**算法分析与设计实验报告**

**第 3 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 杨杰 | 学号 | 201908010705 | | 班级 | 计科1907 |
| 时间 | 6.2 | 地点 | 软件大楼 | | | |
| 实验名称 | 用舍伍德法求解线性时间元素选择问题 | | | | | |
| 实验目的 | 通过上机实验，要求掌握用舍伍德法求解线性时间元素选择问题的问题描述、算法设计思想、程序设计。 | | | | | |
| 实验原理 | 在某些特殊情况下，很容易设计出解选择问题的线性时间算法。例如，找n个元素的最小元素和最大元素显然可以在O（n）时间内完成。如果k≤n/logn，通过堆排序算法可以在O(n+ klogn)=O（n）时间内找出第k小元素。当k≥n/logn时，也一样。  但对于一般的选择问题，特别是中位数的选择问题似乎比找最小元素要难。事实上，从渐近阶的意义上看，它们是一样的。一般的选择问题也可以在O（n）时间内得到解决。一个思路是模仿快速排序算法设计。 | | | | | |
| 实验步骤 | （1） 采用划分的思想，每次在序列[l:r]中随机选择一个轴，将小于轴的元素归为左边，否则归为右边。  （2） 判断当前轴的位置是否为k，如果是那就说明找到了一个解。否则就要判断k的轴位置大小。  （3） 如果轴小于k，说明第k小的数据存在于右边的序列，此时k要迭代为k=k-j+l-1  （4） 否则k存在于左序列。  （5） 更行左右区间端点，直到找到这个解。  以上的思想利用舍伍德的思想随即划分并且选择左右区间的一部分，因此可以在线性时间内求解。 | | | | | |
| 关键代码 | */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*算法过程\*/*  template <class Type>  Type select(Type a[], int l, int r, int k)  {      while (true)      {          if (l >= r)              return a[l];          int i = l, j = l + Random(r - l + 1);*//随机划分，j为[i:r]之间的一个任意位置*          swap(a[i], a[j]);*//交换a[i]和a[j]，将首元素放在划分点a[j]*          j = r + 1;*//将j换成右端点*          Type pivot = a[l];*//pivot选定为轴*          while (true)*//小于pivot的元素归为左边，否则归为右边*          {              while (a[++i] < pivot && i <= r)                  ;              while (a[--j] > pivot && j >= 0)                  ;              if (i >= j)                  break;              swap(a[i], a[j]);          }          if (j - l + 1 == k)              return pivot;*//如果右端点与左端点的距离之差等于k，说明找到一个解*          a[l] = a[j];*//将轴换回中间*          a[j] = pivot;*//*          if (j - l + 1 < k)*//右序列*          {              k = k - j + l - 1;*//k值要减小(j-l+1),前面有比k小的j-l+1个元素*              l = j + 1;          }          else              r = j - 1;*//左序列*      }  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*算法结束\*/*  template <class Type>  Type Select(Type a[], int n, int k)  {      if (k < 1 || k > n)          return INF;*//不存在*      return select<Type>(a, 0, n - 1, k);  } | | | | | |
| 算法复杂度分析 | 运用算法分析的知识，分析并阐述求解本问题算法的各个模块的时空复杂度。   1. 平均时间复杂度：T(n)=O(n) 最坏情况下的时间复杂度为T(n)=O(n^2)   由于算法 Select使用随机数产生器随机地产生l和r之间的一个随机整数，因此产生的划分基准是随机的。可以证明，当用算法 Select对含有n个元素的数组进行划分时，划分出的低区子数组中含有一个元素的概率为2/n，含有i个元素的概率为1/（i=2,3，…，n-1）。设T（n）是算法 Select作用于一个含有n个元素的输入数组上所需的期望时间的上界，T（n）是单调递增的。在最坏情况下，第k小元素总是被划分在较大的子数组中。由此可以得到关于T（n）的递归式：  在这里插入图片描述  在上面的推导中，从第1行到第2行是因为max（1，n-1）=n-1，而  在这里插入图片描述  且n是奇数时，T（n2），T（m2+1），…，7n-1）在和式中均出现2次；n是偶数时，T(n/2+1),T（n/2+2），…，T（n-1）均出现2次，T(n/2)只出现1次。因此，第2行中的和式是第1行中和式的上界。从第2行到第3行是因为在最坏情况下T(n-1）=O（n^2），故可将T（n-1）包含在O（n）项中。解上面的递归式可得T（n）=O（n）;最坏情况下每次划分都在起始位置或结束位置，所以算法计数频次为C1+C2……C\*n所以此时的时间复杂度为O(n^2)   1. 空间复杂度：S(n)=O(n^2)   算法只需要一个数组来存储这个列表，记为C\*n，因此算法空间复杂度为S(n)=O(n). | | | | | |
| 测试结果  （含运行时间） | **小规模数据**      **中规模数据**      **大规模数据**      从三种数据规模的测试用例的运行时间也可以看出算法的时间复杂度为O(n). | | | | | |
| 实验心得 | 经过这次实验，我对于用舍伍德法求解线性时间元素选择问题的相关代码已基本熟悉，算法知识得到了复习与巩固。在写代码与调试的过程中，在解决问题过程中，丰富了个人编程的经历和经验，提高了个人解决问题的能力。 | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**

*//舍伍德法求解线性时间元素选择*

#include <iostream>

#include <windows.h>

#include <fstream>

#include <time.h>

using namespace std;

#define INF 0x7fffffff*//无穷大用于判断是否存在*

#define Random(n) (int)rand() % n

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*算法过程\*/*

template <class Type>

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

{

    while (true)

    {

        if (l >= r)

            return a[l];

        int i = l, j = l + Random(r - l + 1);*//随机划分，j为[i:r]之间的一个任意位置*

        swap(a[i], a[j]);*//交换a[i]和a[j]，将首元素放在划分点a[j]*

        j = r + 1;*//将j换成右端点*

        Type pivot = a[l];*//pivot选定为轴*

        while (true)*//小于pivot的元素归为左边，否则归为右边*

        {

            while (a[++i] < pivot && i <= r)

                ;

            while (a[--j] > pivot && j >= 0)

                ;

            if (i >= j)

                break;

            swap(a[i], a[j]);

        }

        if (j - l + 1 == k)

            return pivot;*//如果右端点与左端点的距离之差等于k，说明找到一个解*

        a[l] = a[j];*//将轴换回中间*

        a[j] = pivot;*//*

        if (j - l + 1 < k)*//右序列*

        {

            k = k - j + l - 1;*//k值要减小(j-l+1),前面有比k小的j-l+1个元素*

            l = j + 1;

        }

        else

            r = j - 1;*//左序列*

    }

}

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*算法结束\*/*

template <class Type>

Type Select(Type a[], int n, int k)

{

    if (k < 1 || k > n)

        return INF;*//不存在*

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

}

template <class Type>

double testTime(Type Select(Type a[], int n, int k), Type a[], int n, int k, Type &numberK)

{

    LARGE\_INTEGER frequency;*//时间对象*

    double dff, begin\_, \_end;*//时钟频率，起始时间，结束时间，时间差*

    QueryPerformanceFrequency(&frequency);*//获得时钟频率*

    dff = (double)frequency.QuadPart;*//取得频率*

    QueryPerformanceCounter(&frequency);

    begin\_ = frequency.QuadPart;*//获得初始值*

    numberK = Select(a, n, k);

    QueryPerformanceCounter(&frequency);

    \_end = frequency.QuadPart;*//获得终止值*

    return (\_end - begin\_) / dff;*//差值除以频率得到时间*

}

int main()

{

    ifstream infile("舍伍德法求解线性时间元素选择\_in.txt", ios::in);*//读入*

    ofstream outfile("舍伍德法求解线性时间元素选择\_out.txt", ios::out);*//写入*

    int n, k, numberK;

    infile >> n >> k;

    int a[n + 1] = {0};

    for (int i = 0; i < n; i++)

        infile >> a[i];*//读入数据*

*/\*测试函数\*/*

    double time = testTime<int>(Select, a, n, k, numberK);

*/\*输出结果\*/*

    outfile << "数据规模：  n=" << n << "  k=" << k << endl;

    if (numberK != INF)

        outfile << "第k个元素为：" << numberK << endl;

    else

        outfile << "元素在序列中不存在！" << endl;

    outfile << "时间花费为：spendTime=" << time \* 1000 << "ms";

    outfile << endl

            << "-------------------------------------------------" << endl;

    infile.close();

    outfile.close();

    cout << "excute sucefully!" << endl;

    return 0;

}

*//舍伍德法求解线性时间元素选择数据生成器*

#include <iostream>

#include <fstream>

#include <time.h>

#define random(a, b) (rand() % (b - a + 1) + a)

using namespace std;

int main()

{

    ofstream outfile("舍伍德法求解线性时间元素选择\_in.txt", ios::out);

    int t,k;

    cin >> t;

    srand((int)time(NULL));*//time seed*

    k=random(1,t);

    outfile<<t<<" "<<k<<endl;

    for(int i=1;i<=t;i++)

    {

        outfile<<random(1,10000)<<" ";

        if(i%20==0)

            outfile<<endl;

    }

    outfile.close();

    return 0;

}