实验预警

本次实验是英特尔酷睿处理器 i7-8750H(CPU天梯图最上方的存在)。

操作系统是 window10 1803家庭版

测试数据为随机产生的随机数， 测试时间度为 顺序查找 和 二分查找

因为c++的 clock（）函数的精度为毫秒级，以下时间数据的复杂度都是 ms

数据规模	顺序成功查找	顺序失败查找的时间复杂度	二分成功查找的时间复杂度	二分失败查找的时间复杂度
100	0	0	0	0
1000	0	0	0	0
10000	0	1	0	0
100000	1	1	0	1
1000000	1	1	1	1

【实验分析】

本次实验中，不同数据规模的查找，返回的时间值都为0/1ms，

其中 二分查找的数组排序是在查找时间以外的，因为数据的生成是随机的，随机生成的数据是无序的，二分查找要求有序

数据生成函数

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <sstream>

using namespace std;

string trans(int);

int main() {
    ofstream output;
    int x;
    cin >> x;
    srand((unsigned)time(NULL));
    output.open("data.txt", ios::trunc);
```

```
    for(int i = 0; i < x; ++i) {
        output << rand() << ' ';
    }

    return 0;
}
```

数据查找

```
#include <iostream>
#include <ctime>
#include <fstream>
#include <sstream>
#include <string.h>
#include <algorithm>
using namespace std;

int shunxufind(int a[],int n,int target)
{
    for(int i=0;i<n;i++)
    {
        if(a[i]==target) return i;
    }
    return -1;
}
int erfenfind(int a[], int n , int target)
{
    int low = 0 ,high = n , middle;
    while(low < high)
    {
       middle = (low + high)/2;
       if(target == a[middle])
           return middle;
       else if(target > a[middle])
           low = middle +1;
       else if(target < a[middle])
           high = middle;
    }
    return -1;
}
bool trans(char a[],int x)
{
    stringstream ss;
    string str;
    ss << x;
    ss >> str;
    str=str+".txt";
    strcpy(a,str.c_str());
}
int main() {
    int n,target;
    cout << "要查找的文件大小为（100,1000,10000,100000,1000000）"<<endl;
    cin >> n;
    cout << "要查找的数字为"<<endl;
    cin >> target;
    int *tempA = new int[n];
```

```
    int *tempB = new int[n];
    char path[n];
    trans(path,n);
    ifstream input(path);
    if (!input.is_open())
    {
        cout << "can not open this file" << endl;
        return 0;
    }
    for (int i = 0; i < n; ++i) {
        input >> tempA[i];
        tempB[i] = tempA[i];
    }
    sort(tempB,tempB+n);
    input.close();
    clock_t start_time1 = clock();
    if(shunxufind(tempA, n,target)==-1) cout << "顺序查找未查找到相应数据    "<<endl;
    else cout << "顺序查找查找到相应数据    "<<target<<endl;
    clock_t end_time1 = clock();
    cout << "顺序查找运行时间："
         << static_cast<double>(end_time1 - start_time1) / CLOCKS_PER_SEC * 1000
         << "ms" << endl;
    start_time1 = clock();
    if(erfenfind(tempB, n,target)==-1)  cout << "二分查找未查找到相应数据    "<<endl;
    else cout << "二分查找查找到相应数据    " <<target<<endl;
    end_time1 = clock();
    cout << "二分查找运行时间："
         << static_cast<double>(end_time1 - start_time1) / CLOCKS_PER_SEC * 1000
         << "ms" << endl;
    return 0;
}
```

实验查找，顺序查找函数为例题3-1所示，二分查找函数为简单的二分查找.

Tips

大概可能是因为处理器原因，没有得到对比度比较高的结果。

后来尝试用更大的数据规模，不过 的确可以写入文件，但是在查找的时候，再过大的数据会崩内存，因此想法没有得到有效结果。

cpu无法限制功率，降频，数据规模也无法修改，故问题就在这里了。