二分搜索

hyliu

2019 年 8 月 17 日

目录

1 二分搜索

1.1 应用场景

在有序数列 $a_1, a_2, a_3, ..., a_n$ 中找到第一个大于 value 的数字 a_i 。
时间复杂度: O(log(n))

1.2 代码实现

1.2.1 找到第一个大于 value 的位置

```
int upper_binary_search(int num[],int value,int n){
    int l = 0, r = n-1 ,pos = -1;
    while (l<r){
```

```
        int mid = l + ((r-l)>>1);
        if (num[mid] > value)
            r = mid - 1;
        else
            l = mid + 1 ;
    }
    return l;
}
```

1.2.2 找到第一个大于等于 value 的位置

```
int lower_binary_search(int num[],int value,int n){
    int l = 0, r = n-1 ,pos = -1;
    while (l<r){
        int mid = l + ((r-l)>>1);
        if (num[mid] >= value)
            r = mid - 1;
        else
            l = mid + 1 ;
    }
    return l;
}
```

1.3 STL 实现

```
iterator lower_bound( const key_type &key )
// 返回一个迭代器，指向键值>= key的第一个元素。

iterator upper_bound( const key_type &key )
//返回一个迭代器，指向键值> key的第一个元素。
```

1.3.1 举例

举例如下:(有序数组)一个数组 number 序列为:4,10,11,30,69,70,96,100.设要插入数字 3,9,111.pos 为要插入的位置的下标, 则

```
pos = lower_bound( number, number + 8, 3) - number;
//pos = 0.即number数组的下标为0的位置。
pos = lower_bound( number, number + 8, 9) - number;
//pos = 1, 即number数组的下标为1的位置（即10所在的位置）。
pos = lower_bound( number, number + 8, 111) - number;
//pos = 8, 即number数组的下标为8的位置
//（但下标上限为7, 所以返回最后一个元素的下一个元素）。
```