《数据结构与算法》实验报告

实验题目	二分查找、堆实践		
实验时间	2023/12/1 9:00-12:00	实验地点	DS3402
实验成绩		实验性质	☐验证性 √设计性 ☐综合性

教师评价：

☐算法/实验过程正确； ☐源程序/实验内容提交 ☐程序结构/实验步骤合理；

☐实验结果正确； ☐语法、语义正确； ☐报告规范；

其他：

评价教师签名：

一、实验目的

1. 掌握堆的基本概念，堆排序以及二分查找的基本原理
2. 训练使用堆与二分查找，通过编程解决不同难度问题的实践能力

二、实验项目内容

注：每道题按下面的格式分别描述

实验题目 1：正整数质因数分解

题目内容：

如果一个整数 a 除以整数 b 的商正好是整数，即 b 能够整除 a，我们就称 b 是 a 的一个“因数”。如果因数 b 恰好是质数，我们就称其为 a 的“质因数”。

现在给你一个正整数 a，请你按照从小到大的顺序输出它所有的质因数。

输入格式:

一行，一个正整数 a(2≤a≤1e7)

输出格式:

一行，用空格分开的若干个正整数，即按照从小到大顺序排列的 a 的所有

质因数（相同数字不重复输出）

代码：

```
#include<bits/stdc++.h>
using namespace std;



int main() {
    int n;
    cin>>n;
    bool* prime=new bool [n];
    for(int k=2;k<=n;k++){
        prime[k]=true;
    }
    prime[1]=false;
    for(int k=2;k<=n;k++){
        if(prime[k]){
            int m=2*k;
            while(m<=n){
                prime[m]=false;
                m=m+k;
            }
        }
    }
    bool flag=1;
    for(int i=2;i<=n;i++){
        if(n%i==0&&prime[i]){
            if(flag){cout<<i;flag=0;}
            else cout<<" "<<i;
            n=n/i;
        }
```

```
    }
}
```

实验题目 2：二分查找

题目内容：对于输入的 n 个整数，先进行升序排序，然后进行二分查找。

输入格式:

测试数据有多组，处理到文件尾。每组测试数据第一行输入一个整数 n(1≤n≤100)，第二行输入 n 个各不相同的待排序的整数，第三行是查询次数 m（1≤m≤100），第四行输入 m 个待查找的整数。

输出格式:

对于每组测试，分 2 行输出，第一行是排序后的升序的结果，每两个数据之间留一个空格；第二行是查找的结果，若找到则输出排序后元素的位置（从 1 开始，每两个数据之间留一个空格），否则输出 0。

代码：

```
#include<bits/stdc++.h>
using namespace std;

int* a;
int n;
int search(int target){
    int low  =  - 1;
    int high = n;
    while(high - low > 1){
        int mid= (low + high) / 2;
        if (a[mid]==target)  return mid+1;
        else if(a[mid]<target) low = mid;
        else high=mid;
    }
```

```
        return 0;
    }

    int main() {
        while(cin>>n){
        a=new int [n];
        for(int i=0;i<n;i++){
            cin>>a[i];
        }
        sort(a,a+n);
        cout<<a[0];
        for(int i=1;i<n;i++){
            cout<<" "<<a[i];
        }
        int m;
        cin>>m;
        int target;
        cin>>target;
        cout<<"\n"<<search(target);
        for(int i=1;i<m;i++){
            cin>>target;
            cout<<" "<<search(target);
        }
            cout<<"\n";
        }
    }
```

实验题目 3：寻找大富翁

题目内容：胡润研究院的调查显示，截至 2017 年底，中国个人资产超过 1 亿元的高净值人群达 15 万人。假设给出 N 个人的个人资产值，请快速找出

资产排前 M 位的大富翁。

输入格式:

输入首先给出两个正整数 N（≤1e6）和 M（≤10），其中 N 为总人数，M 为需要找出的大富翁数；接下来一行给出 N 个人的个人资产值，以百万元为单位，为不超过长整型范围的整数。数字间以空格分隔。

输出格式:

在一行内按非递增顺序输出资产排前 M 位的大富翁的个人资产值。数字间以空格分隔，但结尾不得有多余空格。

代码：

```
#include<bits/stdc++.h>
using namespace std;

int last;
void swftup(long long* a, long long b) {
	long long temp = a[b];
	while (b > 1 && temp < a[b / 2]) {
		a[b] = a[b / 2];
		b = b / 2;
	}
	a[b] = temp;
}
void swftdown(long long* a, long long b) {
	long long temp = a[b];
	while (1) {
		long long child = 2 * b;
		if (child < last && a[child + 1] < a[child]) {
			child++;
		}
```

```
            else if (child > last) break;
            if (a[child] < temp) {
                a[b] = a[child];
                b = child;
            }
            else break;
        }
        a[b] = temp;
}
void insert(long long* a, long long b) {
        last++;
        a[last] = b;
        swftup(a, last);
}
void extract(long long* a) {
        a[1] = a[last];
        last--;
        swftdown(a, 1);
}

int main() {
        int n,m;
        cin>>n>>m;
        if(m>n) m=n;
        long long heap[m+1];
        heap[0]=0;
        last=0;
        long long temp;
        for(int i=0;i<m;i++){
            cin>>temp;
            insert(heap,temp);
```

```
        }
        for(int i=m;i<n;i++){
            cin>>temp;
            if(temp>heap[1]){
                extract(heap);
                insert(heap,temp);
            }
        }
        sort(heap,heap+last+1);
        cout<<heap[last];
        for(int i=1;i<m;i++){
            cout<<" "<<heap[last-i];
        }
    }
```

实验题目 4：愤怒的牛

题目内容：农夫约翰建造了一座有 n 间牛舍的小屋，牛舍排在一条直线上，第 i 间牛舍在 xi 的位置，但是约翰的 m 头牛对小屋很不满意，因此经常互相攻击。约翰为了防止牛之间互相伤害，因此决定把每头牛都放在离其它牛尽可能远的牛舍。也就是要最大化最近的两头牛之间的距离。

牛们并不喜欢这种布局，而且几头牛放在一个隔间里，它们就要发生争斗。为了不让牛互相伤害。John 决定自己给牛分配隔间，使任意两头牛之间的最小距离尽可能的大，那么，这个最大的最小距离是多少呢？

输入格式:

第一行用空格分隔的两个整数 n 和 m，n，m<=105；

第二行为 n 个用空格隔开的整数，表示位置 xi 。xi <MAXINT

输出格式:

输出仅一个整数，表示最大的最小距离值。

代码：

```
#include<bits/stdc++.h>
using namespace std;


int* wu;
int m,n;
bool canlive(int distance){
	int i=0;
	int last=0;
	int cow=1;
	while(i<n && cow<m){
		if((wu[i]-wu[last])>=distance){
			last=i;
			cow++;
		}
		i++;
	}
	if(cow!=m)return false;
	else return true;
}
int search(int left,int right){
	int low    = left - 1;
	int high = right + 1;
	while(high - low > 1){
		int mid= (low + high) / 2;
		if(canlive(mid)) low = mid;
		else high=mid;
	}
	return low;
}
```

```
int main() {
    cin >> n >> m;
    wu = new int [n];
    for(int i=0;i<n;i++){
        cin>>wu[i];
    }
    sort(wu,wu+n);
    cout<<search(0,n);
}
```

实验题目 5：宠物收养场

题目内容：凡凡开了一间宠物收养场。收养场提供两种服务：收养被主人遗弃的宠物和让新的主人领养这些宠物。每个领养者都希望领养到自己满意的宠物，凡凡根据领养者的要求通过他自己发明的一个特殊的公式，得出该领养者希望领养的宠物的特点值 a（a 是一个正整数，a<2^31），而他也给每个处在收养场的宠物一个特点值。这样他就能够很方便的处理整个领养宠物的过程了，宠物收养场总是会有两种情况发生：被遗弃的宠物过多或者是想要收养宠物的人太多，而宠物太少。

被遗弃的宠物过多时，假若到来一个领养者，这个领养者希望领养的宠物的特点值为 a，那么它将会领养一只目前未被领养的宠物中特点值最接近 a 的一只宠物。（任何两只宠物的特点值都不可能是相同的，任何两个领养者的希望领养宠物的特点值也不可能是一样的）如果有两只满足要求的宠物，即存在两只宠物他们的特点值分别为 a-b 和 a+b，那么领养者将会领养特点值为 a-b 的那只宠物。

收养宠物的人过多，假若到来一只被收养的宠物，那么哪个领养者能够领养它呢？能够领养它的领养者，是那个希望被领养宠物的特点值最接近该宠物特点值的领养者，如果该宠物的特点值为 a，存在两个领养者他们希望领养宠物的特点值分别为 a-b 和 a+b，那么特点值为 a-b 的那个领养者将成功领养该宠物。

一个领养者领养了一个特点值为 a 的宠物，而它本身希望领养的宠物的特点值为 b，那么这个领养者的不满意程度为 abs(a-b)。

你得到了一年当中，领养者和被收养宠物到来收养所的情况，请你计算所有收养了宠物的领养者的不满意程度的总和。这一年初始时，收养所里面既没有宠物，也没有领养者。

输入格式:

第一行为一个正整数 n，n<=80000，表示一年当中来到收养场的宠物和领养者的总数。接下来的 n 行，按到来时间的先后顺序描述了一年当中来到收养场的宠物和领养者的情况。每行有两个正整数 a, b，其中 a=0 表示宠物，a=1 表示领养者，b 表示宠物的特点值或是领养者希望领养宠物的特点值。（同一时间呆在收养所中的，要么全是宠物等领养者，要么全是领养者要宠物，这些宠物和领养者的个数不会超过 10000 个）

输出格式:

仅有一个正整数，表示一年当中所有收养了宠物的领养者的不满意程度的总和 mod 1000000 以后的结果。

代码：

```
#include<bits/stdc++.h>
using namespace std;

int search(int target,vector<int>&a,vector<int>&b){
    int n = a.size();
    if(n==0) return 0;
    b.pop_back();
    if(n==1){
        int dis=abs(a[0]-target);
        auto iter = a.erase(a.begin());
        return dis;
    }
    sort(a.begin(),a.begin()+n);
    int low    = 0;
    int high = n-1;
    while(high - low > 1){
        int mid= (low + high) / 2;
        if(a[mid]==target){
            //cout<<"fine."<<endl;
            auto iter = a.erase(a.begin() + mid);
```

```
                return 0;
            }
            else if(a[mid]<target) low = mid;
            else high=mid;
        }
        /*if(low==-1){
            int dis=abs(a[0]-target);
            auto iter = a.erase(a.begin());
            return dis;
        }
        if(high==n){
            int dis=abs(a[low]-target);
            auto iter = a.erase(a.begin()+low);
            return dis;
        }*/
        if(target-a[low]>a[high]-target){
            int dis = abs(a[high]-target);
            //cout<<"higher than target"<<dis<<endl;
            auto iter = a.erase(a.begin() + high);
            return dis;
        }
        else{
            int dis = abs(target-a[low]);
            //cout<<"lower than target "<<dis<<endl;
            auto iter = a.erase(a.begin() + low);
            return dis;
        }
    }
```

```
int main() {
    int n;
    cin >> n ;
    bool people;//来的人是不是人
    vector<int> pet;     //记录特征值
    vector<int> master;//记录特征值
    int unsatasfy=0;          //最后输出：不满意，取模
    int track;              //输入特征值
    for(int i=0;i<n;i++){
        cin>>people;
        cin>>track;
        if(people)master.push_back(track);
        else pet.push_back(track);
        if(people)unsatasfy+=search(track,pet,master);
        else unsatasfy+=search(track,master,pet);
        unsatasfy=unsatasfy%1000000;
    }
    cout<<unsatasfy;
}
```