

이진탐색

순차탐색



• 실습: 다음 코드의 search 함수는 배열 v 내에 key 값의 존재하는 경우 그 위치 번호를, 그렇지 않은 경우 -1을 반환한다고 한다. 순차탐색 방식으로 search 함수를 완성하시오. 아래 search 함수의 시간복잡도는 얼마인가?

```
public class Test {
    public static void main(String[] args) {
        int v[]=new int[1000000];
        Random random=new Random();
        for (int i = 0; i < v.length; i++) v[i]=random.nextInt(1000000);
        int key=1234;
        System.out.println(search(v, key));
    }
    private static int search(int[] v, int key) {
    }
}</pre>
```

정렬된 배열에서의 이진탐색 (binary search)



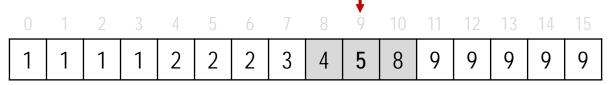
탐색 Key 값 = 8



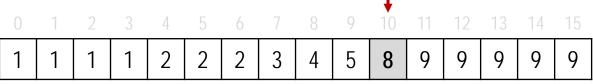
v[0..15]의 중간 위치 값 3을 key와 비교 → 3 < 8 → key는 3의 오른쪽 v[8..15]에 존재



v[8..15]의 중간 위치 값 8을 key와 비교 → 8 < 9 → key는 9의 왼쪽 v[8..10]에 존재



v[8..10]의 중간 위치 값 5를 key와 비교 → 5 < 8 → key는 5의 오른쪽 v[10..10]에 존재



v[10..10]의 중간 위치 값 7을 key와 비교 → 8 == 8 → key 값 발견 → key 값 8의 위치 = 10

정렬된 배열에서의 이진탐색 (binary search)



탐색 Key 값 = 6



v[10..10]의 중간 위치 값 7을 key와 비교 → 6 < 8 → key는 8의 왼쪽 v[10..9]에 존재 → v[10..9]는 부적절 범위 → 키 값 6은 존재하지 않음

정렬된 배열에서의 이진탐색 (binary search)



https://en.wikipedia.org/wiki/Binary_search_algorithm, CC-BY-SA

```
public class Test {
    public static void main(String[] args) {
                                                                      시간복잡도 O(\log n)
         int v[] = \{1,1,1,1,2,2,2,3,4,5,8,9,9,9,9,9,9\}
                                                                      배열 v의 크기가 백만이라면
         Arrays.sort(v); // 배열 정렬
                                                                      순차탐색과 이진탐색의 비교 횟수는?
         int key=7;
             index=binarySearch(v, key);
         if(index<0) System.out.println("배열 내 자료 없음");
         else System.out.println("자료 발견 위치(배열 내 index): "+index);
    private static int binarySearch(int[] v, int key) {
         int left=0, right=v.length-1;
         while(left<=right){</pre>
              int m=(left+right)/2;
                                                                중간 위치 계산
              if(key==v[m]) return m; 		←
                                                                key 발견 시 위치 반환
              if(key<v[m]) right=m-1; 		◀
                                                               key는 v[left..m-1]에 존재
              else left=m+1:
                                                                key는 v[m+1..right]에 존재
         return -1;
                                                                키 부재 시 -1 반환
         실습: 위 binarySearch 함수를 아래 형식의 재귀호출 함수로 재작성하시오
             호출문 → int index=binarySearch(v, key, 0, v.length-1);
             함수 정의문 → private static int binarySearch(int[] v, int key, int left, int right) { ... }
```

References



- ♣ C로 쓴 자료구조론 (Fundamentals of Data Structures in C, Horowitz et al.). 이석호 역. 사이텍미디어. 1993.
- ♣ 쉽게 배우는 알고리즘: 관계 중심의 사고법. 문병로. 한빛아카데 미. 2013.
- ♣ C언어로 쉽게 풀어 쓴 자료구조. 천인국 외 2인. 생능출판사. 2017.
- ♣ 프로그래밍 콘테스트 챌린징, Akiba 등 공저, 로드북, 2011.
- https://introcs.cs.princeton.edu/
- Introduction to Algorithms, Cormen et al., 3rd Edition (The MIT Press)