重庆大学本科学生实验项目任务书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验题目 | 排序算法实践 | | | |
| 实验时间 |  | 实验地点 | |  |
| 实验性质 | □验证性 √设计性 □综合性 | | | |
| 实验目的  1. 掌握各类排序算法的基本原理及时空复杂度分析方法  2. 训练使用排序算法，通过编程解决不同难度问题的实践能力 | | | | |
| 实验内容   1. 排序算法基本原理及时空复杂度分析方法 2. 在线编程解决3道不同难度的编程题，共40分   3． 在线编程时间（答题时长）150分钟，从开始答题时刻起算  4． 在线程序测评平台采用PTA (https://pintia.cn/)  5． 每人必须独立完成编程，可查阅教科书、PPT等资料，不得相互抄袭以及抄袭网上已有的程序  6． 实验课结束后，会对所有程序进行查重，如检测出有抄袭的程序，成绩计零分处理  注意：最后提交完整的实验报告，包括对每道题的算法思路、代码描述、复杂度分析等内容。 | | | | |
| 参考资料：   * Data Structures and Algorithm Analysis (C++ Version) Clifford A. Shaffer   + Introduction to Algorithms, 3rd Edition, MIT Press, T.H. Cormen, et al.   + 《数据结构（ C 语言版）》，严蔚敏，吴伟民编著，清华大学出版社 | | | | |
| 任务下达日期 年 月 日 | | | 完成日期 年 月 日 | |

说明：学院、专业、年级均填全称，如：计算机学院、计算机科学与技术、2018。

**《数据结构与算法》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | |  | | | **姓名** |  |
| **实验题目** | 排序算法实践 | | | | | |
| **实验时间** |  | | **实验地点** |  | | |
| **实验成绩** |  | | **实验性质** | **□验证性** √**设计性 □综合性** | | |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | |
| 一、实验目的  1. 掌握各类排序算法的基本原理及时空复杂度分析方法  2. 训练使用排序算法，通过编程解决不同难度问题的实践能力 | | | | | | |
| 二、实验项目内容    **实验课题1：**  题目内容：  **7-1 选择排序**  本题要求从键盘读入n个整数，对这些数做选择排序。输出选择排序每一步的结果和最终结果。  输入格式:  输入的第一行是一个正整数n，表示 在第二行中会有n个整数。  输出格式:  输出选择排序每一步的结果和最终结果。  解题思路：  每次从数组的无序部分取最小值所在位置下标，与数组无序部分的第一个数交换位置，此时该位置转变为数组有序部分的最后一位。重复操作至整个数组有序。  源代码（格式正确，对关键步骤加注释）：  #include<bits/stdc++.h>  using namespace std;  #define loop(i,start,end) for(int i=start;i<end;i++)  #define rep(i,start,end) for(int i=start;i<=end;i++)  #define rrep(i,end,start) for(int i=end;i>=start;i--)  typedef pair<int,int> P;  typedef long long ll;  typedef unsigned long long ull;  const int N=1e5+10,inf=0x3f3f3f3f,mod=1e9+7;  int n,a[N];  int main()  {  scanf("%d",&n);  for(int i=1;i<=n;i++)  scanf("%d",&a[i]);//读入数据  for(int i=1;i<n;i++)  {  int idx=i;  for(int j=i+1;j<=n;j++)  if(a[idx]>a[j])//向后查找无序部分的最小值并记录下标  idx=j;  swap(a[idx],a[i]);//交换,使下标为i的位置有序  printf("%s %d: ","step",i);  for(int i=1;i<=n;i++)  printf("%d ",a[i]);//打印每次排序的结果  putchar('\n');  }  printf("%s","sorted array: ");  for(int i=1;i<=n;i++)  printf("%d ",a[i]);//打印结果  return 0;  }  时间与空间复杂度分析  **时间复杂度：**  排序和打印部分:内、外循环复杂度均为O（n）  结果输出：Ω（n）  总体：O（n）\*（O（n）+O（n））+Ω（n）=O（n²）  **空间复杂度：**  临时下标index：O（1）  总体：O（1）  **实验课题2：**  题目内容：  **7-2 直接插入排序**  给定一个整数序列，请按非递减序输出采用直接插入排序的各趟排序后的结果。  输入格式:  测试数据有多组，处理到文件尾。每组测试数据第一行输入一个整数n（1≤n≤100），第二行输入n个整数。  输出格式:  对于每组测试，输出若干行，每行是一趟排序后的结果，每行的每两个数据之间留一个空格。  解题思路：  取数组无序部分的第一个数，与前一位数进行比较，如果小于前一位数则进行交换并重复操作，否则结束本次循环  源代码（格式正确，对关键步骤加注释）：  #include<bits/stdc++.h>  using namespace std;  int main()  {  int n,arr[100];  while(cin>>n)//读入元素个数  {  memset(arr,0,sizeof arr);  for(int i=0;i<n;i++)  scanf("%d",&arr[i]);//读入数组  for(int i=1;i<n;i++)  {  sort(arr,arr+i+1);//模拟插入排序，取数组无序部分的第一个数，与前一位数进行比较，如果小于前一位数则进行交换并重复操作，否则结束循环  for(int j=0;j<n-1;j++)  printf("%d ",arr[j]);//打印结果  printf("%d\n",arr[n-1]);  }  }  return 0;  }  时间与空间复杂度分析  **时间复杂度：**  排序和打印部分:内、外循环复杂度均为O（n）  总体：O（n）\*（O（n）+O（n））=O（n²）  **空间复杂度：**  总体：O（0）  **实验课题3：**  题目内容：  **7-3 贝茜放慢脚步**  奶牛贝茜正在参加冬季哞林匹克运动会的越野滑雪比赛。  她以每秒 1 米的速度出发。  但是，随着时间的推移，她变得越来越疲倦，她开始放慢脚步。  每次放慢脚步，贝茜的速度都会降低：减速一次后，她以每秒 1/2 米的速度移动，减速两次后，则以每秒 1/3 米的速度移动，依此类推。  你将被告知何时何地贝茜会减速。  当减速信息格式为：  T 17  意味着，贝茜在某个时间点减速，本例表示比赛开始第 17 秒贝茜减速。  当减速信息格式为：  D 10  意味着，贝茜在某个地点减速，本例表示在行进 10 米处减速。  给定 N 个减速信息，请计算贝茜滑完一千米需要多少秒。  将你的答案四舍五入到最接近的整数（ 0.5 向上舍入为 1）。  输入格式:  第一行包含整数 N(1≤N≤10000)。  接下来 N 行，每行描述一个减速信息，格式为 T x 或 D x。  无论哪种情况，x 都是一个整数，保证所有减速都在贝茜滑完一千米前发生。  可能同时发生多次减速，那么这会使得贝茜的速度一下子变慢很多。  所有减速信息不一定按顺序给出。  输出格式:  输出贝茜滑完一千米所需的总时间。  解题思路：  将定时减速的时间下标和定点减速的位置下标分别进行升序排序，每次取定时减速和定点减速中先发生的那一种情况进行处理，重复操作直至定点减速和定时减速两者中最多剩余一种未完成，单独完成该种后再记录以恒定的最终速度完成剩余路程的用时。  源代码（格式正确，对关键步骤加注释）：  #include<bits/stdc++.h>  using namespace std;  #define loop(i,start,end) for(int i=start;i<end;i++)  #define rep(i,start,end) for(int i=start;i<=end;i++)  #define rrep(i,end,start) for(int i=end;i>=start;i--)  typedef pair<int,int> P;  typedef long long ll;  typedef unsigned long long ull;  const int N=1e5+10,inf=0x3f3f3f3f,mod=1e9+7;  int n,cnt1=0,cnt2=0,cnt=1;  float a[10001],b[10001],s=0,t=0;//当前时间和路程  int main()  {  cin>>n;  for(int i=1;i<=n;i++)  {  char ch;  int x;  cin>>ch>>x;  if(ch=='T')  a[++cnt1]=x;//记录定时减速  else b[++cnt2]=x;//记录定点减速  }  sort(a+1,a+cnt1+1);  sort(b+1,b+cnt2+1);//升序排序  int i=1,j=1;  while(i<=cnt1&&j<=cnt2)//如果在当前的时间和位置，定时减速和定点减速都未结束  {  if(t+(b[j]-s)\*cnt<a[i])//下一次定点减速先于定时减速  {  t+=(b[j]-s)\*cnt;  s=b[j++];//转移至下一次定点减速所在位置，记录在当前速度下的用时  cnt++;//减速  }  else if(t+(b[j]-s)\*cnt>a[i])//下一次定点减速后于定时减速  {  s+=(a[i]-t)/cnt;  t=a[i++];//转移至下一次定时减速所在时间，记录在当前速度下的行进距离  cnt++;//减速  }  else//下一次定点减速和定时减速同时发生  {  cnt+=2;//减速两次  t=a[i++];//转移时间  s=b[j++];//转移距离  }  }  while(i<=cnt1)//如果只剩定时减速未完成  {  s+=(a[i]-t)/cnt;  t=a[i++];  cnt++;  }  while(j<=cnt2)//如果只剩定点减速未完成  {  t+=(b[j]-s)\*cnt;  s=b[j++];  cnt++;  }  t+=(1000-s)\*cnt;//记录全减速完成后至到达终点的耗时  cout<<round(t);//四舍五入  return 0;  }  时间与空间复杂度分析  **时间复杂度：**  排序：O（nlogn）  遍历定时减速和定点减速数组：每个元素在整个过程只进行一次访问，故为O（n）  总体：O（n）+O（nlogn）=O（n²）  **空间复杂度：**  记录每种减速的次数，总减速次数，时间，距离：O（1）  总体：O（1） | | | | | | |
| 三、思考题  用单链表存放一组数据并进行排序，插入排序、冒泡排序、归并排序，快速排序中，哪些算法能实现与顺序表相同的时间复杂度？简要说明排序过程。要求排序过程中，除输入的单链表外，算法的空间复杂度为O(1)。  **插入排序:可以**  时间复杂度为O（n²）。取双指针i,j分别指向单链表的前后两个节点,i=i->next,j=j->next，重复操作直至j==NULL或i->value<=当前元素值<=j->value,令i->next=temp,temp->next=j;  空间复杂度为O（1）。额外的空间开销为指针变量。  **冒泡排序：可以**  时间复杂度为O（n²）。取双指针i,j分别指向单链表的前后两个节点，若节点i的值大于节点j的值，则交换节点i,j的值，i=i->next,j=j->next；否则结束本次循环。每次循环结束后，单链表的后端变成有序状态。  空间复杂度为O（1）。额外的空间开销为指针变量。  **归并排序：可以**  时间复杂度为O（nlogn）。取快慢指针i=j=head,令i=i->next->next,j=j->next，当i指针遍历至单链表末端时，j指针指向链表的中央节点。归并层数仍为logn,每层对半分割数组操作的复杂度由O（1）变为O（n）,归并操作的复杂度仍为O（n），总的时间复杂度为O（logn）\*（O（n）+O（n））=O(nlogn)=O（logn）\*（O（1）+O（n））不变。  空间复杂度为O（1），每次合并时新建头节点为head，重复取两个子链表中值较小的头节点，接在head节点后方，最后删除头节点，返回新的头指针。  **快速排序：可以**  时间复杂度为O（nlogn）。取x=head,y=head->next,z=head->next->next,若y结点元素值大于基准值，则x,y,z指针同时后移一位；反之则将y节点的next指针指向head节点，head节点指向y节点，x节点的next指针指向z节点，此时单链表为y=head->...->x->z->...，令y指针指向z，z指针后移一位，x指针不动。重复操作。  空间复杂度为O（1），额外的空间开销为指针变量。 | | | | | | |