

9 검색

1. 검색의 이해

❖ 검색search

- 컴퓨터에 저장한 자료 중에서 원하는 항목을 찾는 작업
 - 검색 성공 - 원하는 항목을 찾은 경우
 - 검색 실패 - 원하는 항목을 찾지 못한 경우
- 탐색 키를 가진 항목을 찾는 것
 - 탐색 키_{search key} - 자료를 구별하여 인식할 수 있는 키
- 삽입/삭제 작업에서의 검색
 - 원소를 삽입하거나 삭제할 위치를 찾기 위해서 검색 연산 수행



1. 검색의 이해

■ 검색 방법

- 수행 위치에 따른 분류
 - 내부 검색Internal Search : 메모리 내의 자료에 대해서 검색 수행
 - 외부 검색External Search : 보조 기억 장치에 있는 자료에 대해서 검색 수행
- 검색 방식에 따른 분류
 - 비교 검색 방식comparison search method
 - 계산 검색 방식non-comparison method

표 10-1 검색 방법에 따른 분류

검색 방법	설명	종류
비교 검색	검색 대상의 키를 비교하여 검색한다.	순차 검색, 이진 검색, 트리 검색
계산 검색	계수적인 성질을 이용한 계산으로 검색한다.	해싱



2. 순차 검색

❖ 순차 검색 sequential search, 선형 검색 linear search

- 일렬로 된 자료를 처음부터 마지막까지 순서대로 검색하는 방법
- 가장 간단하고 직접적인 검색 방법
- 배열이나 연결 리스트로 구현된 순차 자료 구조에서 원하는 항목을 찾는 방법
- 검색 대상 자료가 많은 경우에 비효율적이지만 알고리즘이 단순하여 구현이 용이함



2. 순차 검색

❖ 정렬되어 있지 않은 자료를 순차 검색

■ 검색 방법

- 첫 번째 원소부터 시작하여 마지막 원소까지 순서대로 키 값이 일치하는 원소가 있는지를 비교하여 찾는다.
 - 키 값이 일치하는 원소를 찾으면 그 원소가 몇 번째 원소인지를 반환
 - 마지막 원소까지 비교하여 키 값이 일치하는 원소가 없으면 찾은 원소가 없는 것이므로 검색 실패



2. 순차 검색

8	30	1	9	11	19	2
---	----	---	---	----	----	---

(a) 정렬되어 있지 않은 자료의 예

① $8 \neq 9$	8	30	1	9	11	19	2
② $30 \neq 9$	8	30	1	9	11	19	2
③ $1 \neq 9$	8	30	1	9	11	19	2
④ $9 = 9$	8	30	1	9	11	19	2

(b) 검색 성공의 예 : 9 검색

① $8 \neq 6$	8	30	1	9	11	19	2
② $30 \neq 6$	8	30	1	9	11	19	2
③ $1 \neq 6$	8	30	1	9	11	19	2
④ $9 \neq 6$	8	30	1	9	11	19	2
⑤ $11 \neq 6$	8	30	1	9	11	19	2
⑥ $19 \neq 6$	8	30	1	9	11	19	2
⑦ $2 \neq 6$	8	30	1	9	11	19	2

(c) 검색 실패의 예 : 6 검색

그림 10-1 정렬되어 있지 않은 자료에서의 순차 검색 예



2. 순차 검색

■ 정렬되지 않은 자료의 순차검색 알고리즘

알고리즘 10-1 정렬되지 않은 자료의 순차 검색

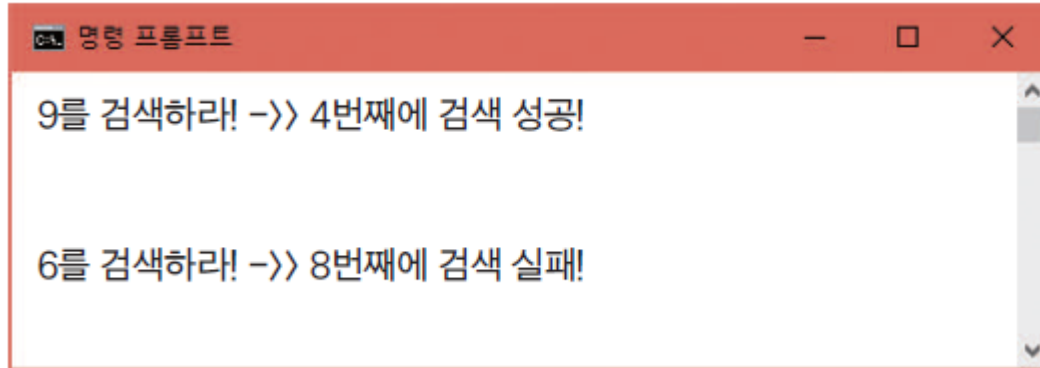
```
sequentialSearch1(a[], n, key)
  i ← 0;
  while (i < n and a[i] ≠ key) do {
    i ← i + 1;
  }
  if (i < n) then return i;
  else return -1;
end sequentialSearch1()
```

- 비교 횟수 - 찾고자 하는 원소의 위치에 따라 결정
 - 찾는 원소가 첫 번째 원소라면 비교 횟수는 1번, 두 번째 원소라면 비교 횟수는 2번, 세 번째 원소라면 비교 횟수는 3번, 찾는 원소가 i번째 원소이면 i번
- 평균 비교 횟수 : $\frac{1}{n} \sum_{i=1}^n i = \frac{1}{n} \times \frac{(nn+1)}{2} = \frac{n+1}{2}$
- 평균 시간 복잡도 : $O(n)$



2. 순차 검색

- 정렬되지 않은 자료를 순차 검색하기 프로그램 : [교재 560p](#)
- 실행 결과



```
명령 프롬프트

9를 검색하라! ->> 4번째에 검색 성공!

6를 검색하라! ->> 8번째에 검색 실패!
```



2. 순차 검색

❖ 정렬된 자료를 순차 검색

- 원소의 키값이 찾는 키값보다 크면 찾는 원소가 없는 것이므로 더 이상 검색을 수행하지 않아도 검색 실패를 알 수 있음

1	2	8	9	11	19	29
---	---	---	---	----	----	----

(a) 정렬된 자료의 예

① $1 < 9$	1	2	8	9	11	19	29
② $2 < 9$	1	2	8	9	11	19	29
③ $8 < 9$	1	2	8	9	11	19	29
④ $9 = 9$	1	2	8	9	11	19	29

(b) 검색 성공의 예 : 9 검색

① $1 < 6$	1	2	8	9	11	19	29
② $2 < 6$	1	2	8	9	11	19	29
③ $8 > 6$	1	2	8	9	11	19	29

→ 검색 종료

(c) 검색 실패의 예 : 6 검색

그림 10-2 정렬된 자료에서의 순차 검색 예



2. 순차 검색

■ 정렬된 자료의 순차검색

알고리즘 10-2 정렬된 자료의 순차 검색

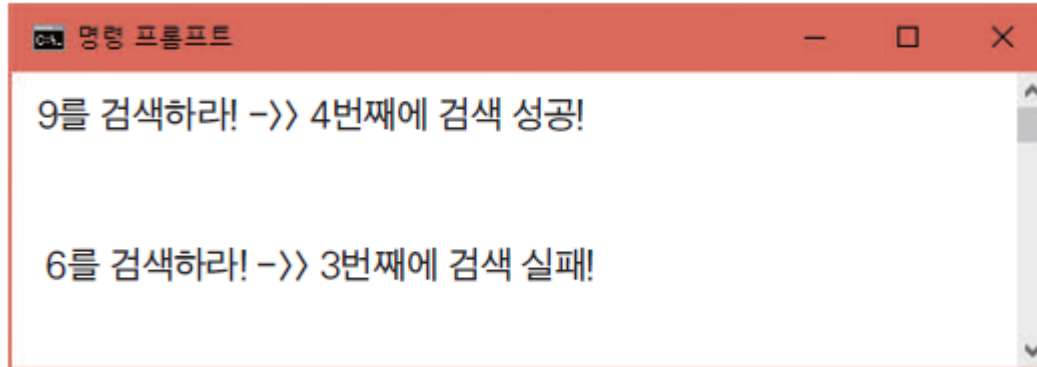
```
sequentialSearch2(a[], n, key)
  i ← 0;
  while (a[i] < key) do {
    i ← i + 1;
  }
  if (a[i] = key) then return i;
  else return -1;
end sequentialSearch2()
```

- 비교횟수 - 찾고자 하는 원소의 위치에 따라 결정
 - 검색 실패의 경우에 평균 비교 횟수가 반으로 줄어듦
- 평균 시간 복잡도 : $O(n)$



2. 순차 검색

- 정렬된 자료를 순차 검색하기 프로그램 : [교재 562p](#)
- 실행 결과



```
C:\> 명령 프롬프트

9를 검색하라! ->> 4번째에 검색 성공!

6를 검색하라! ->> 3번째에 검색 실패!
```



3. 이진 검색

❖ 이진 검색 binary search, 이분 검색, 보간 검색 interpolation search

- 자료의 가운데에 있는 항목을 키 값과 비교하여 다음 검색 위치를 결정하여 검색을 계속하는 방법
 - 찾는 키 값 > 원소의 키 값 : 오른쪽 부분에 대해서 검색 실행
 - 찾는 키 값 < 원소의 키 값 : 왼쪽 부분에 대해서 검색 실행
- 키를 찾을 때까지 이진 검색을 순환적으로 반복 수행함으로써 검색 범위를 반으로 줄여가면서 더 빠르게 검색
- 정복 기법을 이용한 검색 방법
 - 검색 범위를 반으로 분할하는 작업과 검색 작업을 반복 수행
- 정렬되어있는 자료에 대해서 수행하는 검색 방법

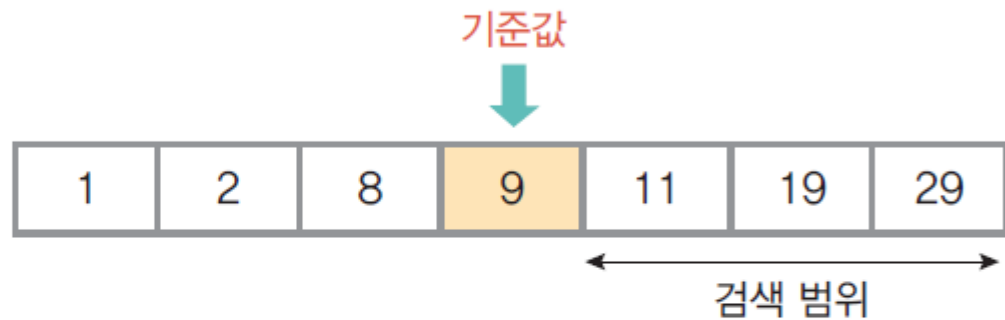


3. 이진 검색

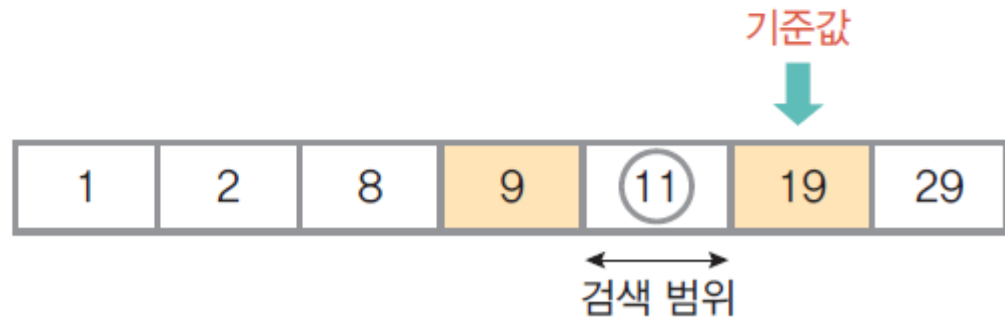
1	2	8	9	11	19	29
---	---	---	---	----	----	----

(a) 이진 검색 자료 예

① $11 > 9 \rightarrow$ 오른쪽 종료



② $11 < 19 \rightarrow$ 왼쪽 종료



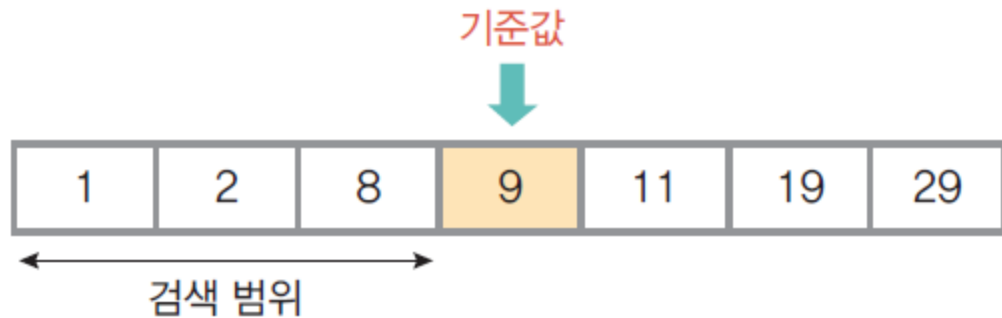
③ $11 = 11 \rightarrow$ 검색 성공

(b) 검색 성공의 예 : 11 검색

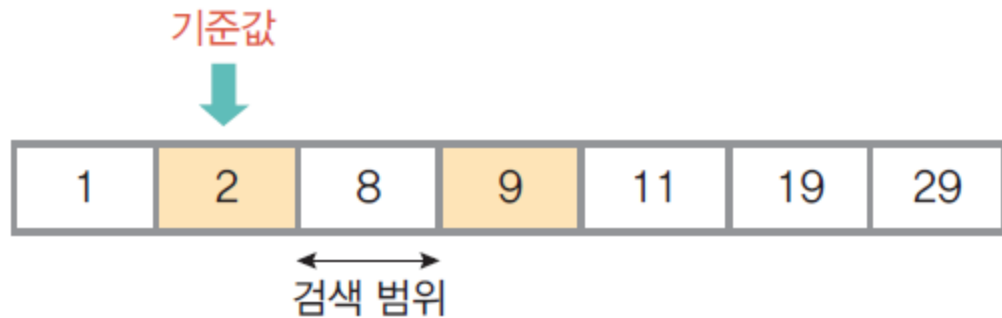


3. 이진 검색

① $6 < 9 \rightarrow$ 왼쪽 종료



② $6 > 2 \rightarrow$ 오른쪽 종료



③ $6 \neq 8 \rightarrow$ 검색 종료

(c) 검색 실패의 예 : 6 검색

그림 10-4 이진 검색의 예



3. 이진 검색

■ 이진 검색 알고리즘

알고리즘 10-5 이진 검색

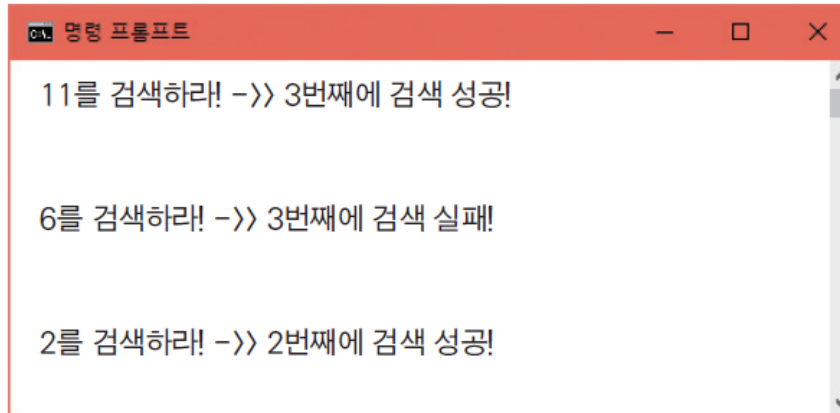
```
binarySearch(a[], begin, end, key)
    middle ← (begin + end) / 2;
    if (key = a[middle]) then return 1;
    else if (key < a[middle]) then binarySearch(a[], begin, middle - 1, key);
    else if (key > a[middle]) then binarySearch(a[], middle + 1, end, key);
    else return -1;
end binarySearch()
```

- n개의 자료에 대한 이진 검색의 메모리 사용량은 n이 됨.
- 이진 검색에서 검색 범위를 1/2로 분할하면서 비교하는 연산에 대한 시간 복잡도는 $O(\log_2 n)$



3. 이진 검색

- 정렬된 자료를 이진 검색하기 프로그램 : [교재 568p](#)
- 실행 결과



```
명령 프롬프트
11를 검색하라! ->> 3번째에 검색 성공!

6를 검색하라! ->> 3번째에 검색 실패!

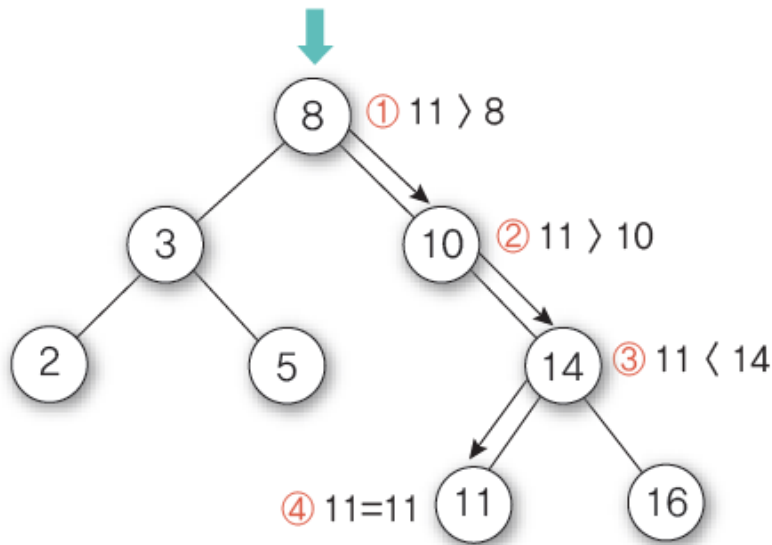
2를 검색하라! ->> 2번째에 검색 성공!
```



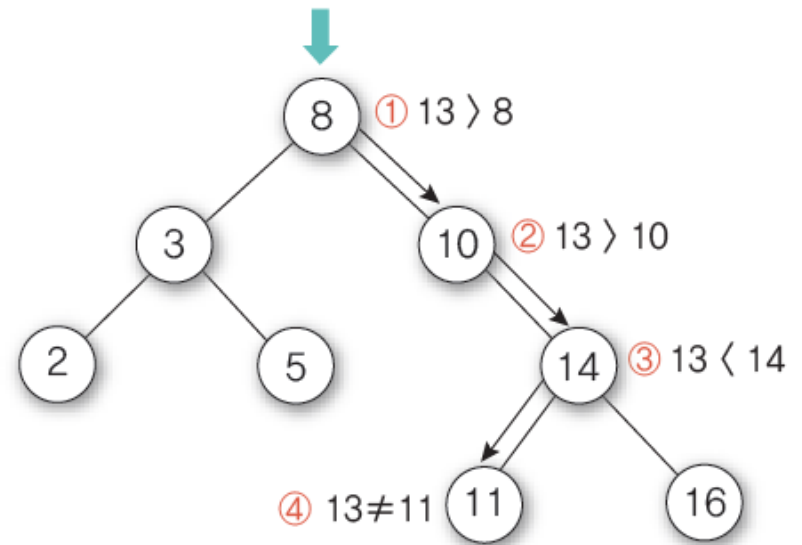
4. 이진 트리 검색

❖ 이진 트리 검색 binary tree search

- 7장에서 설명한 이진 탐색 트리를 사용한 검색 방법
- 원소의 삽입이나 삭제 연산에 대해서 항상 이진 탐색 트리를 재구성하는 작업 필요



(a) 검색 성공의 경우



(b) 검색 실패의 경우

그림 10-5 이진 트리 검색의 예: 11 검색과 13 검색



4. 이진 트리 검색

- 이진 검색을 이용해 영어 사전 구현하기 프로그램 : [교재 570p](#)
- 실행 결과

```
영단어 프로그램
-----
1 : 출력
2 : 입력
3 : 삭제
4 : 검색
5 : 종료

-----
2

[단어 입력] 단어를 입력하시오 : come

[단어 입력] 단어의 뜻을 입력하시오 : 오다, 나오다, 도착하다.

-----
1 : 출력
2 : 입력
3 : 삭제
4 : 검색
5 : 종료

-----
2

[단어 입력] 단어를 입력하시오 : active

[단어 입력] 단어의 뜻을 입력하시오 : 활동적인, 의욕적인

-----
1 : 출력
2 : 입력
3 : 삭제
4 : 검색
5 : 종료

-----
2

[단어 입력] 단어를 입력하시오 : boy

[단어 입력] 단어의 뜻을 입력하시오 : 소년

-----
1 : 출력
2 : 입력
3 : 삭제
4 : 검색
5 : 종료

-----
2

[단어 입력] 단어를 입력하시오 : passion

[단어 입력] 단어의 뜻을 입력하시오 : 열정, 정열
```

```
영단어 프로그램
-----
1 : 출력
2 : 입력
3 : 삭제
4 : 검색
5 : 종료

-----
2

[단어 입력] 단어를 입력하시오 : boy

[단어 입력] 단어의 뜻을 입력하시오 : 소년

이미 같은 단어가 있습니다!

-----
1 : 출력
2 : 입력
3 : 삭제
4 : 검색
5 : 종료

-----
1
[사전 출력]
active : 활동적인, 의욕적인
boy : 소년
come : 오다, 나오다, 도착하다.
passion : 열정, 정열
[사전 끝]

-----
1 : 출력
2 : 입력
3 : 삭제
4 : 검색
5 : 종료

-----
5
```





Thank You

