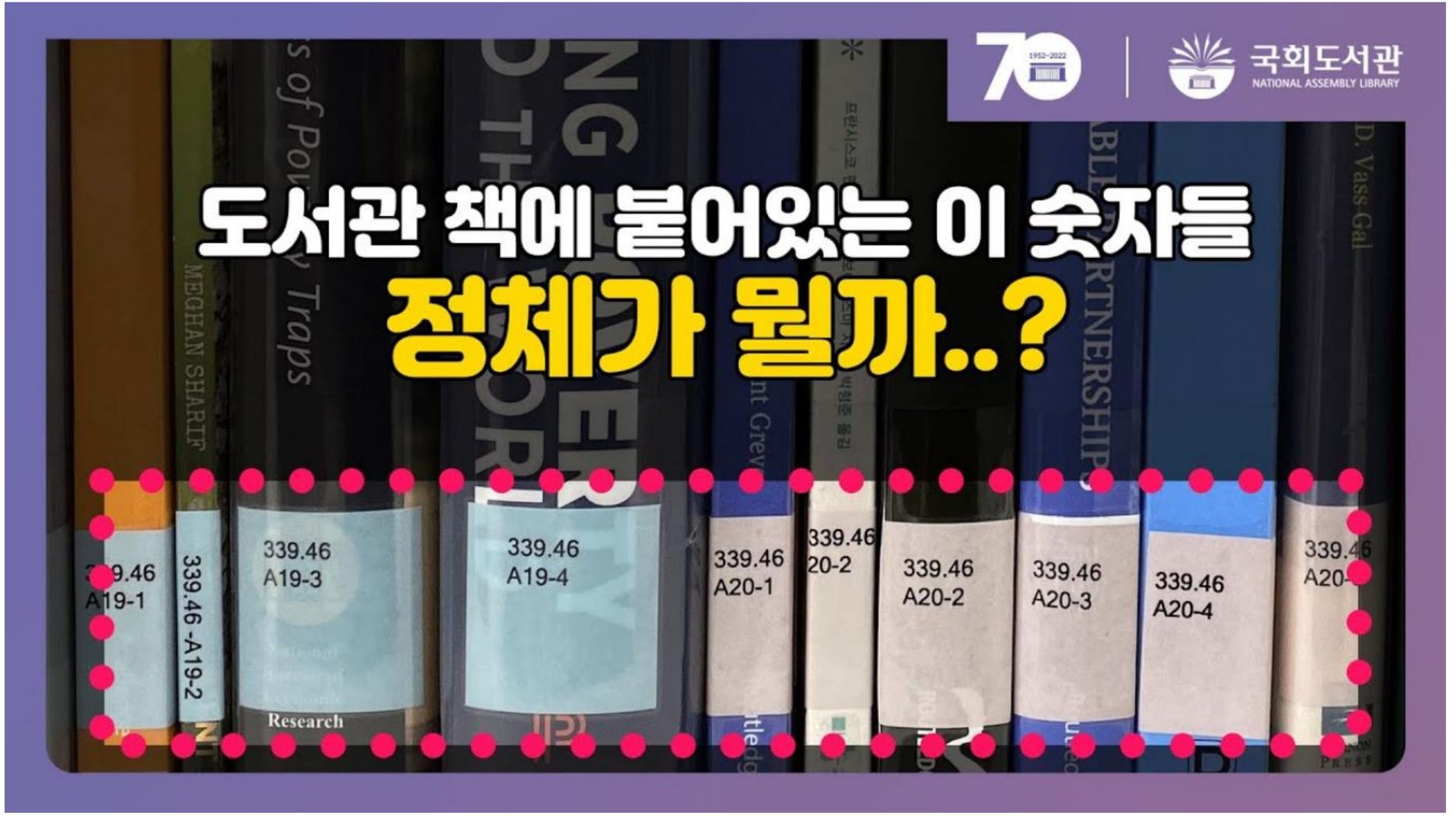
코테스터디 - 탐색

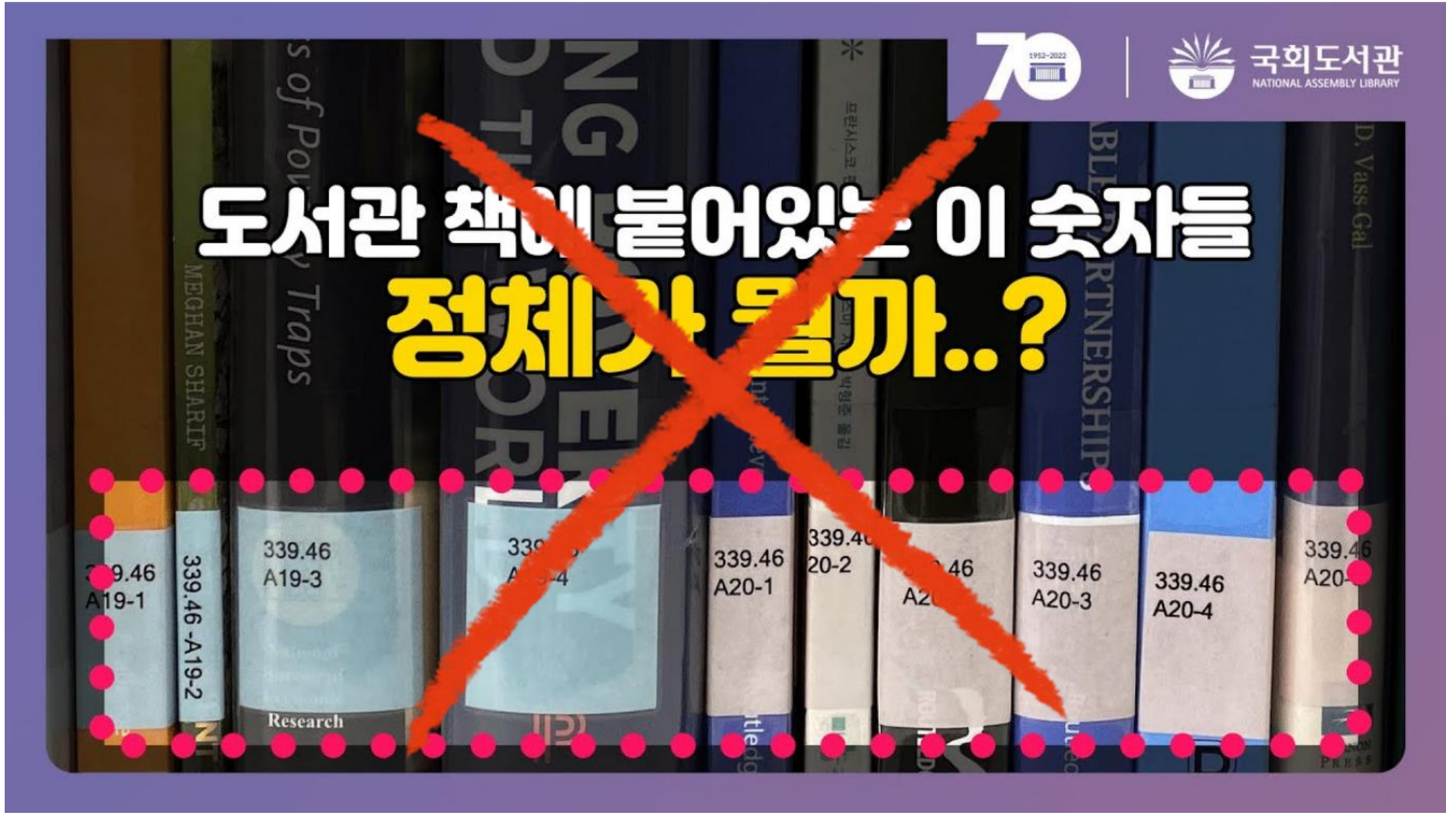
도서관의 사례

원하는 책을 어떻게 찾을것인가?

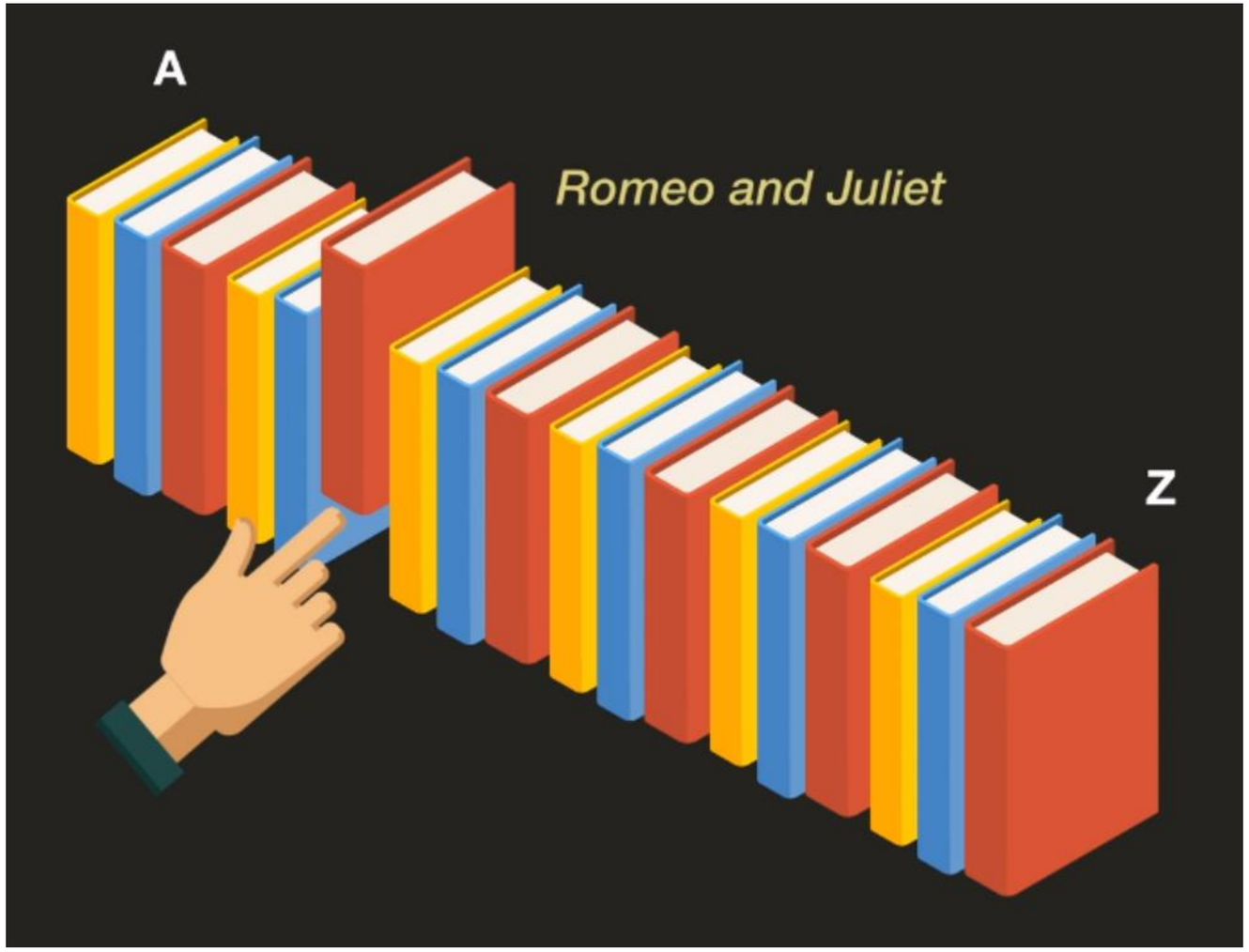
택이 있는 경우 = 택을 따라간다



택이 없는경우?


1952-2022
국회도서관
NATIONAL ASSEMBLY LIBRARY
도서관 책에 붙어있는 이 숫자들
정체가 뭘까..?
339.46
A19-1
339.46 -A19-2
339.46
A19-3
339.46
A20-1
339.46
A20-3
339.46
A20-4
Research

택이 없는경우? = 선형탐색



선형 탐색의 방식

1. 배열/리스트를 **순회**합니다.
2. 배열/리스트를 순회하면서 **‘하나씩 값을 비교’**합니다.
3. 원하는 값을 찾는 경우 순회를 **멈추고 값을 반환**합니다.

선형 탐색의 방식

코딩을 하며 원하는 값이 어떻게 찾아지는지 확인해봅시다!

선형 탐색의 방식 - 코드

```java
public class Main {
    no usages
    public static void main(String[] args) throws IOException {
        String []arr = {"킥","퓝킥","캬캬","들켰다!","호호","메롱","안돼~","에베벱","아닌데~", "집가고싶다"};
        for(int i=0; i<arr.length; i++){
            if(arr[i] == "들켰다!"){
                System.out.println(i+1 +"번만에 찾았다!");
                System.out.println(arr[i]);
                break;
            }
        }

    }

}
```

report

코테 스터디

이분탐색

3주차 코테스터디 - 이분탐색

CONTETS

- 알고리즘 이해
- 알고리즘 분석 및 실습
- 문제 풀기 (쉬운 문제 ~)

불편함 해소!

알고리즘 이해 

내가 왜 코딩테스트를 준비 해야하지...?

알고리즘 이해

내가 왜 코딩테스트를 준비 해야하지...?

그리드 알고리즘

5585번 | 제출 | 맞힌 사람 | 숏코딩 | 재채점 결과 | 채점 현황 | 내 제출 | 질문 게시판

거스름돈 다국어

☆ 한국어

브론즈 II

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	45536	29835	25434	65.476%

문제

타로는 자주 JOI잡화점에서 물건을 산다. JOI잡화점에는 잔돈으로 500엔, 100엔, 50엔, 10엔, 5엔, 1엔이 충분히 있고, 언제나 거스름돈 개수가 가장 적게 잔돈을 준다. 타로가 JOI잡화점에서 물건을 사고 카운터에서 1000엔 지폐를 한장 냈을 때, 받을 잔돈에 포함된 잔돈의 개수를 구하는 프로그램을 작성하시오.

알고리즘 이해

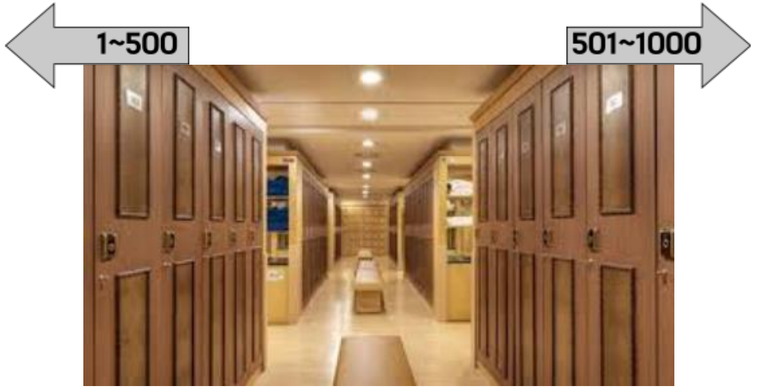
이분 탐색 알고리즘 (BINARY SEARCH)

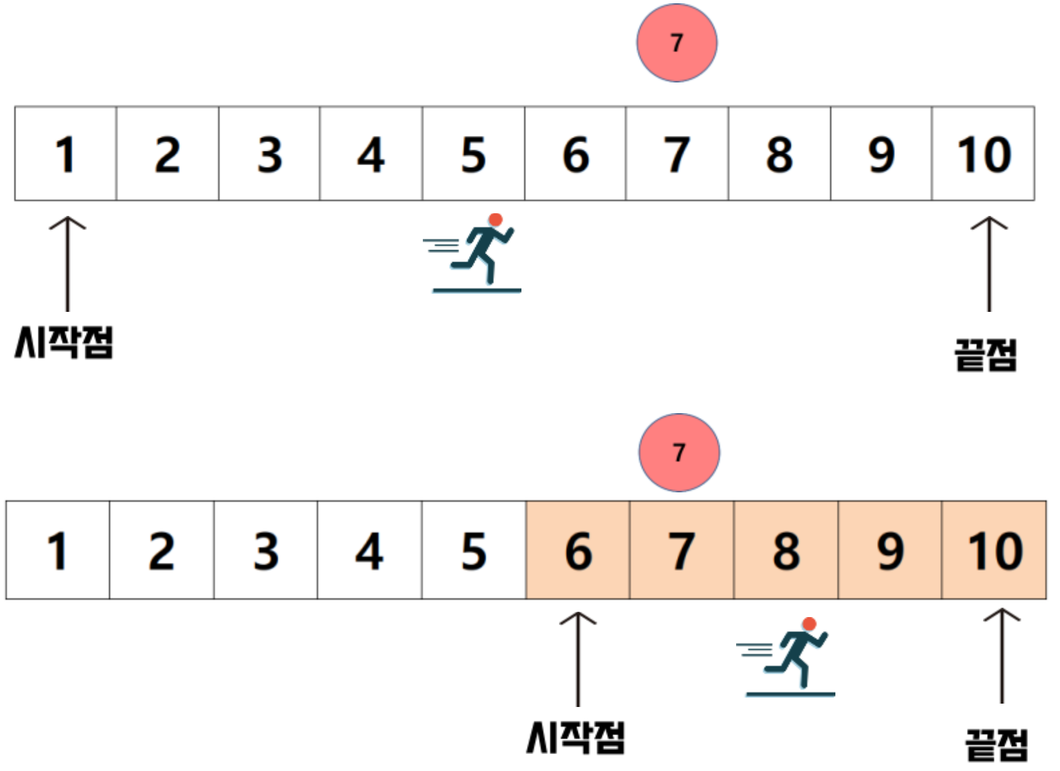
- 정렬돼 있는 데이터에서 특정한 값을 찾아내는 알고리즘
- 탐색 범위를 반으로 나누어 찾는 값을 포함하는 범위를 좁혀가는 방식으로 동작한다

알고리즘 이해

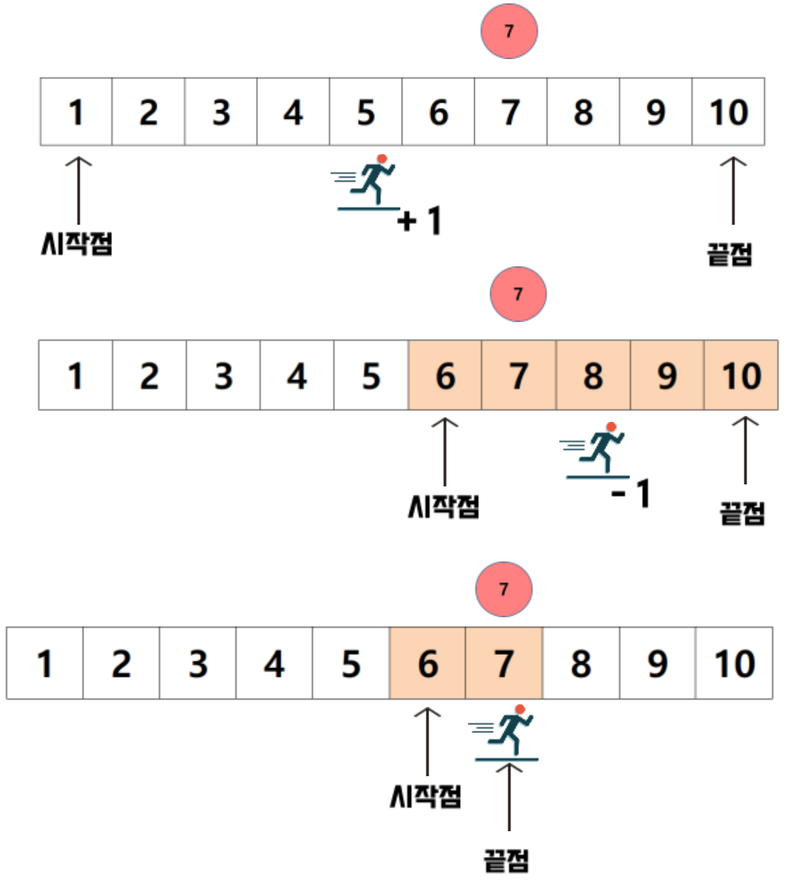
알고리즘 이해

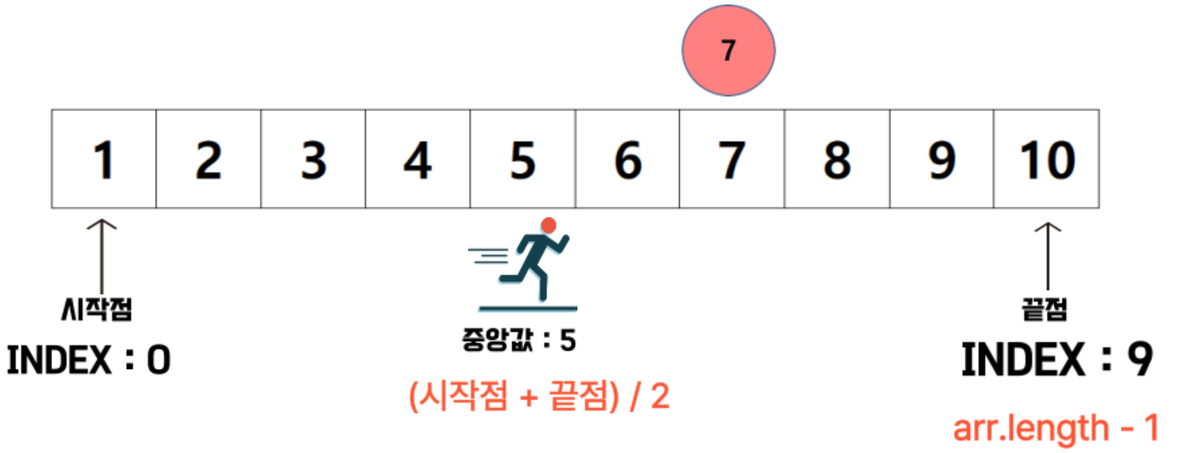
알고리즘 이해





알고리즘 이해


7
1 2 3 4 5 6 7 8 9 10
시작점
+1
끝점
7
1 2 3 4 5 6 7 8 9 10
시작점
-1
끝점
7
1 2 3 4 5 6 7 8 9 10
시작점
끝점


7
1 2 3 4 5 6 7 8 9 10
시작점
INDEX : 0
중앙값 : 5
(시작점 + 끝점) / 2
끝점
INDEX : 9
arr.length - 1

알고리즘 이해

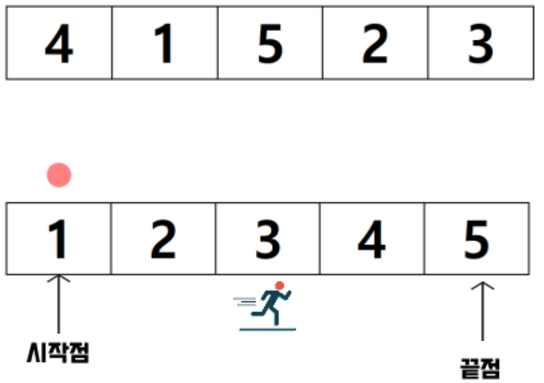
실습

실습 - 수찾기 (백준 1920번) 풀이 설명

```java
import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        int N = in.nextInt();  // 배열의 크기 N을 입력 받음
          int[] arr = new int[N];  // 크기가 N인 정수 배열을 선언

          for(int i = 0; i < N; i++) { // [4, 1, 5, 2,3]
            arr[i] = in.nextInt();  // 배열의 각 요소에 입력 받은 값을 저장
          }

        // 배열의 값을 출력
          System.out.print("Before array : ");
        for(int i = 0; i < N; i++) {
            System.out.print(arr[i]);  // 각 요소를 한 줄씩 출력
          }

        System.out.println();
        // 배열을 정렬
          Arrays.sort(arr); );            //     [1, 2, 3, 4, 5]

        // 정렬된 배열의 값을 출력
          System.out.print("After array : ");
        for(int i = 0; i < N; i++) {
            System.out.print(arr[i]);  // 각 요소를 한 줄씩 출력
          }
    }
}
```

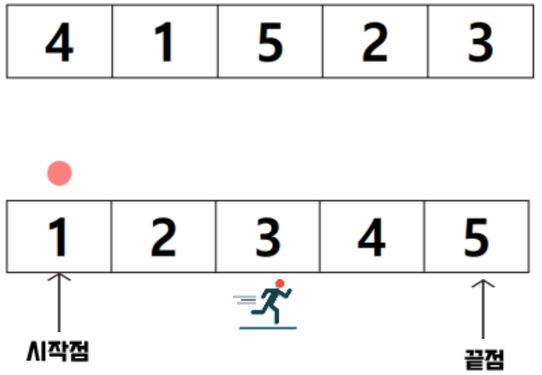
- 첫째 줄: 자연수 N (1 이상 100,000 이하) 5
- 둘째 줄: N개의 정수 A[1], A[2], ..., A[N] 4 1 5 2 3
- 셋째 줄: 자연수 M (1 이상 100,000 이하) 5
- 넷째 줄: M개의 수들 1 3 7 9 5

N 길이의 배열에 M 개의 수들이 있는 배열에 A[N] 값이 있는지 찾아내는 방법이다.



실습 - 수찾기 (백준 1920번) 풀이 설명

```java
import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        int N = in.nextInt();  // 배열의 크기 N을 입력 받음
          int[] arr = new int[N];  // 크기가 N인 정수 배열을 선언

          for(int i = 0; i < N; i++) { // [4, 1, 5, 2,3]
            arr[i] = in.nextInt();  // 배열의 각 요소에 입력 받은 값을 저장
          }

        // 배열의 값을 출력
          System.out.print("Before array : ");
        for(int i = 0; i < N; i++) {
            System.out.print(arr[i]);  // 각 요소를 한 줄씩 출력
          }

        System.out.println();
        // 배열을 정렬
          Arrays.sort(arr); );            //     [1, 2, 3, 4, 5]

        // 정렬된 배열의 값을 출력
          System.out.print("After array : ");
        for(int i = 0; i < N; i++) {
            System.out.print(arr[i]);  // 각 요소를 한 줄씩 출력
          }
    }
}
```



출력 결과

```
C:\Users\student\.jdks\openjdk-2
5
4 2 1 3 5
42135
Sorted array:12345
```

실습 - 수찾기 (백준 1920번) 풀이 설명

```java
//(정렬된 배열:arr, 찾고자 하는 값:key)
public static void binarySearch(int[] arr, int key){
    int lo = 0;                 // 탐색 범위의 왼쪽 끝 인덱스
    int hi = arr.length - 1;    // 탐색 범위의 오른쪽 끝 인덱스

    // lo가 hi보다 커지기 전까지 반복한다.
    while (lo <= hi) {
        int mid = (lo + hi) / 2;    // 중간위치를 구한다.

        // key값이 중간 위치의 값보다 작을 경우
        if (key < arr[mid]) {
            hi = mid - 1;
        }
        // key값이 중간 위치의 값보다 클 경우
        else if (key > arr[mid]) {
            lo = mid + 1;
        }
        // key값과 중간 위치의 값이 같을 경우
        else {
            System.out.println(1);
            return;
        }
    }

    // 찾고자 하는 값이 존재하지 않을 경우
    System.out.println(0);
}
}
```

4	1	5	2	3

출력 결과

M으로 들어오는 값	M에 대한 결과 값
1	1
3	1
7	0
9	0
5	1

```
C:\Users\student\.jdks\openjdk-22.0.1\bin\java.exe "-javaa
5
4 1 5 2 3
41523
Sorted array:12345
5
1 3 7 9 5
1
1
0
0
```

이분 탐색 알고리즘의 사용 조건

1. 정렬된 배열(or리스트)이거나 정렬할 수 있는 배열(or리스트)이어야 가능합니다.

간단한영상

해시탐색

우리는 어떻게 물건을 계산할까요?

물건에 번호를 매김으로써!

해시란?

값을 부여하여서 구분하는것 !

Key와 Value

해시는 Key와 Value로 이루어져 있습니다!
물건 계산을 비유로 들면, "바코드의 값" 이라는 "key"와
신라면에 해당하는 가격이 "value"에 해당합니다.

해싱과 해시함수

해싱(Hashing): 키를 해시 함수를 이용해
배열의 인덱스로 변환시켜 데이터를 배열에 저장하는 방식
ex) 물건의 바코드화

해시 함수 : 키를 인덱스(N >=0)로 변환시키는 함수
ex) 바코드화 하는 규칙

해시함수의 시간복잡도

해시함수의 시간복잡도는 O(1)
정말정말~~ 드물게 O(N)
그렇기 때문에, 탐색속도가 정말~정말~ 빠르다
O(N)이 나오는 경우는 **해시충돌이 처음부터 끝까지 이루어지는경우!**

해시탐색 실습!

Su su su Supernova

Nova

Can't stop hyperstellar

원초 그걸 찾아

Bring the light of a dying star

불러낸 내 우주를 봐 봐

해시충돌(Hash collision)

간단하게 요약하자면 같은 key에 여러 value가 부여된 상황을 의미합니다!
ex) 한바코드에 여러 값이 저장되어있는 경우

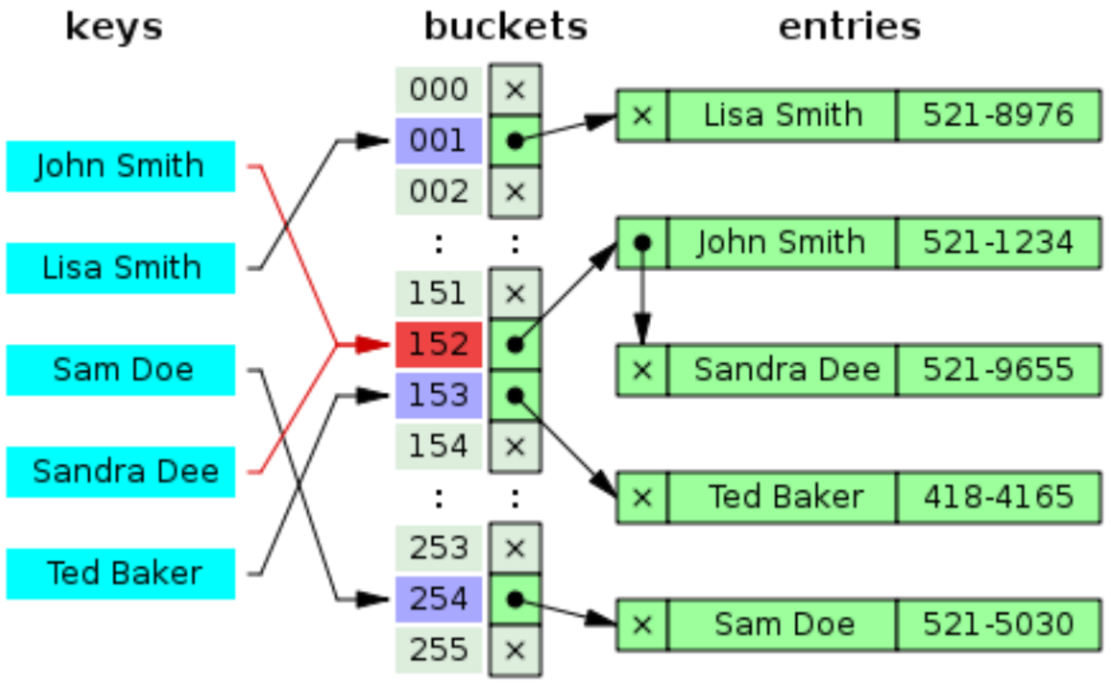
=>같은 key에 여러 value를 저장하는 것이 가능한 일인지?

가능은 하다만..

가능은 하지만 이런 해시 충돌이 계속해서 이루어지게 되면, 일정 키의 버켓(바코드)에 데이터가 집중되게 됩니다.
그리고 이렇게 해시충돌이 많이 일어나게 되면, 해시테이블의 성능을 떨어뜨립니다.
=> 따라서, 해시충돌을 해결하는것이 성능향상에 도움이 된다!

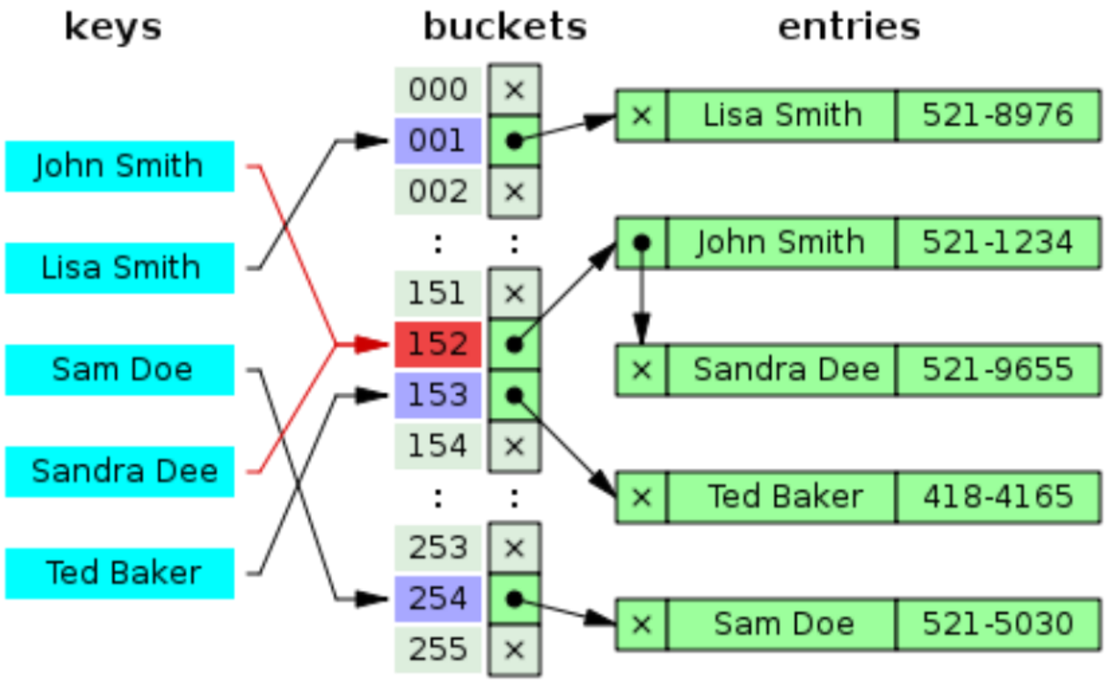
해시충돌 해결의 방법

체이닝

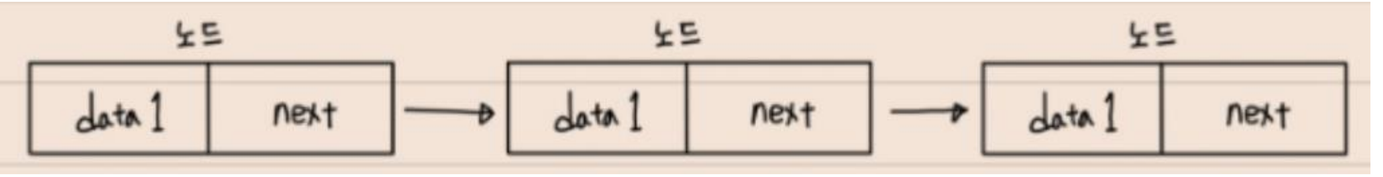


개방 주소법

체이닝



체이닝이란 해시충돌이 발생시 연결 리스트를 이용하여 데이터를 연결하는 방식입니다.



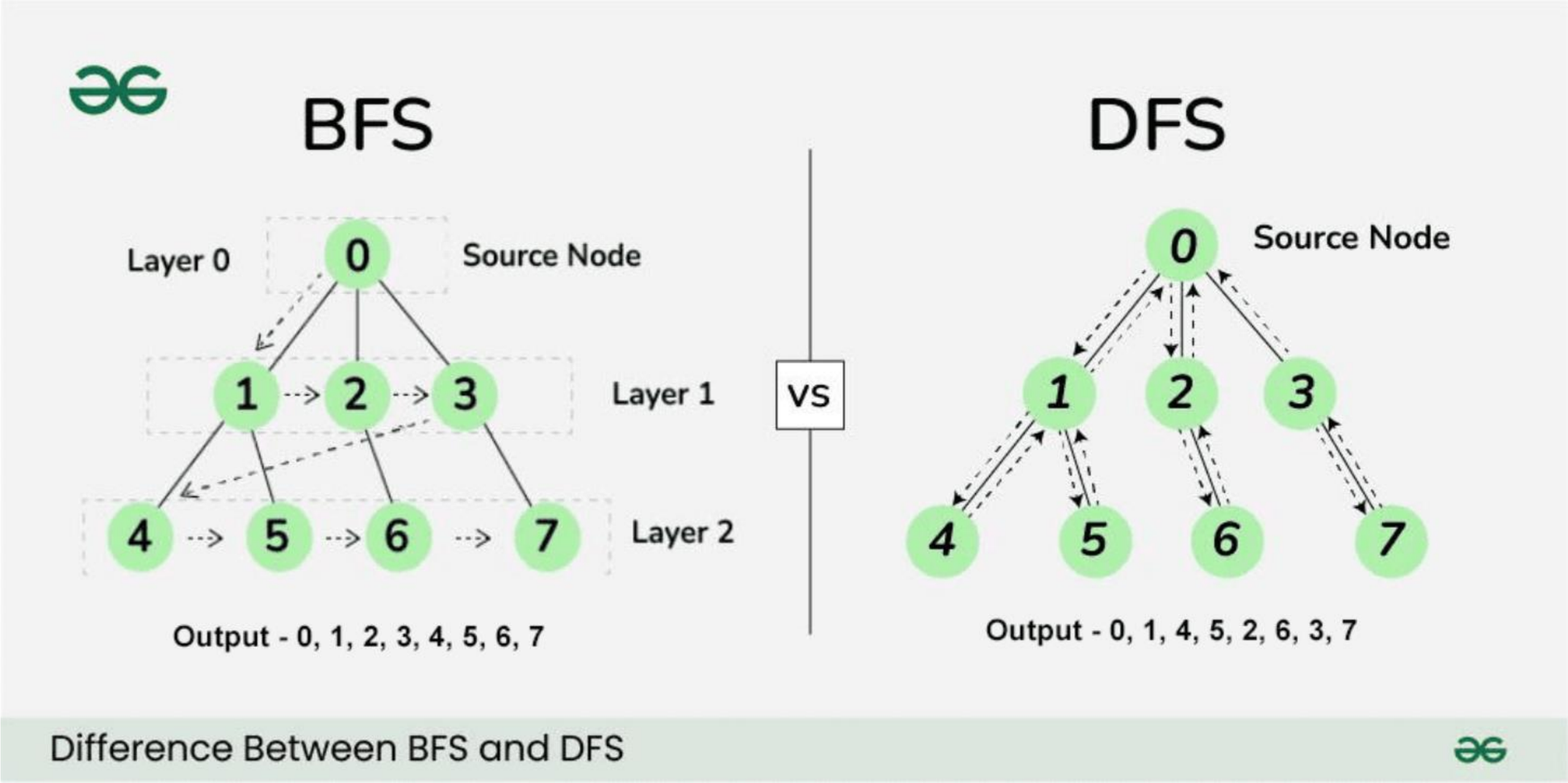
개방 주소법

해시 충돌이 일어나면 다른 버켓에 데이터를 삽입하는 방식을 개방 주소법이라고 합니다. 개방 주소법은 대표적으로 3가지가 있습니다.

선형 탐색(Linear Probing): 해시충돌 시 다음 버켓, 혹은 몇 개를 건너뛰어 데이터를 삽입한다.

제곱 탐색(Quadratic Probing): 해시충돌 시 제곱만큼 건너뛴 버켓에 데이터를 삽입(1,4,9,16..)

이중 해시(Double Hashing): 해시충돌 시 다른 해시함수를 한 번 더 적용한 결과를 이용함.


BFS
Layer 0
0
Source Node
1
2
3
Layer 1
4
5
6
7
Layer 2
Output - 0, 1, 2, 3, 4, 5, 6, 7
VS
DFS
0
Source Node
1
2
3
4
5
6
7
Output - 0, 1, 4, 5, 2, 6, 3, 7
Difference Between BFS and DFS

커밍 쑨

감사합니다