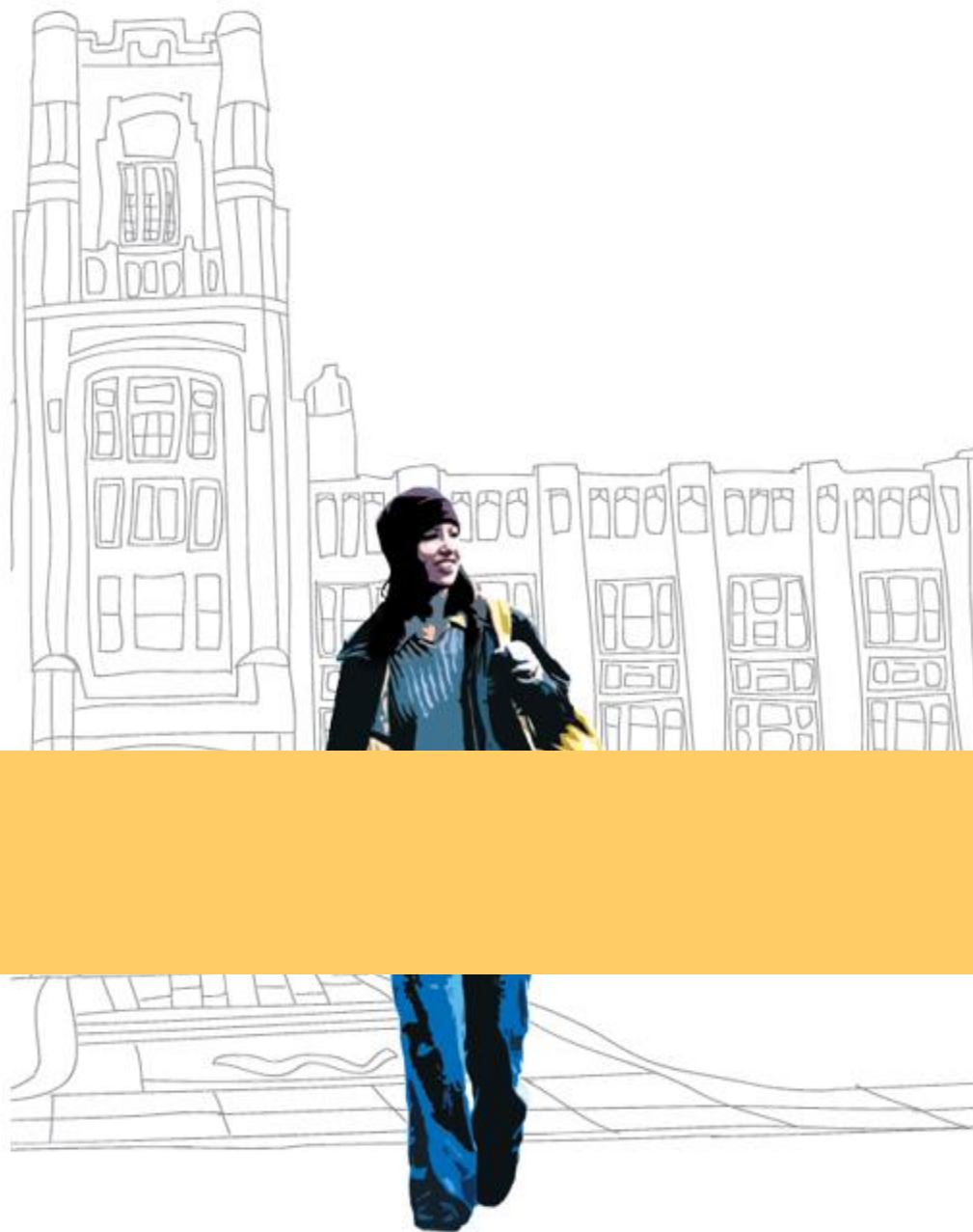


12

검색



이 장에서 다룰 내용

1

검색

2

순차 검색

3

이진 검색

❖ 검색(search)

- **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

❖ 검색 방법

■ 수행 위치에 따른 분류

- 내부 검색 - 메모리 내의 자료에 대해서 검색 수행
- 외부 검색 - 보조 기억 장치에 있는 자료에 대해서 검색 수행

■ 검색 방식에 따른 분류

- 비교 검색 방식(comparison search method)
 - 검색 대상의 키를 비교하여 검색하는 방법
 - 순차 검색, 이진 검색, 트리 검색
- 계산 검색 방식(non-comparison method)
 - 계수적인 성질을 이용한 계산으로 검색 하는 방법
 - 해싱

■ 검색 방법의 선택

- 자료 구조의 형태와 자료의 배열 상태에 따라 최적의 검색 방법 선택

□ 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

□ 순차 검색

❖ 정렬되지 않은 순차자료구조에서의 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

■ 순차 검색 예) 검색 성공의 경우

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

9를 검색하는 경우

① $9 \neq 8$

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

② $9 \neq 30$

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

③ $9 \neq 1$

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

④ $9 = 9$

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

■ 순차 검색 예) 검색 실패의 경우

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

6을 검색하는 경우

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

(b) 검색 실패의 경우

□ 순차 검색

■ 정렬되어있지 않은 자료에 대한 순차검색 알고리즘

```
sequentialSearch1(a[],n, key)
```

[알고리즘 12-1]

```
  i ← 0;
```

```
  while (i < n and a[i] ≠ key) do {
```

```
    i ← i + 1;
```

```
  }
```

```
  if (i < n) then return i;
```

```
  else return -1;
```

```
end sequentialSearch1()
```

- 비교횟수 - 찾고자 하는 원소의 위치에 따라 결정

➤ 찾는 원소가 첫 번째 원소라면 비교횟수는 1번, 두 번째 원소라면 비교횟수는 2번, 세 번째 원소라면 비교횟수는 3번, 찾는 원소가 i번째 원소이면 i번,

...

➤ 정렬되지 않은 원소에서의 순차 검색의 평균 비교 횟수
= $1/n(1+2+3+ \dots + n) = \underline{(n+1)/2}$

총합, 총선

$$\frac{n+1}{2}$$

- 평균 시간 복잡도 : $O(n)$

□ 순차 검색

❖ 정렬되어 있는 순차자료구조에서의 순차 검색

■ 검색 방법

- 순서대로 검색하면서 키 값을 비교하여, 원소의 키 값이 찾는 키 값보다 크면 찾는 원소가 없는 것이므로 더 이상 검색을 수행하지 않고 검색종료
- 정렬되어있는 자료에 대한 순차 검색 예)

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

9를 검색하는 경우

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

(a) 검색 성공의 경우

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

6을 검색하는 경우

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

→ 검색 종료

(b) 검색 실패의 경우

■ 정렬되어있는 자료에 대한 순차검색 알고리즘

```
sequentialSearch2(a[],n, key)
```

[알고리즘 12-2]

```
  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 줄어든다.
 - 정렬되어있는 원소에서의 순차 검색의 평균 비교 횟수
$$= 1/n(1+2+3+ \dots + n) \times 1/2 = (n+1)/4$$
- 평균 시간 복잡도 : $O(n)$

□ 순차 검색

❖ 순차 검색 프로그램

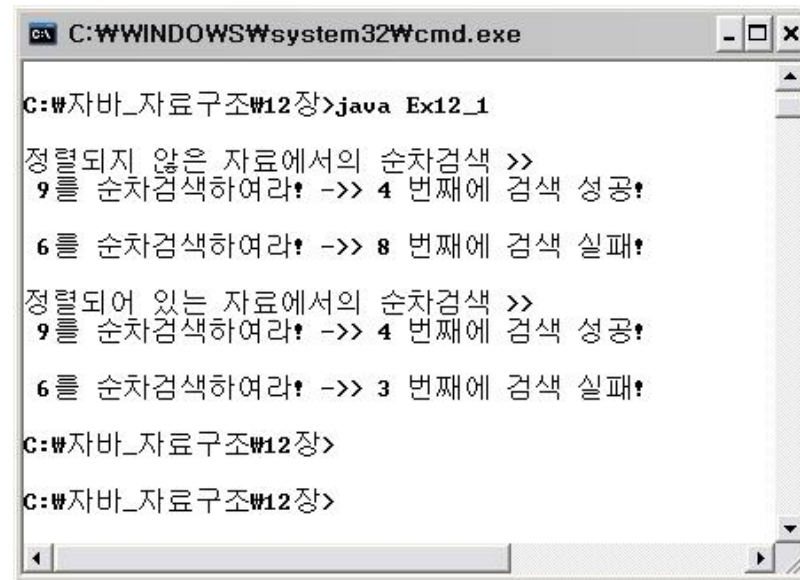
[예제 12-1]

```
01 class Search{
02     public void sequentialSearch1(int a[], int size, int key){
03         int i=0;
04         System.out.printf("\n %d를 순차검색하여라! ->> ", key);
05         while(i<size && (a[i]!=key)) i++;
06         if(i<size)
07             System.out.printf("%d 번째에 검색 성공! \n", i+1);
08         else
09             System.out.printf("%d 번째에 검색 실패! \n", i+1);
10     }
11
12     public void sequentialSearch2(int a[], int size, int key){
13         int i=0;
14         System.out.printf("\n %d를 순차검색하여라! ->> ", key);
15         while(a[i] < key) i++;
16         if(a[i] == key)
17             System.out.printf("%d 번째에 검색 성공! \n", i+1);
```

>> 계속

```
18         else
19             System.out.printf("%d 번째에 검색 실패! \n", i+1);
20     }
21 }
22
23 class Ex12_1{
24     public static void main(String args[]){
25         int a1[] = {8, 30, 1, 9, 11, 19, 2};
26         int size = a1.length;
27         Search S = new Search();
28         System.out.printf("\n정렬되지 않은 자료에서의 순차검색 >> ");
29         S.sequentialSearch1(a1, size, 9);
30         S.sequentialSearch1(a1, size, 6);
31
32         int a2[] = {1, 2, 8, 9, 11, 19, 29};
33         size = a2.length;
34         System.out.printf("\n정렬되어 있는 자료에서의 순차검색 >> ");
35         S.sequentialSearch2(a2, size, 9);
36         S.sequentialSearch2(a2, size, 6);
37     }
38 }
```

■ 실행 결과



```
C:\WINDOWS\system32\cmd.exe

C:\자바_자료구조\12장>java Ex12_1

정렬되지 않은 자료에서의 순차검색 >>
9를 순차검색하여라! ->> 4 번째에 검색 성공!

6를 순차검색하여라! ->> 8 번째에 검색 실패!

정렬되어 있는 자료에서의 순차검색 >>
9를 순차검색하여라! ->> 4 번째에 검색 성공!

6를 순차검색하여라! ->> 3 번째에 검색 실패!

C:\자바_자료구조\12장>
C:\자바_자료구조\12장>
```

□ 이진 검색

❖ 이진 검색(binary search)

- 이분 검색, 보간 검색(interpolation search)이라고도 함
- 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

$\log_2 n$

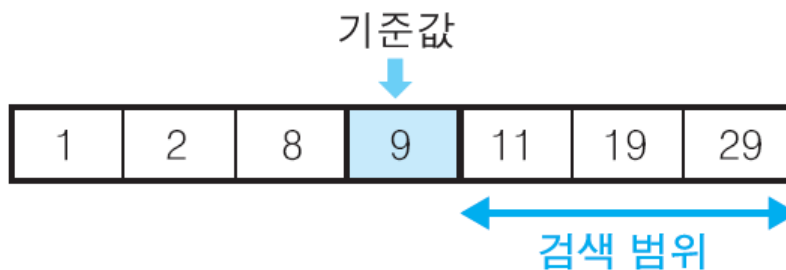
이진 검색

이진 검색의 예

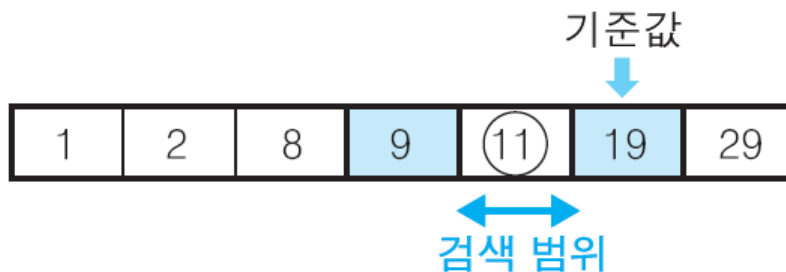
1	2	8	9	11	19	29
---	---	---	---	----	----	----

11을 검색하는 경우

① $11 > 9 \rightarrow$ 오른쪽 검색



② $11 < 19 \rightarrow$ 왼쪽 검색



③ $11 = 11 \rightarrow$ 검색 성공

(a) 검색 성공의 경우

이진 검색

이진 검색의 예

6을 검색하는 경우

① $6 < 9 \rightarrow$ 왼쪽 검색

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

기준값



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



검색 범위

기준값



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



검색 범위

② $6 > 2 \rightarrow$ 오른쪽 검색

③ $6 \neq 8 \rightarrow$ 검색 종료

(b) 검색 실패의 경우

❖ 이진 검색 알고리즘

```
binarySearch(a[], low, high, key)
```

[알고리즘 12-3]

```
    middle ← (low+high)/2;
```

```
    if (key = a[middle]) then return i;
```

```
    else if (key < a[middle]) then binarySearch(a[], low, middle-1, key);
```

```
    else if (key > a[middle]) then binarySearch(a[], middle+1, high, key);
```

```
    else return -1;
```

```
end binarySearch()
```

- 삽입이나 삭제가 발생했을 경우에 항상 배열의 상태를 정렬 상태로 유지하는 추가적인 작업 필요
- 시간 복잡도 : $O(\log_2 n)$

A woman wearing a dark jacket and a beanie is walking from left to right. She is carrying a yellow bag. In the background is a line-art sketch of a building with many windows. The scene is set against a white background with a vertical grey hatched bar on the left side. A thick orange horizontal band is positioned across the middle of the image, containing the text "Thank You !".

Thank You !