Recursion review

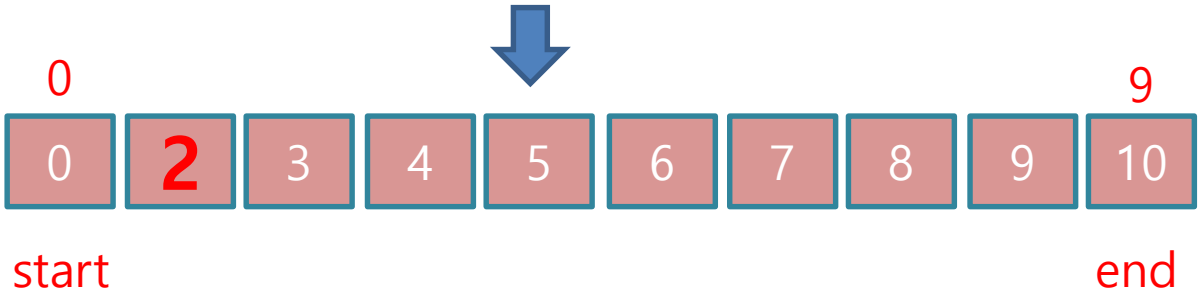
Binary search

Binary search
: 알고리즘

Target : 2일 때

```
mid = (start + end)//2
```

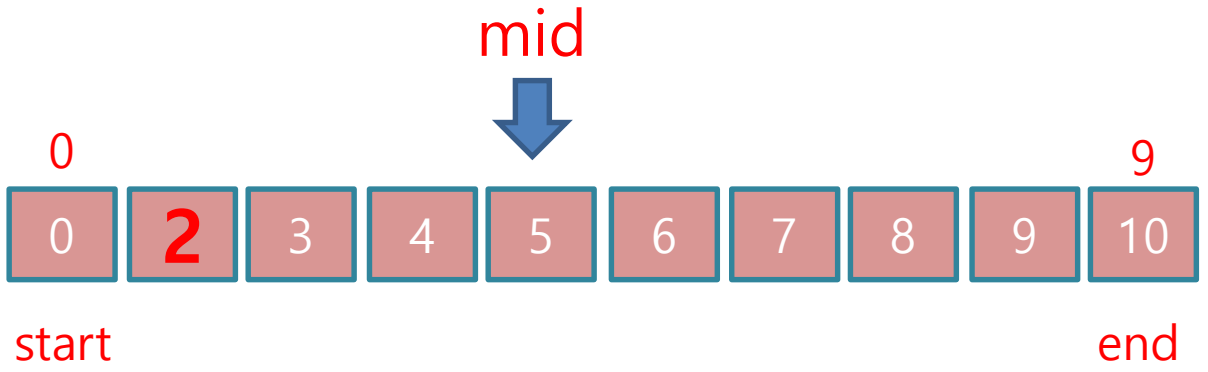
mid : (0 + 9)//2 = 4



Binary search
: 알고리즘

Target : 2일 때

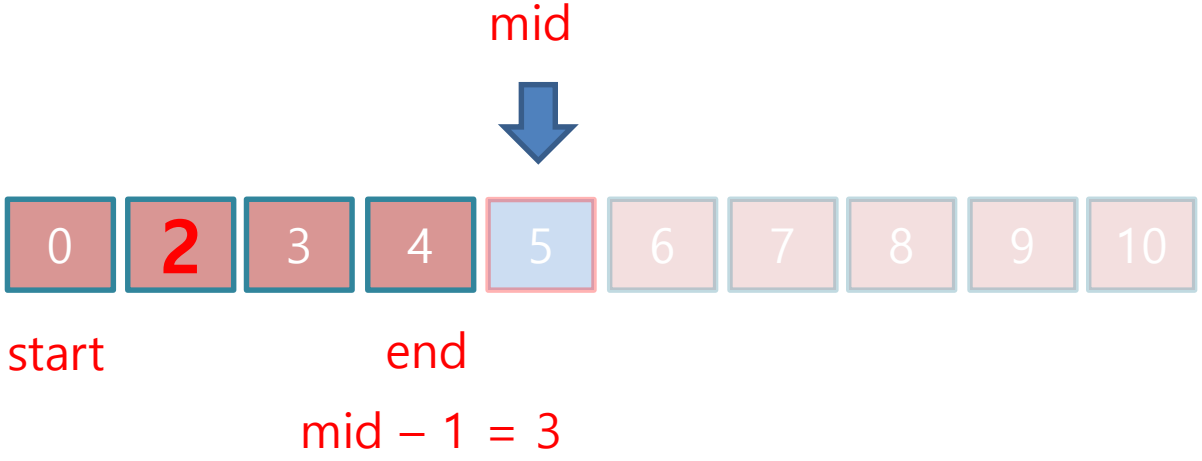
```
elif data[mid] > target:
    end = mid - 1
```



Binary search
: 알고리즘

Target : 2일 때

```
elif data[mid] > target:
    end = mid - 1
```

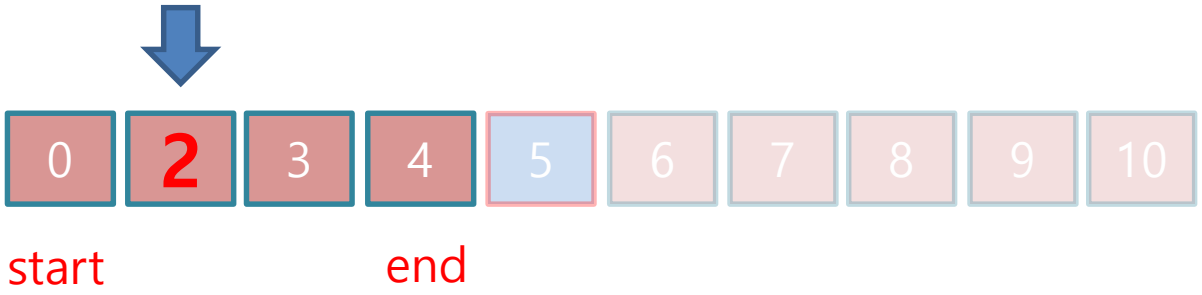


Binary search
: 알고리즘

Target : 2일 때

```
mid = (start + end)//2
```

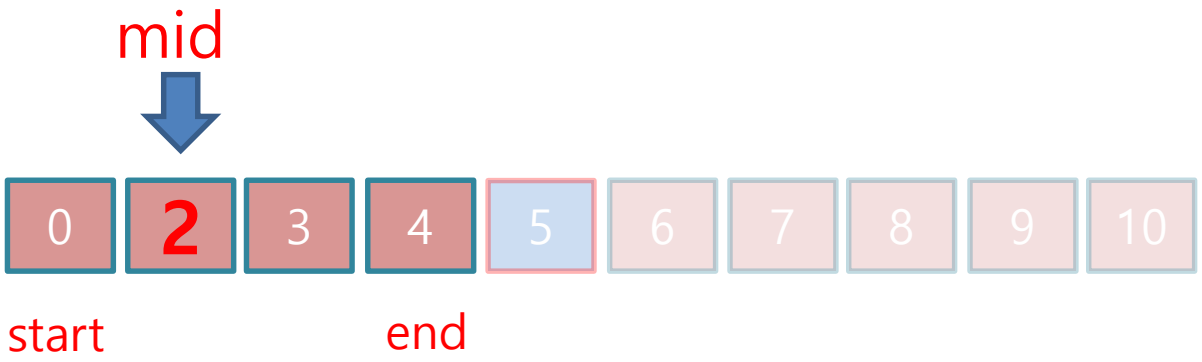
mid : (0 + 3)//2 = 1



Binary search
: 알고리즘

Target : 2일 때

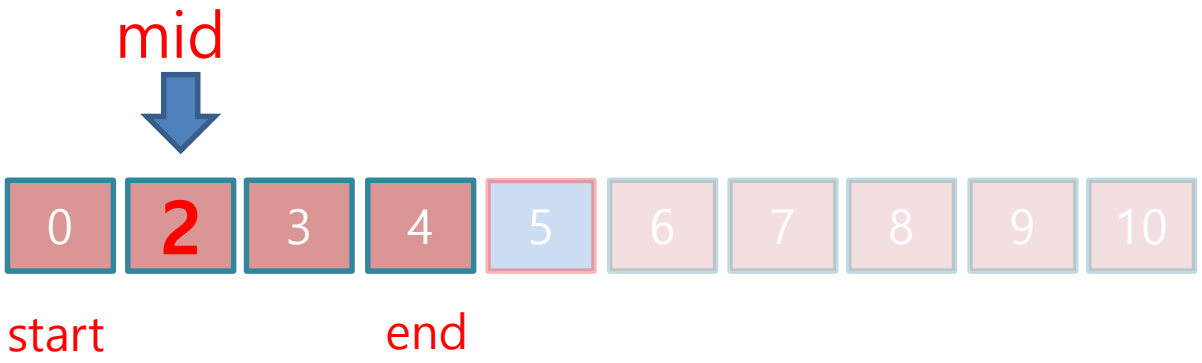
```
if data[mid] == target:
    return mid
```



Binary search
: 알고리즘

Target : 2일 때

```
if data[mid] == target:
    return mid
```

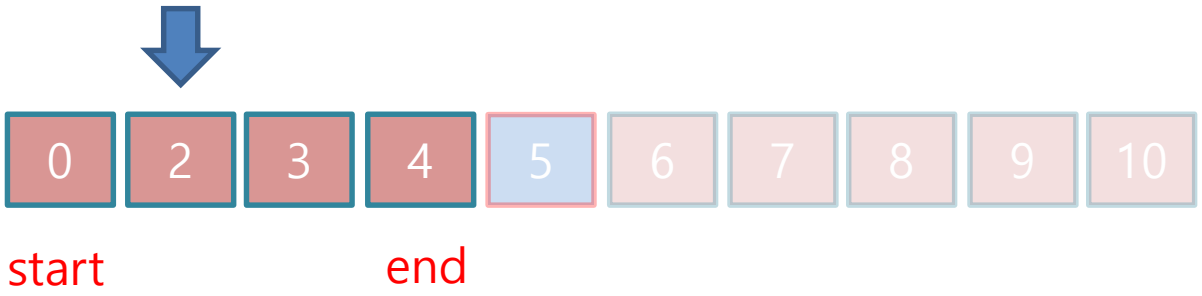


Target과 mid의 값이 같으므로 인덱스 1 반환

Binary search
: 알고리즘

만약 찾는 원소가 없을 때
: target : 1

```
elif data[mid] > target:
    end = mid - 1
```

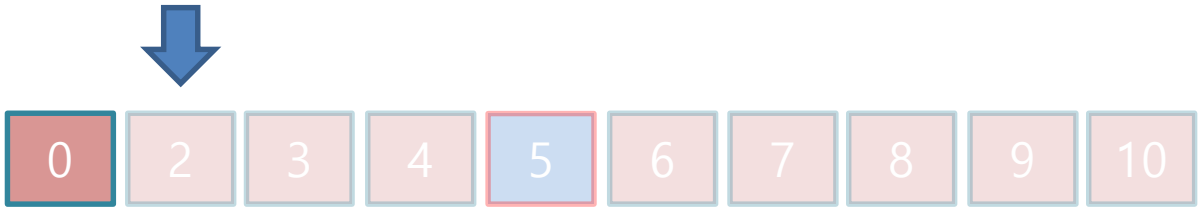


Binary search
: 알고리즘

만약 찾는 원소가 없을 때
: target : 1

```
elif data[mid] > target:
    end = mid - 1
```

mid



start

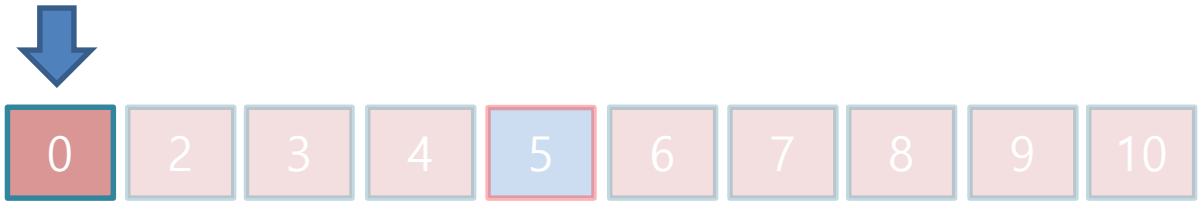
end = mid - 1

Binary search
: 알고리즘

만약 찾는 원소가 없을 때
: target : 1

```
mid = (start + end)//2
```

mid = (0 + 0)//2 = 0

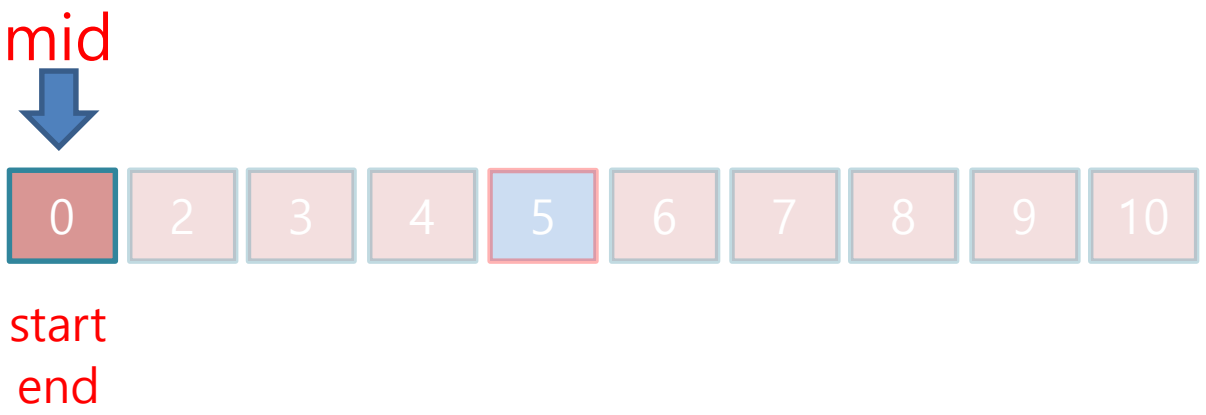


start
end

Binary search
: 알고리즘

만약 찾는 원소가 없을 때
: target : 1

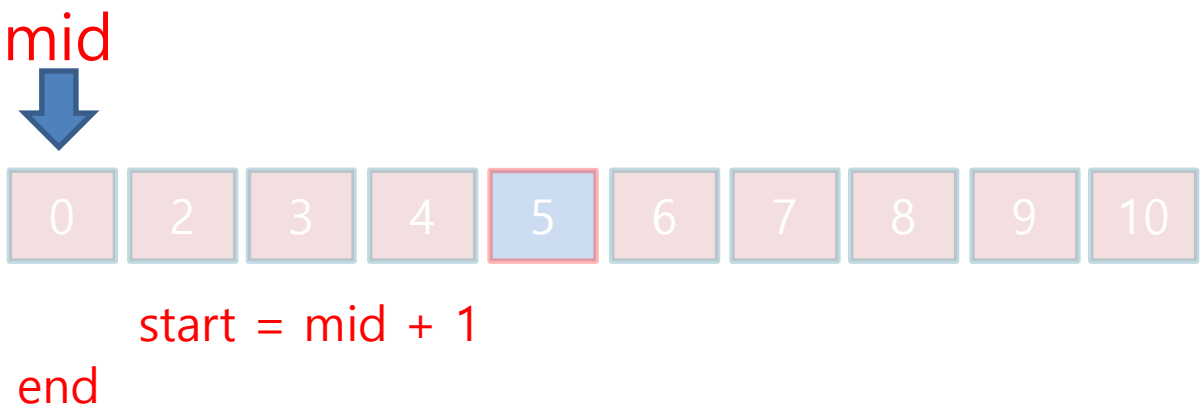
data[mid] < target



Binary search
: 알고리즘

만약 찾는 원소가 없을 때
: target : 1

찾는 원소가 없을 경우
Start가 end보다 커진다

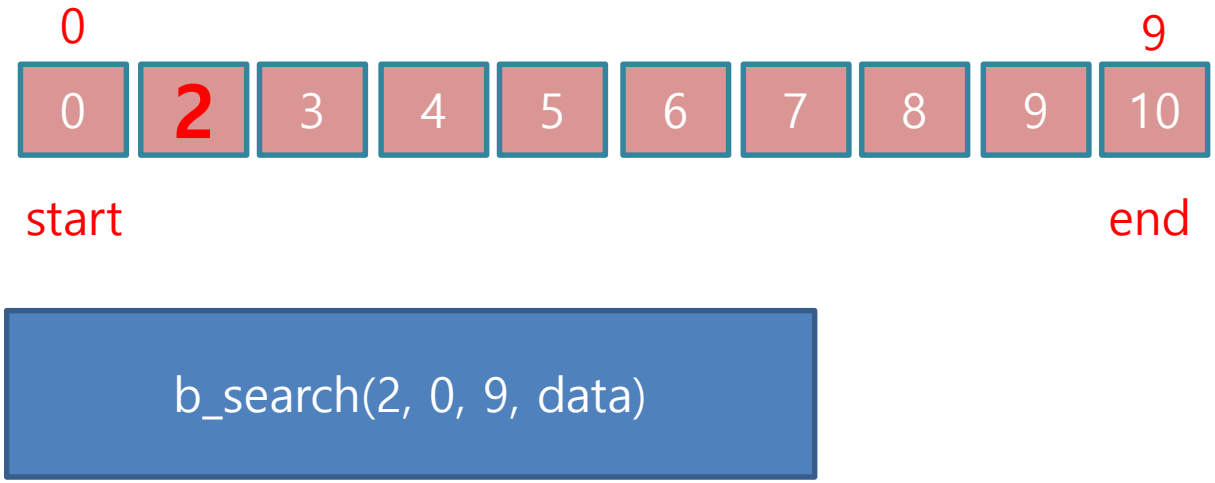


Binary search
: 재귀적 구현

```
def b_search(target, start, end, data):
    if start > end:
        return None

    mid = (start + end)//2

    if data[mid] == target:
        return mid
    elif data[mid] > target:
        end = mid - 1
    else:
        start = mid + 1

    return b_search(target, start, end, data)
```

Binary search
: 재귀적 구현

Target : 2일 때



Binary search
: 재귀적 구현

Target : 2일 때