二分查找

题目描述

给定长度为n的严格递增序列L，使用二分查找搜索元素k的序列下标i，使得$L_i = k$。

输入格式

每个数据一行$n+2$个数字。

其中，第一个数字为序列长度n；第2至第$n+1$个数字为序列L；第$n+2$个数字为需要查找的值k。

输出格式

一个数字i。

下标从0开始，如果数字不存在，则输出-1。

样例 #1

样例输入 #1

```
5 1 2 3 4 5 4
```

样例输出 #1

```
3
```

样例 #2

样例输入 #2

```
0 0
```

样例输出 #2

```
-1
```

提示

$0 \leq n \leq 100000$

$\forall i < L.length - 1, L_i < L_{i+1}$

k, L_i在int范围内

---

solution using built-in `bisect`

```
import bisect


def solve(lst: list, num: int) -> int:
    if num in lst:
        return bisect.bisect_left(lst, num)
    else:
        return -1


if __name__ == '__main__':
    from contextlib import suppress

    with suppress(EOFError):
        while True:
```

```
            s = input().split()
            print(solve(list(map(int, s[1:-1])), int(s[-1])))
```

solution using built-in `sort`

```
def solve(lst: list, num: int) -> int:
    try:
        return lst.index(num)
    except ValueError:
        return -1


if __name__ == '__main__':
    from contextlib import suppress

    with suppress(EOFError):
```

```
        while True:
            s = input().split()
            print(solve(list(map(int, s[1:-1])), int(s[-1])))
```

real solution

```
def solve(lst: list, num: int) -> int:
    lth = len(lst)
    if lst[0] < num < lst[-1]:
        l, r = 0, lth - 1
        while l < r:
            index = (l + r) // 2
            value = lst[index]
            if value < num:
                l = index
            elif value > num:
```

```
                r = index
            else:
                return index
    return -1


if __name__ == '__main__':
    from contextlib import suppress

    with suppress(EOFError):
        while True:
            s = input().split()
            print(solve(list(map(int, s[1:-1])), int(s[-1])))
```