上海工程技术大学

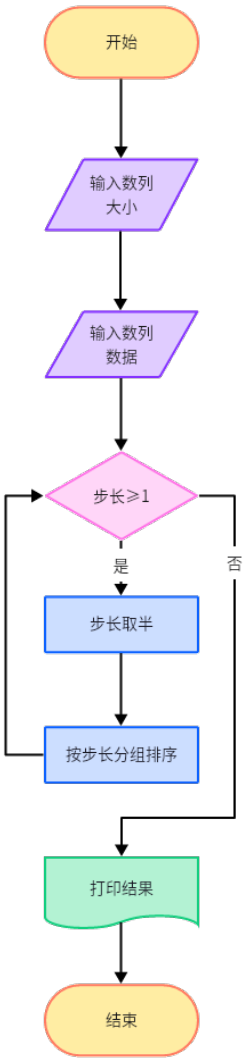
实 验 报 告

实验五　排序

给出一组关键字序列，分别给出用希尔排序、直接选择排序和快速排序算法从小到大排序的结果。

1. 希尔排序

1.1. 流程图



1.2. 代码

```
#include <cstdio>
#include <cstdlib>
typedef int ElementType;
typedef struct LNode *List;
struct LNode{
    ElementType Data[20];
    int Last;
};
List Creat(){
    List PtrL;
    PtrL = (List)malloc(sizeof(struct LNode));
    PtrL->Last = -1;
    return PtrL;
}
void Insert(ElementType X,int i,List PtrL){
    PtrL->Data[i]=X;
    PtrL->Last++;
}
void ShellSort(List PtrL){
    int i=0,j=0;
    int temp=0;
    int incre=0;
    for(incre=(PtrL->Last+1)/2;incre>0;incre/=2){
        for(i=incre;i<PtrL->Last+1;i++){
            temp=PtrL->Data[i];
            j=i-incre;
            while(j>=0 && temp<PtrL->Data[j]){
                PtrL->Data[j+incre]=PtrL->Data[j];
                j-=incre;
            }
            PtrL->Data[j+incre]=temp;
        }
    }
}
int main(){
    List SqList=Creat();
    int i, data=0, num;
    printf("序列中数据个数：\n");
    scanf("%d",&num);
    printf("请输入数据：\n");
    for(i=0;i<num;i++){
        scanf("%d",&data);
        Insert(data,i,SqList);
```

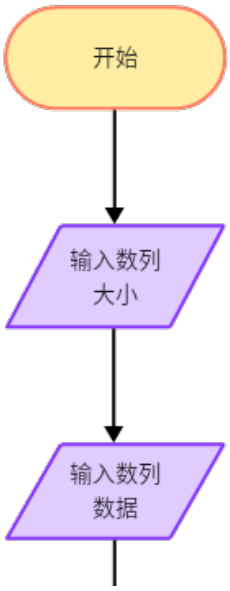
```
    }
    printf("排序前的序列为：\n");
    for(i=0;i<num;i++){
        printf("%d ",SqList->Data[i]);
    }
    printf("\nShell 排序后的序列为：\n");
    ShellSort(SqList);
    for(i=0;i<num;i++){
        printf("%d ",SqList->Data[i]);
    }
    return 0;
}
```

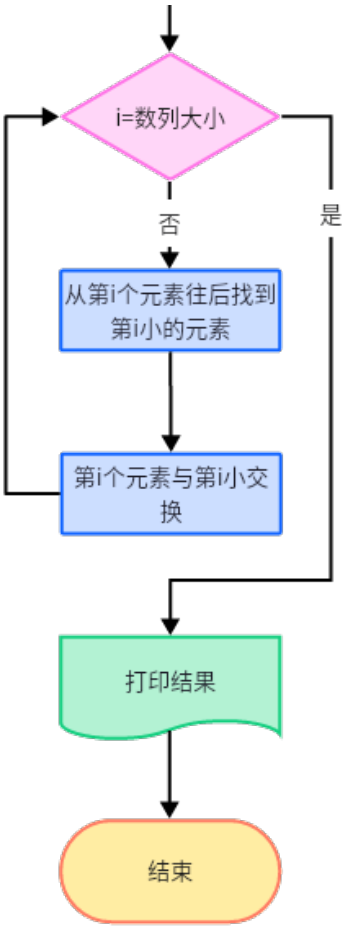
1.3. 结果

```
序列中数据个数：
6
请输入数据：
12 223 3425 23 786 1
排序前的序列为：
12 223 3425 23 786 1
Shell排序后的序列为：
1 12 23 223 786 3425
```

2. 直接选择排序

2.1. 流程图





2.2. 代码

```
#include <cstdio>
#include <cstdlib>
typedef int ElementType;
typedef struct LNode *List;
struct LNode{
    ElementType Data[20];
    int Last;
};
List Creat(){
    List PtrL;
    PtrL = (List)malloc(sizeof(struct LNode));
    PtrL->Last = -1;
    return PtrL;
}
void Insert(ElementType X,int i,List PtrL){
    PtrL->Data[i]=X;
    PtrL->Last++;
}
void StraightSelectionSort(List PtrL){
    int i=0,j=0;
```

```
    int temp=0;
    int index;
    for(i=0;i<PtrL->Last;i++){
        index=i;
        for(j=i+1;j<PtrL->Last+1;j++){
            if(PtrL->Data[index]>PtrL->Data[j]){
                index=j;
            }
        }
        temp=PtrL->Data[i];
        PtrL->Data[i]=PtrL->Data[index];
        PtrL->Data[index]=temp;
    }
}
int main(){
    List SqList=Creat();
    int i, data=0, num;
    printf("序列中数据个数：\n");
    scanf("%d",&num);
    printf("请输入数据：\n");
    for(i=0;i<num;i++){
        scanf("%d",&data);
        Insert(data,i,SqList);
    }
    printf("排序前的序列为：\n");
    for(i=0;i<num;i++){
        printf("%d ",SqList->Data[i]);
    }
    printf("\n 直接选择排序后的序列:\n");
    StraightSelectionSort(SqList);
    for(i=0;i<num;i++){
        printf("%d ",SqList->Data[i]);
    }
    return 0;
}
```

2.3. 结果

序列中数据个数：

5

请输入数据：

122 221 21 345 3

排序前的序列为：

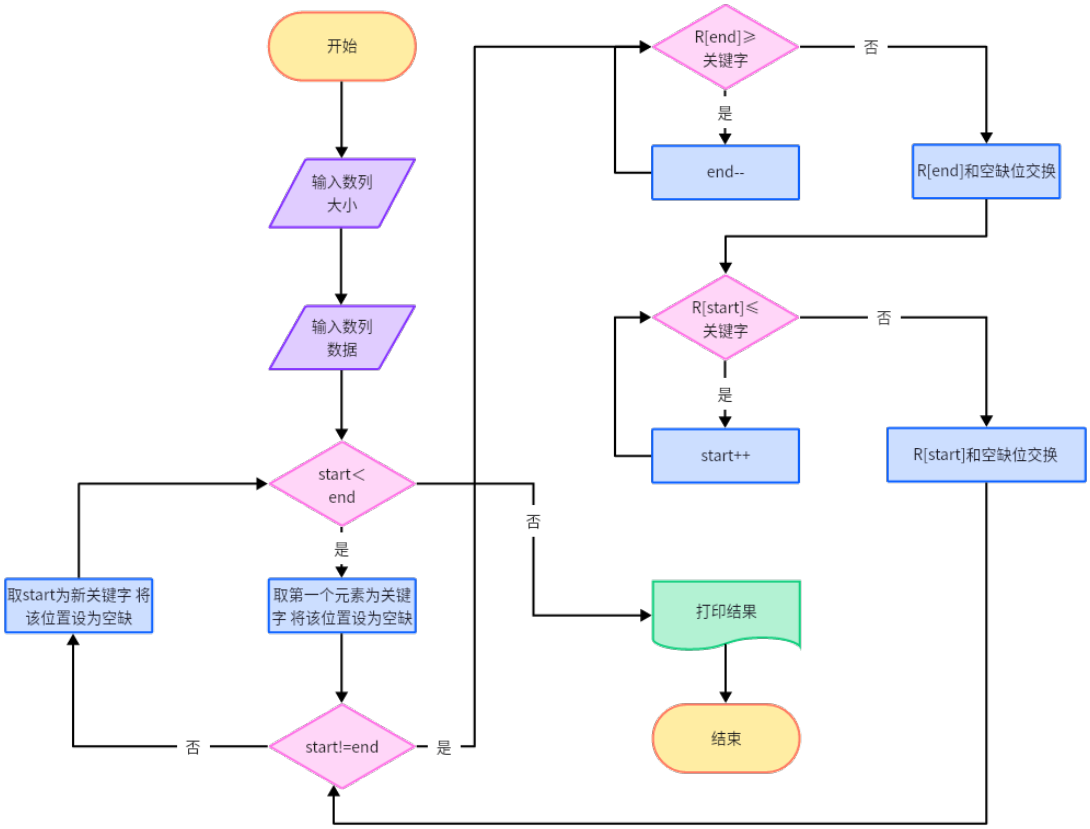
122 221 21 345 3

直接选择排序后的序列：

3 21 122 221 345

3. 快速排序

3.1. 流程图



3.2. 代码

```
#include <cstdio>
#include <cstdlib>
typedef int ElementType;
typedef struct LNode *List;
struct LNode{
    ElementType Data[20];
    int Last;
};
```

```
List Creat(){
    List PtrL;
    PtrL = (List)malloc(sizeof(struct LNode));
    PtrL->Last = -1;
    return PtrL;
}
void Insert(ElementType X,int i,List PtrL){
    PtrL->Data[i]=X;
    PtrL->Last++;
}
void QuickSort(List PtrL, int start, int end)
{
    if (start > end)
        return;
    int tmp = PtrL->Data[start];
    int i = start;
    int j = end;
    while (i != j) {
        while (PtrL->Data[j] >= tmp && j > i)
            j--;
        while (PtrL->Data[i] <= tmp && j > i)
            i++;
        if (j > i) {
            int t = PtrL->Data[i];
            PtrL->Data[i] = PtrL->Data[j];
            PtrL->Data[j] = t;
        }
    }
    PtrL->Data[start] = PtrL->Data[i];
    PtrL->Data[i] = tmp;
    QuickSort(PtrL, start, i - 1);
    QuickSort(PtrL, i + 1, end);
}
int main(){
    List SqList=Creat();
    int i, data=0, num;
    printf("序列中数据个数：\n");
    scanf("%d",&num);
    printf("请输入数据：\n");
    for(i=0;i<num;i++){
        scanf("%d",&data);
        Insert(data,i,SqList);
    }
```

```
    printf("排序前的序列为：\n");
    for(i=0;i<num;i++){
        printf("%d ",SqList->Data[i]);
    }
    printf("\n 快速排序后的序列:\n");
    QuickSort(SqList, 0, num - 1);
    for(i=0;i<num;i++){
        printf("%d ",SqList->Data[i]);
    }
    return 0;
}
```

3.3. 结果

```
序列中数据个数：
5
请输入数据：
12 13 45 234 1
排序前的序列为：
12 13 45 234 1
直接选择排序后的序列：
1 12 13 45 234
```

4. 实验小结

希尔排序是一种比较高效的排序算法，它通过将待排序的数组分成多个子序列来提高排序效率。希尔排序还可以针对不同的数据量进行优化，可以根据实际情况来确定它的步长，从而进一步提高排序效率。直接选择排序比较简单，但是它的时间复杂度较高，对于大规模数据的排序效率并不理想。快速排序是一种比较常用的排序算法，它通过选择一个基准元素，将待排序的数组分成两个子序列，然后再递归地对子序列进行排序。