中山大学计算机学院本科生实验报告

（2021 学年第 1 学期）

课程名称：Data structures and algorithms　　任课教师：张子臻

年级	20 级	专业（方向）	软件工程
学号	20337270	姓名	钟海财
电话	13397996670	Email	2940599563@qq.com
开始日期	2021/9/30	完成日期	2021/10/6

1. 实验题目

1) Binary Search

2) 方程求解

3) 最大值最小化

2. 实验目的

通过使用二分查找法，解决以下问题：

1.找一串有序数字中是否存在目标数，若存在则返回该数字的位置，反之返回 -1 ；

2.已知方程的表达式，且函数是单调递增函数，告诉 y 的值，求出对应的 x 值，且 y 与 f(x)的误差小于 1e-6 ；

3．将一串有序数字分为若干个划分，使这些划分数字和的最大值最小化，求出该最小值。

3.程序设计

1）Binary Search

设计思路

定义 4 个 int 型变量，bottom，top，mid，index：bottom 最开始为 0，top 最开始为 length-1，mid=(top+index)/2, index=-1。

由于是找到目标数字在该串数字中最后出现的位置，通过二分查找，则最先在 bottom 找到 target，再将 bottom 逐渐往后(令 bottom=mid+1)找到 target 至结束，且每次找到 target 时都令 index=mid。则每次循环中 bottom、top 分别与 mid 的关系为：

```
mid=(top+index)/2 ;
if( s[mid] > target )    top = mid-1 ;
  else{
          bottom = mid+1 ;
           if(s[mid] == target) index=mid ;
    }
```

最终返回 index 的值。

代码：

```
//找 target 最后出现的位置
#include "binSearch.h"
int binSearch(const int s[], const int size, const int target)
{
    int bottom = 0 ;
    int top = size-1 ;
    int index = -1 ;
    while(bottom <= top ){
        int mid = (bottom+top)/2 ;
        if( s[mid] > target )  top = mid-1 ;
        else{
            bottom = mid+1 ;//bottom 最先找到 target,然后从前往后找到 target
至结束
            if(s[mid] == target) index=mid ;
        }
    }
    return index;
}
```

2）方程求解

设计思路

由于 y=f(x)=e^x+lnx-1 是关于 x 的单调递增函数,则 x1<x2,有 f(x1)<f(x2) ,则对于给出的 y(0 < y < 1e10)值要求对应的 x 值，可通过二分查找，开始定义 3 个 long double 类型的变量: bottom=0，top=24（f(24)>1e10），mid=(bottom+top)/2.

满足条件精度不足(abs(exp(mid)+log(mid)-1-y) >= 1e-6)时循环，且每次循环时:

if(exp(mid)+log(mid)-1 < y) bottom = mid ;

else top = mid ;

mid = (bottom+top)/2 ;

代码：

```
#include<cmath>
#include "solve.h"
using namespace std ;

long double solve(long double y){
    long double bottom = 0 ;
    long double top = 24 ;
    long double mid = top/2 ;
    while( abs(exp(mid)+log(mid)-1-y) >= 1e-6 ){
        if( exp(mid)+log(mid)-1 < y) bottom = mid ;
        else top = mid ;
        mid = (bottom+top)/2 ;
    }
    return mid;
}
```

3）最大值最小化

设计思路：

循环设计：通过 int num; while(cin>>num){}完成每次读入一串数字的循环，并将有序数组读入数组 data[]中。

确定上下界：设置两个 int 变量 max 和 min，表示子序列和最大值的上界和下界，初始时 min= data[i]的最大值，max=data[i]的和-划分数+1。

循环查找：通过二分查找，当 min<=max 时循环,令 mid=(max+min)/2,设置一个 ok 函数(判断 mid 是否可以做子序列和的最大值的上界，mid 可以做子序列和最大值的上界时返回 ture，否则返回 false)。

if(ok(data,num,groups,mid)){ //mid 可以做子序列和的最大值时

max=mid-1;

min_of_max = mid ;

}

else　min = mid+1 ; //mid 不能做子序列和的最大值时

通过 min 与 max 的相互靠近至最终相等，我们便得到了最小的“子序列和的最大值”,完成了最大值最小化。

ok 函数：判断 mid 是否可以做子序列和的最大值的上界

```
bool ok(int *data,int num,int groups,int mid){
    int sum = 0 ;
    int groups0 = 1; //记录按子序列和的最大值为 mid 时的划分数
    for(int i = 0 ; i < num ; ++i){
        sum+=data[i] ;
        if(sum > mid){ //如果 sum+data[i]>mid 时就新增 1 组
            sum=data[i] ;
            ++groups0 ;
        }
    }
    if(groups0 <= groups) return true ;//组数和小于等于 groups，说明可以分组
    else return false ;
}
```

代码：

```
//最大值最小化
#include<iostream>
using namespace std;
bool ok(int *data,int num,int groups,int mid);//判断 mid 是否可以做子序列和的最大值

int main(){
    int num;//数字个数
    while(cin >> num ){
    int groups; //划分数
    cin >> groups ;
    int data[num] ;
    int sum= 0 ;
    int min = 0 ;//子序列和的最大值的下界
    for(int i = 0 ; i < num ; ++i){
        cin>>data[i] ;
        sum+=data[i] ;
```

```
            if(data[i]>min) min = data[i] ;//子序列和的最大值的下界= max_of_data[i]
        }
        int max=sum-groups+1 ;//子序列和的最大值的上界,其余 n-1 组都只有 1 个 1 时
        int min_of_max ;//最小的子序列和的最大值
        while( min<=max){
            int mid = (min+max)/2 ;
            if(ok(data,num,groups,mid)){ //mid 可以做子序列和的最大值时
                max=mid-1;
                min_of_max = mid ;
                }
            else  min = mid+1 ; //mid 不能做子序列和的最大值时
        }

        cout << min_of_max <<endl ;
        }
    }

    bool ok(int *data,int num,int groups,int mid){
        int sum = 0 ;
        int groups0 = 1; //记录按子序列和的最大值为 mid 时的划分数
        for(int i = 0 ; i < num ; ++i){
            sum+=data[i] ;
            if(sum > mid){ //如果 sum+data[i]>mid 时就组数+1，并新增 1 组
                sum=data[i] ;
                ++groups0 ;
            }
        }
        if(groups0 <= groups) return true ;//组数和小于等于 groups，说明可以分组
        else return false ;
}
```

4.程序运行与测试

1）Binary Search

序号	输入	输出	是否通过
1	**{0,1,1,3,3,3,6,6} ， 3**	**5**	是
2	**{0,1,3,4,5,6,8,12,12,12,14} ， 12**	**9**	是

2）方程求解

序号	输入	输出	是否通过
1	7362.123	8.90394	是
2	123	4.80754	是
3	2844682	14.861	是

3）最大值最小化

序号	输入	输出	是否通过
1	6 3 1 2 3 2 5 4	7	是
2	12 4 1 2 4 5 6 7 8 8 9 12 14 17	26	是
3	15 5 1 2 4 5 6 7 8 8 9 12 14 17 21 27 30	41	是

把三个测试输入样例同时输入也行。

5.实验总结与心得

本次实验让我熟悉了二分查找法的使用，二分查找法适用于在有序数据中查找目标 target，通过与 mid 的比较，将 bottom 和 top 逐渐靠拢，最终 bottom==top，在此过程中如果 target 存在一定能找到，时间复杂度为 O(lgN)。

在含有多个 target 的比较过程中，我们将 mid 的数据 = target 时情况归类在令 bottom->mid 或 top->mid 会使最终找到的 target 的出现的位置不同：

归类在 bottom->mid 中：最终找到的 target 最后出现的位置。

归类在 top->mid 中：最终找到的 target 最先出现的位置

附录、提交文件清单