**算法分析与设计实验报告**

**第 7 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 周思宇 | 学号 | 201608030201 | | 班级 | 计科1601 |
| 时间 | 6.1 | 地点 | 028 | | | |
| 实验名称 | **Sherwood型线性时间选择算法** | | | | | |
| 实验目的 | 通过上机实验，要求掌握Sherwood型线性时间选择算法的问题描述、算法设计思想、程序设计。  利用Sherwood型随机化思路求解线性时间选择算法，并计算出程序运行所需要的时间。 | | | | | |
| 实验原理 | 利用Sherwood型随机化思路求解线性时间选择算法，并计算出程序运行所需要的时间。 | | | | | |
| 实验步骤 | ① 参考课件、教材、其它资料，将伪代码改成正式程序代码。  ② 用一组小数据，手工验证程序正确性，发现可能的错误并修复。  ③ 自己设计代码，生成小、中、大规模数据，分别存到三个.txt文件。  ④ 对三种规模的数据，检测程序运行时间，观察并记录结果和发现。 | | | | | |
| 关键代码 | ➀快速排序查找第n小  int select(int a[],int l,int r,int k)  {//计算a[l:r]中第k小元素  srand((unsigned)time(NULL));  int rnd=rand();  while(true)  {  if(l>=r) return a[l];  int i=l;  int j=l+(rnd%(r-l+1));  swap(a[i],a[j]);  j=r+1;  int pivot=a[l];  while(true)  {  while(a[++i]<pivot);  while(a[--j]>pivot);  if(i>=j)  break;  swap(a[i],a[j]);  }  if(j-l+1==k) return pivot;  a[l]=a[j];  a[j]=pivot;  if(j-l+1<k){  k=k-j+l-1;  l=j+1;}  else r=j-1;  }  }  ➁洗牌  void shuffle(int a[],int n)  {  for(int i=0; i<n; i++)  {  int rnd=rand();  int j = i+(rnd%(n-i));  swap(a[i],a[j]);  }  } | | | | | |
|  | **下述时间不包含读取数据时间，只包含算法运行时间。**  100万小数据      1000万中数据      1亿大数据 | | | | | |
| 实验心得 | 对于选择问题而言,用拟中位数作为划分基准可以保证在最坏的情况下用线性时间完成选择。如果只简单地用待划分数组的第一个元素作为划分基准，则算法的平均性能较好，而在最坏的情况下需要O(n^2)计算时间。舍伍德选择算法则随机地选择一个数组元素作为划分基准，这样既保证算法的线性时间平均性能，又避免了计算拟中位数的麻烦  也可借助于随机预处理技术，不改变原有的确定性算法，仅对其输入进行随机洗牌，同样可收到舍伍德算法的效果。例如，对于确定性选择算法，可以用下面的洗牌算法shuffle将数组a中元素随机排列，然后用确定性选择算法求解。这样做所收到的效果与舍伍德型算法的效果是一样的。 | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**

//

// main.cpp

// 舍伍德

//

// Created by zsy on 2018/6/1.

// Copyright © 2018年 zsy. All rights reserved.

//

//随机化算法 线性时间选择 输入预处理，洗牌

#include<iostream>

#include<cstdlib>

#include<ctime>

#include <algorithm>

using namespace std;

int a[100000001];

int select(int a[],int l,int r,int k)

{//计算a[l:r]中第k小元素

srand((unsigned)time(NULL));

int rnd=rand();

while(true)

{

if(l>=r) return a[l];

int i=l;

int j=l+(rnd%(r-l+1));

swap(a[i],a[j]);

j=r+1;

int pivot=a[l];

while(true)

{

while(a[++i]<pivot);

while(a[--j]>pivot);

if(i>=j)

break;

swap(a[i],a[j]);

}

if(j-l+1==k) return pivot;

a[l]=a[j];

a[j]=pivot;

if(j-l+1<k){

k=k-j+l-1;

l=j+1;}

else r=j-1;

}

}

/\*int shuffle(int a[],int n)

{

for(int i=0; i<n; i++)

{

int rnd=rand();

int j = i+(rnd%(n-i));

swap(a[i],a[j]);

}

}\*/

int Select(int a[],int n,int k)

{

return select(a,0,n-1,k);

}

int main()

{

int k,n;

//while(1)

{

printf("生成数据个数：");

scanf("%d",&n);

printf("搜索第几小？");

scanf("%d",&k);

FILE \*out1;

out1=fopen("产生随机元素","wb");

srand((unsigned)time(NULL));

for(int i=0;i<n;i++)

fprintf(out1,"%d ",rand());

fclose(out1);

FILE \*in,\*out;

in=fopen("产生随机元素","rb");

out=fopen("输出文件","wb");

fprintf(out,"原数组一共有%d个数\r\n",n);

fprintf(out,"选择这其中的第%d小的数\r\n",k);

//fprintf(out,"原数组为：\r\n");

for(int i=0;i<n;i++)

{

fscanf(in,"%d",&a[i]);

//fprintf(out,"%d ",a[i]);

}

int result;

clock\_t start,end;

start=clock();

//shuffle(a,n);

result=Select(a,n,k);

end=clock();

fprintf(out,"\r\n第%d小的元素为：%d\r\n",k,result);

fprintf(out,"The time is %6.3f",(double)(end-start)/CLK\_TCK);

}

return 0;

}