若以有序表表示的静态查找表，则查找过程可以基于**折半**进行。折半也成二分，所以叫二分查找，或折半查找。

作为**二分查找的对象**的数据必须是顺序存储的**有序表**，通常是定位按关键字值从小到大排列的表。

**基本思想**

首先取整个有序表的中间记录的关键字值与给定值相比较，若相等，则查找成功。否则以中间位置的数据元素为分界点，将查找表分为左右两个子表，并判断待查找的关键字key是在左子表还是右子表，再在左或右子表重复上述操作，直到找到关键字为key的记录或者子表长度为0.

**步骤**

引进low、high、mid三个数值，分表表示待查找区域的第一条记录、最后一条记录和中间记录的数组下标。

1.初始化，low=0，high=array.length-1

2.当high>=low时，重复下列步骤

（1）mid=（low+high）/2 向下取整

（2）若key与array[mid]的关键字值相等，查找成功，返回mid

（3）若key与array[mid]的关键字不相等

key<array[mid] 🡪 high=mid-1

key>array[mid] 🡪low=mid+1

3.当low>high时，查找失败，返回-1

代码：

public static int binarySearch(int[] array,int key){

int low=0,high=array.length-1; //初始化low、high、mid指针

int mid;

while(high>=low){

mid=(low+high)/2; //当前比较的位置

if(key==array[mid]) //查找成功，返回下标

return mid;

else if(key<array[mid]) //当key值比当前比较的值小，缩小范围到前半段

high=mid-1;

else //当key值比当前比较的值大，缩小范围到后半段

low=mid+1;

}

return -1; //当low>high，说明遍历完毕，没有返回值，查找失败返回-1

}

**平均查找长度ASL**

线性表被多次平分

最坏情况下，查找次数k=

一次查找，比较的节点数p有一个（20）

二次查找，比较的节点数p有两个（21）

i次查找，比较的节点数p有2**i-1**个

在每个节点朝找概率相同的情况下，平均查找长度为

ASL==-1

不管成功与失败，二分查找比顺序查找要快很多，但是**线性表必须按照要求进行排序**，排序的最佳时间复杂度为O（n）

线性表的二分查找**仅仅适合于顺序存储结构**，对于动态查找表，顺序存储的插入删除等运算都不是很方便，所以**二分查找一般适用与一旦建立就很少需要改动而又经常需要查找的静态顺序表。**