**实验1 INSERTSORT和MERGESORT的对比**

1. **实验内容：**

掌握InsertSort和MergeSort算法的概念和基本思想。

1. **实验题：**

随机生成5000个元素的整型数组，分别使用InsertSort和MergeSort完成对数组的顺序排序，并比较两种算法的效率。

1. **实验步骤：**

|  |
| --- |
| 1. **程序设计说明：**(算法设计思路)   设计算法InsertSort() 插入排序和MergeSort() 归并排序  分别调用，并分别计算时间 |
| 1. **程序代码**（经调试正确的源程序）   #include <stdio.h>  #include <stdlib.h>  #include <time.h>  void InsertSort(int r[ ],int n)  {  int i,j,x;  for(i=1;i<n;i++)  {  x=r[i];  for(j=i-1;j>=0;j--)  if(r[j]>x)  r[j+1]=r[j];  else break;  r[j+1]=x;  }  }  void merge(int number[],int first,int last,int mid)  {  int number\_temp[5000]={0};  int i=first,j=mid+1,k;  for(k=0;k<=last-first;k++)  {  if (i==mid+1)  {  number\_temp[k]=number[j++];  continue;  }  if (j==last+1)  {  number\_temp[k]=number[i++];  continue;  }  if (number[i]<number[j]) number\_temp[k]=number[i++];  else number\_temp[k]=number[j++];  }  for (i=first,j=0;i<=last;i++,j++)  number[i] = number\_temp[j];  }  void merge\_sort(int number[],int first,int last)  {  int mid=0;  if(first<last)  {  mid=(first+last)/2;  merge\_sort(number,first,mid);  merge\_sort(number,mid+1,last);  merge(number,first,last,mid);  }  }  int main(void)  {  int i,a[5000],j,start,finsh;  printf("output the integers(1-5000):\n");  srand((unsigned)time(NULL));  for(i=0;i<5000;i++)  {  a[i]=1+rand()%100;  printf("%4d ",a[i]);  }  putchar('\n');  start=clock();  //merge\_sort(a,0,4999);//44  InsertSort(a,5000);//53  finsh=clock();  for(j=1;j<=5000;j++)  printf("%4d ",a[j-1]);  putchar('\n');  printf("Merge sort run time:\n");  printf("%4d ",finsh-start);  putchar('\n');  return 0;  } |
| 1. **程序运行结果（**测试数据和运行结果**）**   **插入排序时间44**  **归并排序时间 53** |
| **四、算法复杂性分析（**对所编写程序的时间复杂性和空间复杂性的分析**）** |
| **五、实验中遇到的问题及解决方法** |
| 1. **实验总结**   **归并排序时间复杂度小于插入排序** |