

磁盘文件最优化存储问题

一. 问题分析:

由于代价函数时间期望: $t = \sum_{1 \leq i \leq j \leq n} p_i p_j d(i, j)$, 为了使时间期望更低, 可以使出现概率

最高的文件放在最中间, 这样, 其它文件到概率最大的文件的距离和最低, 使得其满足最优化, 尽量使得概率较高的文件放在中心的磁道附近。

二. 算法设计与分析:

采用的贪心策略: 将概率较高的文件放在磁道附近:

算法设计:

(1) 首先对所有文件出现的概率进行排序, 使得排序后满足:

$$p_1 \geq p_2 \geq \dots \geq p_n$$

(2) 把 f_1 放在中间磁道, 将 f_2 和 f_3 分居于 f_1 两侧, f_4 位于 f_2 的左侧...

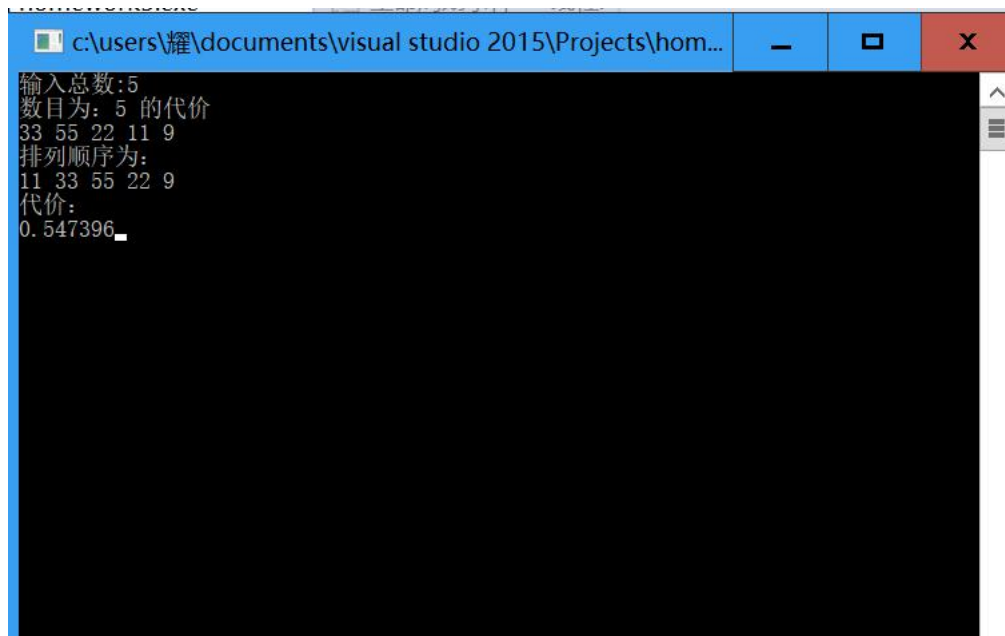
(3) 计算总的代价

算法复杂度分析:

最主要的算法复杂度在于对文件概率大小的排序。因此算法复杂度为 $O(n \lg n)$

三. 程序实现:

通过控制台输入, 首先输入文件数目 m , 接着输入 m 个文件的读取次数, 然后在控制台上显示结果, 其运行结果如图下所示, 输入例题中的示例可以得到:



```
输入总数:5
数目为: 5 的代价
33 55 22 11 9
排列顺序为:
11 33 55 22 9
代价:
0.547396_
```

四. 代码:

```
// homework5.cpp : 定义控制台应用程序的入口点。
//
```

```

#include "stdafx.h"
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>
using namespace std;
int main()
{
    cout << "输入总数:";
    int num; //文件数目多少
    vector<int> vl; //存储文件访问次数
    vector<int> rl; //存储结果
    int wait;
    int total = 0;
    int direction = 1;
    float cost = 0;
    //输入数据
    cin >> num;
    cout << "数目为: " << num << " 的代价"<<endl;
    int middle = int(num / 2);
    for (int i = 0; i < num; i++)
    {
        int temp = 0;
        cin >> temp;
        rl.push_back(0);
        vl.push_back(temp);
    }
    //对文件概率进行排序
    sort(vl.begin(), vl.end());
    for (int j = 0; j < num; j++)
    {
        total = total + vl[j];
    }
    //对磁道进行安排
    for (int j = 0; j < num; j++)
    {
        int index = middle+abs(int((j+1)/2))*direction;
        rl[index] = vl[num-1-j];
        direction = -direction;
    }
    cout << "排列顺序为: " << endl;
    //计算代价
    for (int i = 0; i < num; i++)

```

```
{  
    for (int j = 0; j < num; j++)  
    {  
        cost = float(r1[i])*float(r1[j])*abs(i - j)+cost;  
    }  
  
    cout << r1[i]<<" ";  
}  
cost = cost / float(total*total*2);  
//输出结果  
cout <<endl<<"代价: " <<endl<< cost;  
cin >> wait;  
return 0;  
}
```