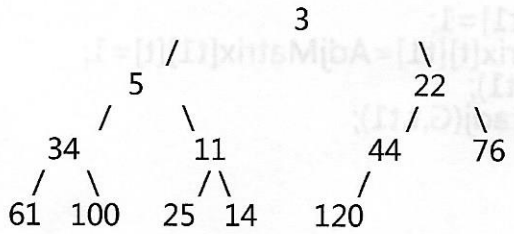
1.题目出错

2.ABCDEFGHIJKLMN



3. 平均查找长度=(1*5+2*4+3*3+4*1)/13=26/13=2

4.将下面数据表建成小根堆。



1.设计算法将链表分解成两个链表，其中一个存储原表中值为奇数的结点，另一链表中存储

值为偶数的结点。

```
void divideLists(node* L, node* &L1,node* &L2)
{
    L1 = new node; L2 = new node;
    L1->next = NULL; L2->next = NULL;
    node *u1, *n1, *u2,*n2;
    n1 = L1; n2 = L2;
    while (L->next)
    {
        if (L->next->data % 2 != 0)
         {
            u1 = new node;
            u1->data = L->next->data;
            u1->next = n1->next;
            n1->next = u1;
            n1 = u1;
            L = L->next;
        }else{
            u2 = new node;
            u2->data = L->next->data;
            u2->next = n2->next;
            n2->next = u2;
            n2 = u2;
            L = L->next;
        }
    }
}
```

2.设计算法求哈夫曼树的带权路径长度。

```
int hfm(node *&L)
{
    int sum=0;
    if(L->lchild!=NULL){
        sum+=L->data;
        sum+=hfm(L->lchild);
        sum+=hfm(L->rchild);
    }
    return sum;
}
```

3.

```
void PreOrderTraverse(BiNode *T,BiNode *pre)
{
      if (T)
      {
            cout <<"("<< T->data << "," << pre->data <<")";
            PreOrderTraverse(T->lChild,T);
            PreOrderTraverse(T->rChild,T->lChild);
       }
}
```

4.

```
 bool visited[maxnum];
void check(graph &g, int v)
{
          int connum = dfs_v(g, v, false);
          int connum_ = dfs_v(g, v, true);
          if (connum_ > connum)
                    cout << "删去后不是连通图" << endl;
          else cout << "删去后是连通图" << endl;
```

```
}
int dfs_v(graph &g, int v, bool b)
{
	int connum = 0;
	for (int id = 0; id < g.pointnum; id++)
		visited[id] = false;
	if (b)
		visited[v] = true;
	int v_ = firstadj(g, v,b);
	dfs(g, v_);
	connum++;
	for (int id = 1; id < g.pointnum; id++)
	{
		if (!visited[id]){
			dfs(g, id);
			connum++;
		}
	}
	return connum;
}
void dfs(graph g,int v)
{
	int w;
	visited[v]=true;
	visited[v] = true;
	w = firstadj(g, v);
	while (w != 0){
		if (visited[w] == false)
			dfs(g, w);
		w = nextadj(g, v, w);
	}
}
```