《算法分析与设计》

实 验 报 告

学 号

姓 名

年 级 2020

专 业 软件工程

院 系 计算机与人工智能学院

二O二二年三月

# 实验2.1采用快速排序对某序列进行排序

1. 实验目的

|  |
| --- |
| （1）正确理解算法的执行过程实验比较数据输入规模对算法执行时间的影响。  （2）实验比较不同算法对问题求解时间的影响。  （3）掌握算法时间效率的分析方法。 |

1. 实验任务

|  |
| --- |
| 1. 编写快速排序算法对应的程序，并计算序列长度为n时算法的运行时间。要求将运行时间保存到文件中。文件分成两行，第1行为n的值，第2 行为该算法的执行时间，以ms为单位。 2. 编写程序，计算n = 1000~100000,每次递增1000情况下算法的运 行时间，要求将运行时间保存到文件中。文件分为两行，第1行为n的值，用 逗号隔开，第2行为对应n的执行时间，以ms为单位，用逗号隔开。 |

1. 实验环境

|  |
| --- |
| * 1. 硬件环境  1. 计算机：拯救者R7000P 2020H 2. CPU: AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz 3. RAM：16GB    1. 软件环境 4. 操作系统：Windows11家庭中文版 5. 开发工具：Visual Studio Code |

1. 实验步骤及结果

|  |
| --- |
| * 1. 实验预习      1. 程序运行截图   任务（1）运行截图如图1所示；  任务（2）运行截图如图2所示。    图1 任务（1）对应的程序执行结果    图2 任务（2）对应的程序执行结果   * + 1. 程序代码   #include <stdio.h>  #include <time.h>  #include <windows.h>  #define MAXSIZE 100000  void Quick\_sort(int *s*[], int *l*, int *r*) *//快速排序,数组，基准数，个数*  {      if (*l* < *r*)      {          int i = *l*, j = *r*, x = *s*[*l*];          while (i < j)          {              while (i < j && *s*[j] >= x) *// 从右向左找第一个小于x的数*                  j--;              if (i < j)  *s*[i++] = *s*[j];              while (i < j && *s*[i] < x) *// 从左向右找第一个大于等于x的数*                  i++;              if (i < j)  *s*[j--] = *s*[i];          }  *s*[i] = x;          Quick\_sort(*s*, *l*, i - 1); *// 递归调用*          Quick\_sort(*s*, i + 1, *r*);      }  }  void RandomData(char \**FileName*, int *size*) *//写出随机生成数*  {      int i, j;      FILE \*fp;      srand((unsigned)time(NULL));      if ((fp = fopen(*FileName*, "w")) == NULL)      {          printf("No such file exist.");          exit(0);      }      for (i = 0; i < *size*; i++)      {          j = rand() % 99999;          fprintf(fp, "%d   ", j);  *//printf("%d   ", j);*      }      printf("\n");      fclose(fp);  }  void GetData(char \**FileName*, int *a*[])  {      FILE \*fp;      if ((fp = fopen(*FileName*, "r")) == NULL)      {          printf("No such file exist.");          exit(0);      }      else      {          int data, i = 0;          while (fscanf(fp, "%d", &data) != EOF)          {  *a*[i] = data;              i++;          }      }      fclose(fp);  }  void StroeNandTime(char \**FileName*, int *n*[],double *time*[],int *flag*) *//存储时间*  {      FILE \*fp;      if ((fp = fopen(*FileName*, "a+")) == NULL)      {          printf("No such file exist.");          exit(0);      }      if (*flag*==0)          fprintf(fp, "%10d\n%8fms", *n*[0], *time*[0] \* 1000);      else if(*flag*==1)      {          for (int i = 0; i < 100;i++)              fprintf(fp, "%15d,  ", *n*[i]);          fprintf(fp, "\n");          for (int i = 0; i < 100; i++)              fprintf(fp, "%fms,  ", *time*[i] \* 1000);      }      fclose(fp);  }  int main()  {      int n, flag, num[MAXSIZE];      double time;      \_LARGE\_INTEGER starttime, endtime;      double Freq; *//计时器频率*      \_LARGE\_INTEGER f; *//计时器频率*      int N[MAXSIZE];      double Time[MAXSIZE];      QueryPerformanceFrequency(&f);      Freq = (double)f.QuadPart;      printf("请输入n:");      scanf("%d", &n);      printf("是否递增100次(1是0否):");      scanf("%d", &flag);      if (flag==1)      {          RandomData("D:/HELLOWORLD/2\_1\_before.txt", n\*100);          GetData("D:/HELLOWORLD/2\_1\_before.txt", num);          for (int i = 1; i <= 100; i++)          {              QueryPerformanceCounter(&starttime);              Quick\_sort(num, 0, n - 1);              QueryPerformanceCounter(&endtime);              time = (endtime.QuadPart - starttime.QuadPart) / Freq;              N[i - 1] = n;              Time[i - 1] = time;              n = n + 1000;              printf("此次快速排序耗时%fms\n", time \* 1000);          }          StroeNandTime("D:/HELLOWORLD/2\_1\_n\_time.txt", N, Time, flag);      }      else if(flag==0)      {          RandomData("D:/HELLOWORLD/2\_1\_before.txt", n);          GetData("D:/HELLOWORLD/2\_1\_before.txt", num);          QueryPerformanceCounter(&starttime);          Quick\_sort(num, 0, n - 1);          QueryPerformanceCounter(&endtime);          time = (endtime.QuadPart - starttime.QuadPart) / Freq;          N[0] = n;          Time[0] = time;          StroeNandTime("D:/HELLOWORLD/2\_1\_n\_time.txt", N, Time, flag);          printf("此次快速排序耗时%fms\n", time\*1000);      }      system("pause");      return 0;  } |