

Binary Search

정찬웅

INDEX

01

About

이분탐색에 대해
알고 있으면 좋은가?

02

Example

반복문
재귀

03

Use

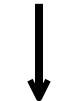
그래서 언제
사용하는 것이 좋을까?

About

범위를 반으로 쪼갠다.

중간보다 낮은지 높은지 확인한다.

중간보다 낮다면(혹은 높다면) 나머지 반을 버린다.



범위를 반으로 쪼갠다.

중간보다 낮은지 높은지 확인한다.

중간보다 낮다면(혹은 높다면) 나머지 반을 버린다.



범위를 반으로 쪼갠다.

중간보다 낮은지 높은지 확인한다.

중간보다 낮다면(혹은 높다면) 나머지 반을 버린다.

About

1000 길이의 리스트를 탐색해서 978번째의 요소를 찾는다면 ?

for, while

0부터 978까지 1씩 더해가면서 탐색한다.

내가 0부터 978까지 숫자를 세본다고 생각해보자
굉장히 힘들다..

binary search

500보다 큰가 ? yes

750보다 큰가 ? yes

875보다 큰가 ? yes

930보다 큰가 ? yes

...

...

우리는 이걸 많이 해봤다.. 어디서 ? 술마실때 !

About

술게임으로 많이 했던 익숙한 게임이니까 코딩으로도 충분히 할 수 있다 !

안되겠으면 술한잔하면서 다시 원리를 깨우쳐보자..

어쨌든 이진탐색을 활용한다면
매우 큰 수를 탐색할 때의 시간을 기하급수적으로 감소시킬 수 있다.

Example

반복문

```
int binarySearch (int arr[], int low, int high, int key) {  
    while (low <= high) {  
        int mid = low + (high-low) / 2;  
  
        if (arr[mid] == key) // 종료 조건 1 검색 성공.  
            return mid;  
        else if (arr[mid] > key)  
            high = mid - 1;  
        else  
            low = mid + 1;  
    }  
    return -1 ; // 종료 조건2 (low > high) 검색 실패.  
}
```

재귀

```
int binarySearch (int arr[], int low, int high, int key) {  
    if (low > high) // 종료 조건 2 검색 실패.  
        return -1;  
  
    int mid = low + (high-low)/2;  
  
    if (arr[mid] == key) // 종료 조건 1 검색 성공.  
        return mid;  
    else if (arr[mid] > key)  
        return binarySearch(arr, low, mid-1, key);  
    else  
        return binarySearch(arr, mid+1, high, key);  
}
```

Use

그래서 언제 사용하는것이 좋은가 ?

문제에서 "저는 이진탐색을 사용해서 풀어주세요 !!" 라고 하지 않기 때문에 적절한 경우에 사용하는것이 가장 중요한 문제일 것이다.

1. 정렬이 가능한지 확인하자.

- 정렬이 되어있다면 더 좋음

2. 무작위 요소 접근에 대해 $O(1)$ 의 시간복잡도를 가져야 한다.

- 요소를 접근할 때, 인덱스를 통해 바로 접근하지 못하는 자료구조라면 사용하지 않는것이 좋다.
- 파이썬 리스트의 경우, 인덱스를 통해 $O(1)$ 의 복잡도로 바로 접근할 수 있어서 사용 하기 좋음
 - $O(1)$ 로 접근 가능하다면 총 시간 복잡도는 $O(N\log N)$

Use

유의사항

1. 이분탐색보다 더 효율이 좋은 탐색이 있을 수 있다. (문제마다 다름)
 - 이분탐색도 효율적이지만, 자료구조와 문제에 따라 효율이 달라질 수 있으므로 주의하자
 - 다른 탐색방법을 숙지해서 필요에 맞게 사용하는것이 제일 좋음
2. 시간복잡도에 대해 생각해보자.
 - 시간복잡도는 알고리즘 문제 해결에 있어 가장 중요한 요소 중 하나이다.
 - 아무리 구현을 잘 해도 시간초과가 된다면 틀리므로,
자신이 자주 사용하는 자료구조의 시간복잡도는 알고 있으면 좋을 것이다.

감사합니다