2.9 이진 탐색

개요

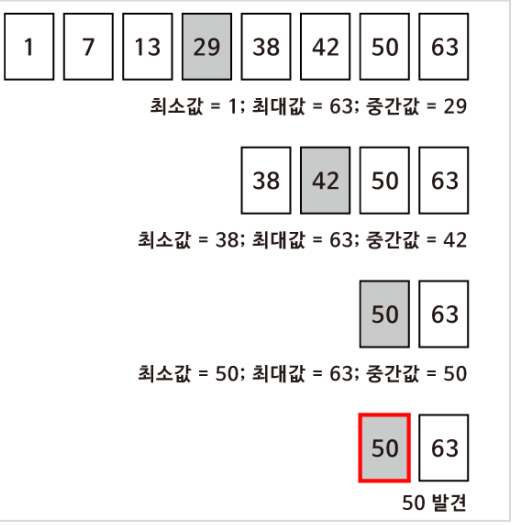
주어진 배열을 **탐색**하는데 사용할 수 있는 다양한 알고리즘이 있습니다. 그중 하나는 선형 탐색이라는 것인데 이것은 시간이 다소 오래 걸릴 수 있습니다. 만약 여러분이 찾고자 하는 원소가 배열의 끝에 있다고 가정해봅시다. 선형 알고리즘을 사용하면 찾고자 하는 원소를 찾기 위해 배열 전체를 탐색해야 합니다. 선형 탐색보다 더 빠른 알고리즘으로 **이진 탐색**이 있습니다. 이진 탐색은 **자료를 절반으로 나눈 후 찾는 값이 어느 쪽에 있는지 파악해 탐색의 범위를 반으로 줄여나가는 탐색 알고리즘**입니다.

핵심개념

* 이진 탐색
* 중간값

효율성과 비효율성

```
1  set minimum  = 0  and maximum = n - 1
2  find middle of array
3  if k is less than array[middle]
4      set maximum to middle - 1
5      go to line 2
6  else if k is greater than array[middle]
7      set minimum to middle + 1
8      go to line 2
9  else if k is equal to array[middle]
10     you found k in the array!
11 else
12     k is not in the array
```



▲ 〈코드 1〉

배열을 정렬하는 데는 버블 정렬, 삽입 정렬, 선택 정렬 등 다양한 방법이 있습니다. 이 **정렬 방법은 사실 이진 탐색을 구현하는 데 유용**합니다. 이진 탐색의 기본은 **정렬된 배열**을 만들 수 있다는데 가정을 두고 있습니다. 이진 탐색 의사 코드인 〈코드 1〉을 참고하여 배열에서 50을 찾아봅시다.

① 먼저 최소값을 배열의 첫 번째 값(1), 그리고 최대값을 배열의 마지막 값(63)으로 정합니다.
② 그리고 **중간값**을 계산합니다. 중간값은 네 번째 값(29)가 될 수도 있고, 다섯 번째 값(38)이 될 수도 있습니다. 어떤 값을 정하던 알고리즘은 일관적이기 때문에 상관은 없습니다.
③ 29를 중간값으로 사용했을 때, 50은 29보다 크기 때문에 왼편에 있는 값은 살펴볼 필요가 없습니다.
④ 최소값은 중간값 오른편에 있는 38로 정해지고, 최대값은 그대로 남겨둔 후 같은 방식으로 진행합니다.

이진 탐색 vs 선형 탐색

우리가 어떤 부분을 좋게 만들면 반드시 다른 부분에서 비용이 발생한다는 것은 컴퓨터 과학에서는 일반적인 사실입니다. 이진 탐색은 일반적으로 선형 탐색보다 탐색 속도가 짧지만, 배열을 정리하는 시간이 추가됩니다. 그렇기 때문에 배열의 값을 선형 탐색하는 것이 배열을 정렬한 후 이진 탐색을 사용하는 것보다 빠르게 보이기도 하고, 실제로 어떤 상황에서는 선형 탐색이 이진 탐색보다 빠르기도 합니다.
하지만 배열을 여러 번 탐색할 계획하고 있다면 이진 탐색이 유용하게 사용됩니다. 찾고자 하는 값이 배열에 없는 경우 선형 탐색은 배열의 길이가 얼마든 상관없이 배열 전체를 훑어보아야 배열 안에 찾고자 하는 값이 없다는 것을 알 수 있습니다. 이에 반해 이진 탐색에서는 더 효율적으로 사용할 수 있습니다. 의사 코드를 수행해보면, **찾고자 하는 값이 최소값보다 작거나 최대값보다 크다면 해당 원소가 배열 안에 없다는 것을 알 수 있습니다.**