1. D
2. C
3. B
4. D
5. C
6. D
7. 归并排序

void Merging(int \**list1*,int *list1\_size*,int \**list2*,int *list2\_size*)

    {

        int i=0,j=0,k=0;

        int temp[10];

        while(i<*list1\_size*&&j<*list2\_size*)

        {

            if(*list1*[i]<*list2*[j])

            {

                temp[k++]=*list1*[i++];

            }

            else

            {

                temp[k++]=*list2*[j++];

            }

        }

        while(i<*list1\_size*)

        {

            temp[k++]=*list1*[i++];

        }

        while(j<*list2\_size*)

        {

            temp[k++]=*list2*[j++];

        }

        for(int m=0;m<*list1\_size*+*list2\_size*;m++)

        {

*list1*[m]=temp[m];

        }

    }

    void MergeSort(int *k*[],int *n*)

    {

        int \*list1=*k*;

        int list1\_size=*n*/2;

        int \*list2=*k*+*n*/2;

        int list2\_size=*n*-list1\_size;

if(list1\_size>1)

        MergeSort(list1, list1\_size);

if(list2\_size>1)

        MergeSort(list2, list2\_size);

        Merging(list1,list1\_size,list2,list2\_size);

    }

1. 基数排序

typedef int Type;

typedef struct Node

{

    Type Key;

    int Next;

}Node;

*//在Key为整形的条件下*

int GetDigit(Type *n*,int *index*)

{

    int i=0;

    for(int i=1;i<*index*;++i)

    {

*n*/=10;

    }

    return *n*%10;

}

void RadixSort(Node\* *array*,int *n*,int *d*)*//d是最大值的位数，array[0]中不存储数据*

{

    int i;

    for(i=0;i<*n*;i++)

    {

*array*[i].Next=i+1;

    }

*array*[*n*].Next=0;

    int front[10],rear[10];

    for(i=1;i<=*d*;++i)

    {

        int j;

        for(j=0;j<10;j++)

        {

            front[j]=rear[j]=0;

        }

        int p=*array*[0].Next;

        while(p)

        {

            int k=GetDigit(*array*[p].Key,i);

            if(front[k]==0)

            {

                front[k]=rear[k]=p;

            }

            else

            {

*array*[rear[k]].Next=p;

                rear[k]=p;

            }

            p=*array*[p].Next;

        }

        j=0;

        while(front[j]==0)

        {

            ++j;

        }

*array*[0].Next=front[j];

        int last=rear[j];

        for(j=j+1;j<10;++j)

        {

            if(front[j])

            {

*array*[last].Next=front[j];

                last=rear[j];

            }

        }

*array*[last].Next=0;

    }

}

1. void NameSort(char *name*[][31],int *n*)

{

    for(int i=0;i<*n*-1;i++)

    {

        for(int j=*n*-1;j>i;j--)

        {

            if(strcmp(*name*[j-1],*name*[j])>0)

            {

                char temp[31]={0};

                strcpy(temp,*name*[j-1]);

                strcpy(*name*[j-1],*name*[j]);

                strcpy(*name*[j],temp);

            }

        }

    }

}

1. const int N=10;

typedef struct RNode

{

    int key; *//数据域(也是关键字域)*

    struct RNode \*next; *//指针域*

} RNode, \*RList;

typedef RList R[N]; *//链表类型, 常变量N≥n*

typedef struct List

{

    RList front;

    RList rear;

}List;

void RadixSort(RList &*R*)

{

    RList p=*R*->next;

    List L[10];

    for(int i=0;i<3;++i)

    {

        for(int i=0;i<=9;i++)

        {

            L[i].front=NULL;

            L[i].rear=NULL;

        }

        while(p)

        {

            RList q=p;

            RList t=new RNode;

            t->next=NULL;

            t->key=q->key;

            int left=((t->key)/(int)(pow(10,i))%10);

            if(L[left].front==NULL)

            {

                L[left].front=L[left].rear=t;

            }

            else

            {

                L[left].rear->next=t;

            }

            L[left].rear=t;

            p=p->next;

        }

        int j=0;

        while(L[j].front==NULL)

            ++j;

*R*->next=L[j].front;

        p=L[j].front;

        int last=j;

        for(j=j+1;j<=9;j++)

        {

            if(L[j].front!=NULL)

            {

                L[last].rear->next=L[j].front;

                last=j;

            }

        }

        L[last].rear->next=NULL;

    }

}

1. A
2. B
3. (1) 426,87,225,61,170,503,897,908,653,512

(2) 61,87,170,225,426,512,897,503,653,908

14. typedef struct Node

{

Type key[100];

int length;

}SqList;

void TreeSort(SqList\* L)

{

int height = (int)ceil(log(L->length) / log(2)) + 1;//树的高度

int full = pow(2, height) - 1;//结点总数

int prefull = pow(2, height - 1) - 1;//除去叶子结点的结点数

Type\* t = new Type(full + 1);

for (int i = 1; i <= prefull; i++)

{

t[i] = INT\_MAX;//前面结点

}

for (int i = 0; i < L->length; i++)

{

t[prefull + i + 1] = L->key[i];

}

for (int i = prefull + L->length + 1; i <= full; i++)

{

t[i] = INT\_MAX;//多余结点

}

//为非叶子结点赋值

for (int i = prefull; i >= 1; i--)

{

t[i] = min(t[2 \* i], t[2 \* i + 1]);

}

for (int i = 0; i < L->length; i++)

{

L->key[i] = t[1];

int j = 1;

while (t[2 \* j] == t[1] || t[2 \* j + 1] == t[1])

{

j \*= 2;

if (t[j] != t[1])

{

j++;

}

if (2 \* j >= full)

{

break;

}

}

t[j] = INT\_MAX;

for (int k = j / 2; k >= 1; k /= 2)

{

t[k] = min(t[2 \* k], t[2 \* k + 1]);

}

}

}