> 今天的算法分享是**二分**，内容如下：

- 使用的笔记软件为**vs code**。
- 语法为**markdown**.

[TOC]

二分算法介绍

二分算法（Binary Search）是一种常见的搜索算法，它通过 repeatedly 将搜索范围缩小一半，直到找到目标元素或确定其不存在。

核心思路

二分算法的核心在于利用数据的有序性或单调性，不断将可能的范围缩小一半，从而高效地定位满足条件的解。典型应用包括在区间内查找特定数值、确定最大/最小满足条件的数值等。

常用条件

- 数据必须是**有序**的（升序或降序都可以，但代码需根据顺序调整）。
- 二分查找的本质是不断地缩小查找区间，直到找到目标元素或确定其不存在。

基本流程

1. 设定初始区间：根据题意，设定左右边界 *l* 和 r。
2. 确定中间位置：计算中间位置 m = (l + r) / 2。
3. 判断中间位置的值：
4. 如果 m 的值等于目标值，返回 m。
5. 如果 m 的值小于目标值，更新左边界 l = m + 1。
6. 如果 m 的值大于目标值，更新右边界 r = m - 1。
7. 重复步骤 2 到步骤 6，直到 l > r。
8. 返回 -1。

代码示例

无注释版本

```
#include <iostream>
using namespace std;

int main() {
    int a[] = {1, 3, 5, 7, 9, 11, 13, 15};
    int n = sizeof(a) / sizeof(a[0]);
    int x = 7;
    int l = 0, r = n - 1, m, found = -1;

    while (l <= r) {
        m = (l + r) / 2;
```

```
        if (a[m] == x) {
            found = m;
            break;
        } else if (a[m] < x) {
            l = m + 1;
        } else {
            r = m - 1;
        }
    }

    if (found != -1) {
        cout << "Found at index: " << found << endl;
    } else {
        cout << "Not found" << endl;
    }

    return 0;
}
```

带注释版

```
#include <iostream>
using namespace std;

int main() {
    int a[] = {1, 3, 5, 7, 9, 11, 13, 15}; // 已排序的数组
    int n = sizeof(a) / sizeof(a[0]); // 数组大小
    int x = 7; // 要查找的目标值
    int l = 0, r = n - 1; // 左边界和右边界初始化
    int m, found = -1; // m为中点索引，found用于标记找到的索引

    while (l <= r) { // 当左边界不超过右边界时继续
        m = (l + r) / 2; // 计算中点
        if (a[m] == x) { // 如果中点值等于目标值
            found = m; // 标记找到的索引
            break; // 退出循环
        } else if (a[m] < x) { // 如果中点值小于目标值
            l = m + 1; // 将左边界右移至中点右侧
        } else { // 如果中点值大于目标值
            r = m - 1; // 将右边界左移至中点左侧
        }
    }

    if (found != -1) { // 判断是否找到
        cout << "Found at index: " << found << endl; // 输出找到的索引
    } else {
        cout << "Not found" << endl; // 输出未找到
    }

    return 0;
```

```
}
```

适用场景

> - 在有序数组中查找特定数值。
> - 确定最大/最小满足条件的数值。
> - 解决二分查找问题。
> - 解决动态规划问题。
> - 解决贪心算法问题。

题目练习

1. 洛谷 -p2249.有序序列中查找元素

【深基13.例1】查找

题目描述

输入 n 个不超过 10^9 的单调不减的（就是后面的数字不小于前面的数字）非负整数 $a_1,a_2,\dots,a_{n}$，然后进行 m 次询问。对于每次询问，给出一个整数 q，要求输出这个数字在序列中第一次出现的编号，如果没有找到的话输出 -1 。

输入格式

第一行 2 个整数 n 和 m，表示数字个数和询问次数。

第二行 n 个整数，表示这些待查询的数字。

第三行 m 个整数，表示询问这些数字的编号，从 1 开始编号。

输出格式

输出一行，m 个整数，以空格隔开，表示答案。

样例 #1

样例输入 #1

```
11 3
1 3 3 3 5 7 9 11 13 15 15
1 3 6
```

样例输出 #1

```
1 2 -1
```

提示

数据保证，$1 \leq n \leq 10^6$，$0 \leq a_i,q \leq 10^9$，$1 \leq m \leq 10^5$

本题输入输出量较大，请使用较快的 IO 方式。

参考题解

```
#include<cstdio>
using namespace std;

int n,m,q,a[1000005];

int find(int x) //二分查找
{
    int l=1,r=n;
    while (l<r)
    {
        int mid=l+(r-l)/2;
        if (a[mid]>=x) r=mid;
        else l=mid+1;
    }

    if (a[l]==x) return l; //找都了就输出他的位置
    else return -1; // 没找到输出-1
}

int main()
{
    scanf("%d %d",&n,&m); //读入

    for (int i=1 ; i<=n ; i++)
        scanf("%d",&a[i]); //还是读入

    for (int i=1 ; i<=m ; i++)
    {
        scanf("%d",&q);
        int ans=find(q); //看看查找的结果
        printf("%d ",ans); //输出
    }

    return 0;
}
```

2. 洛谷 -p1663.银行贷款

银行贷款

题目描述

当一个人从银行贷款后，在一段时间内他（她）将不得不每月偿还固定的分期付款。这个问题要求计算出贷款者向银行支付的利率。假设利率按月累计。

输入格式

三个用空格隔开的正整数。

第一个整数表示贷款的原值 w_0，第二个整数表示每月支付的分期付款金额 w，第三个整数表示分期付款还清贷款所需的总月数 m。

输出格式

一个实数，表示该贷款的月利率（用百分数表示），四舍五入精确到 $0.1%$。

数据保证答案不超过 $300.0%$。

样例 #1

样例输入 #1

```
1000 100 12
```

样例输出 #1

```
2.9
```

提示

数据保证，$1 \leq w_0, w\leq 2^{31}-1$，$1 \leq m\leq 3000$。

参考题解

预备知识——pow（）函数 pow（n，m）返回n的m次方（值为double）需要定义头文件cmath调用哦

数学推导： 设读入的三个数分别为n,m,k.

先写出方程看看：

k ∑{m*[1/（1+ans)]^i}=n

i=1

如果将ans看作自变量，n看作因变量时

容易证明，此函数具有单调性。

所以二分枚举ans的值，再进行判断即可

化简得后，用等比数列的求和公式得

[1/（1+ans)]^i=1-n/m*ans 小提示： 注意，月利率可能大于1（什么亏心银行）！

```
#include<bits/stdc++.h>
using namespace std;
double n,m,k,l,r;
bool pd(double x){//判断当前答案是否满足
    return (pow(1.0/(1.0+x),k)>=1-n/m*x);//pow函数，上文已讲
}
int main(){
    cin>>n>>m>>k;
    ////////////////////二分模板
    l=0;r=10;//月利率可能大于1
    while(r-l>=0.0001){//注意精度问题
        double mid=(l+r)/2;
        if(pd(mid))r=mid;
        else l=mid;
    }
    ////////////////////
    cout<<fixed<<setprecision(1)<<l*100;//输出一位小数哦
    return 0;
}
```

注1：题解均来自洛谷官方网站

注2：欢迎大家后续通过ai与洛谷进行更进一步的学习，有什么不同想法欢迎一起讨论。

返回目录