7. 이진탐색

이진 탐색 : 반으로 쪼개면서 탐색하기

이진 탐색은 배열 내부의 데이터가 정렬되어 있어야만 사용할 수 있는 알고리즘이다. 데이터가 무작위일 때는 사용할 수 없지만, 이미 정렬되어 있다면 매우 빠르게 데이터를 찾을 수 있다. 이진 탐색은 탐색 범위를 절반씩 좁혀가며 데이터를 탐색하는 특징이 있다.

이진 탐색에선 시작점, 끝점, 중간점 3개의 변수를 사용한다. 찾으려는 데이터와 중간점 위치에 있는 데이터를 반복적으로 비교해 원하는 데이터를 찾는 것이다.

이진 탐색의 경우 한 번 확인할 때마다 확인하는 원소의 개수가 절반씩 줄어든다는 점에서 시간 복잡도가 O(logN) 이다.

이진 탐색 구현 방법은 2가지가 있다. 하나는 재귀, 하나는 반복문을 이용하는 방법이다.

```
# 재귀를 이용한 이진 탐색
def binary_search(array, target, start, end):
  if start > end:
    return None

  mid = (start + end) // 2

  if array[mid] == target:
    return mid
  elif array[mid] > target:
    return binary_search(array, target, start, mid - 1)
  else:
    return binary_search(array, target, mid + 1, end)


# 반복문을 이용한 이진 탐색
def binary_search(array, target, start, end):
  while start <= end:
    mid = (start + end) // 2

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

```
    elif array[mid] > target:
      end = mid - 1
    else:
      start = mid + 1

  return None
```

트리 자료구조

이진 탐색은 전제 조건이 데이터 정렬이다. 예를 들어 동작하는 프로그램에서 데이터를 정렬해두는 경우가 많으므로 이진 탐색을 효과적으로 사용할 수 있다. 데이터베이스는 내부적으로 대용량 데이터 처리에 적합한 트리 자료구조를 이용해 항상 데이터가 정렬되어 있다.

트리 자료구조는 노드와 노드의 연결로 표현하며 여기서 노드는 정보의 단위로 어떠한 정보를 가지고 있는 개체로 이해할 수 있다. 트리 구조는 몇 가지 주요한 특징이 있다.

- 트리는 부모 노드와 자식 노드의 관계로 표현된다.
- 트리의 최상단 노드를 루트 노드라고 한다.
- 트리의 최하단 노드를 단말 노드라고 한다.
- 트리에서 일부를 떼어내도 트리 구조이며 이를 서브 트리라고 한다.
- 트리는 파일 시스템과 같이 계층적이고 정렬된 데이터를 다루기에 적합하다.

이진 탐색 트리

트리 자료구조 중에서 가장 간단한 형태가 이진 탐색 트리이다. 이진 탐색 트리란 이진 탐색이 동작할 수 있도록 고안된 자료구조로 아래와 같은 특징이 있다.

- 부모 노드보다 왼쪽 자식 노드가 작다.

- 부모 노드보다 오른쪽 자식 노드가 크다.

이진 탐색 트리에서 데이터를 넣고 빼는 방법은 알고리즘보다는 자료구조에 가까우며, 이진 탐색 트리 자료구조를 구현하도록 요구하는 문제는 출제 빈도가 낮으므로, 이 책에서는 이진 탐색 트리를 구현하는 방법을 소개하지는 않는다. 즉, 이진 탐색 트리가 구현되어 있다고 가정하고 데이터를 조회하는 과정만 살펴본다. (책 참조)

빠르게 입력받기

이진 탐색 문제는 입력 데이터가 많거나, 탐색 범위가 매우 넓은 편이다. 예를 들어 데이터의 개수가 1000만개를 넘어가거나 탐색 범위의 크기가 1000억 이상이라면 이진 탐색 알고리즘을 의심해야 한다. 하지만 입력의 개수가 매우 크기 때문에 sys.readline() 함수를 이용해야만 시간 초과를 피할 수 있다.

문제 1. 부품 찾기

동빈이네 전자 매장에는 부품이 N 개 있다. 각 부품은 정수 형태의 고유한 번호가 있다. 어느 날 손님이 M개 종류의 부품을 대량으로 구매하겠다며 당일 날 견적서를 요청했다. 동빈이는 문의한 부품을 모두 확인해 견적서를 작성해야 한다. 이때 가게 안에 부품이 모두 있는지 확인하는 프로그램을 작성하라.

예를 들어 가게의 부품이 총 5개일 때 부품 번호가 다음과 같다고 하자.

- N = 5
- [8, 3, 7, 9, 2]

손님은 총 3개의 부품이 있는지 확인 요청했는데 부품 번호는 다음과 같다.

- M = 3
- [5, 7, 9]

이때 손님이 요청한 부품 번호의 순서대로 부품을 확인해 부품이 있으면 yes, 없으면 no를 출력한다.

이때, 입력 조건은 N이 100만 이하, M이 10만 이하 정수이다.또한 각 부품의 번호는 최대 100만이다.

💡 해결

정렬해서 이진탐색한다.

```
import sys

N = int(sys.stdin.readline())
my_arr = list(map(int, sys.stdin.readline().strip().split()))
M = int(sys.stdin.readline())
want_arr = list(map(int, sys.stdin.readline().strip().split()))

my_arr.sort()

def binary_search(target, start, end):
    while start <= end:
        mid = (start + end) // 2

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

    return -1

for info in want_arr:
    if binary_search(info, 0, N - 1) != -1:
        print('yes', end=' ')
    else:
        print('no', end=' ')
```

문제 2. 떡볶이 떡 만들기

오늘 동빈이는 여행 가신 부모님을 대신해 떡집 일을 한다. 동빈이네 떡볶이 떡은 떡의 길이가 일정하지 않다. 대신에 한 봉지 안에 들어가는 떡의 총 길이는 절단기로 잘라서 맞춰준다.

절단기에 높이H 를 지정하면 줄지어진 떡을 한 번에 절단한다. 높이가 H보다 긴 떡은 H 위의 부분이 잘릴 것이고, 낮은 떡은 잘리지 않는다.

손님이 왔을 때 요청한 총 길이가 M일 때 적어도 M 만큼의 떡을 얻기 위해 절단기에 설정할 수 있는 높이의 최댓값을 구하는 프로그램을 작성하라.

이때, M의 범위가 2000만 이하이다.

풀이

이것도 이진탐색 문제이다. 높이 h를 mid로 생각하고 가장 길이가 긴 떡을 기준으로 start, end를 잡아가며 잘라 테스트한다.