二分总结

在数组A中查找target

```
public int func(int A[],int target){
    int lo=0,hi=A.length-1;
    while(lo<=hi){
        int mid=(hi+lo)/2;
        if(A[mid]>target)
            hi=mid-1;
        else if(A[mid]<target)
            lo=mid+1;
        else
            return true;
    }
    return false;
}
```

两个二分衍生的函数

lower_bound 寻找第一个大于等于target的数，初始传入[0,A.length]，因为我们可能找到符合条件的元素不在数组之中，例如[1,2,3,4]而target=5,数组中不存在大于等于target的数，我们应该返回A.length，表示如果存在这个数，它应该的存在的位置

```
public int lower_bound(int A[],int target){
    int lo=0,hi=A.length;
    while(lo<hi){
        int mid=(hi+lo)/2;
        // 如果A[mid]大于等于target，那么hi不动，让lo逼近hi
        if(A[mid]>=target)
            hi=mid;
        else
            lo=mid+1;    //如果A[mid]<target，那么往后找
    }
    return lo;
}
```

upper_bound 寻找第一个大于target的数，初始传入[0,A.length]
假如数组中没有大于target的数，那么会返回A.length，也就是为了保持升序序列，插入时target应该在A中插入的位置

观察下面代码，其实和lower_bound差别就在于A[mid]>target这里

```java
public int upper_bound(int A[],int target){
    int lo=0,hi=A.length;
    while(lo<hi){
        int mid=(hi+lo)/2;
        // 如果A[mid]大于target，那么hi不动，让lo逼近hi
        if(A[mid]>target)
            hi=mid;
        else
            lo=mid+1;    //如果A[mid]<target，那么往后找
    }
    return lo;
}
```