CHAPTER.25

# 검색의 기본 및 순차 검색 알고리즘의 원리와 구현

# 학습 내용

[1] 검색의 기본

[2] 순차 검색 알고리즘의 원리와 구현

# 학습 목표

- 검색의 개념을 설명할 수 있다.
- 검색을 위한 코드 형식에 대해 설명할 수 있다.
- 검색의 다양한 알고리즘에 대해 설명할 수 있다.

# 01 검색의 기본

1) 검색이란?
2) 검색의 개념
3) 검색 알고리즘의 종류

# 1] 검색이란?

세종사이버대학교

**‘검색’은 정렬된 상태에서 빠르게 원하는 것을 찾을 수 있음**

- 뒤죽박죽 섞여 있는 단어 퍼즐에서는 단어 찾는 데 오랜 시간이 걸림

# 1] 검색이란?

세종사이버대학교

**‘검색’은 정렬된 상태에서 빠르게 원하는 것을 찾을 수 있음**

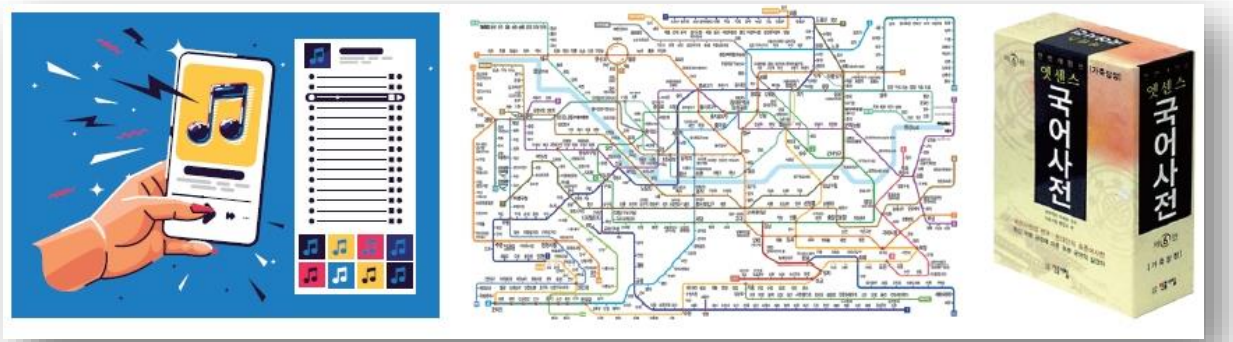
- 알파벳 순서로 되어 있는 퍼즐에서는 빠르고 쉽게 단어를 찾을 수 있음

# 2] 검색의 개념



**어떤 집합에서 원하는 것을 찾는 것으로, '탐색'이라고도 함**

- **검색의 다양한 예**



# 2] 검색의 개념



- 검색에는 순차 검색, 이진 검색, 트리 검색 등이 있음
- 검색에 실패하면 -1을 반환하는 것이 일반적

# 3] 검색 알고리즘의 종류



## 1 순차 검색

- 검색할 집합이 정렬되어 있지 않은 상태일 때
- 처음부터 차례대로 찾아보는 것으로, 쉽지만 비효율적임
- 집합의 데이터가 정렬되어 있지 않다면
  이 검색 외에 특별한 방법 없음

# 3] 검색 알고리즘의 종류



## 2 이진 검색

- 데이터가 정렬되어 있다면 이진 검색도 사용 가능
- 순차 검색에 비해 월등히 효율적이라
  데이터가 몇 천만 개 이상이어도 빠르게 찾아낼 수 있음

## 3 트리 검색

- 데이터 검색에는 상당히 효율적이지만
  트리의 삽입, 삭제 등에는 부담

02

# 순차 검색 알고리즘의 원리와 구현

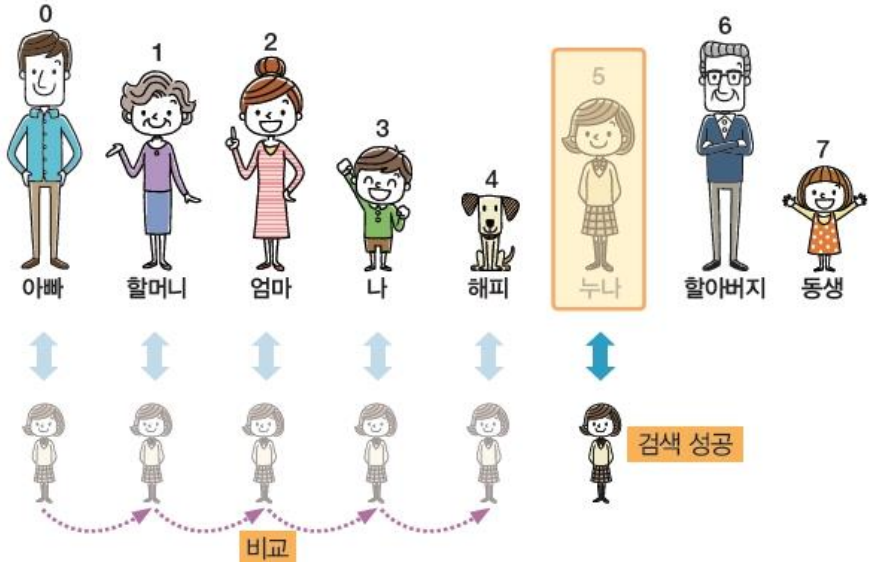
## 1) 순차 검색

## 정렬되지 않은 집합의 순차 검색 원리와 구현

- 키 순으로 정렬되지 않은 집합

## 정렬되지 않은 집합의 순차 검색 원리와 구현

- 검색에 성공하는 경우



## 정렬되지 않은 집합의 순차 검색 원리와 구현

- 검색에 실패하는 경우

## 정렬되지 않은 데이터의 순차 검색

```
1  ## 클래스와 함수 선언 부분 ##
2  def seqSearch(ary, fData) :
3      pos = -1
4      size = len(ary)
5      print('## 비교한 데이터 ==> ', end = ' ')
6      for i in range(size) :
7          print(ary[i], end = ' ')
8          if ary[i] == fData :
9              pos = i
10             break
11     print()
12     return pos
13
```

## 정렬되지 않은 데이터의 순차 검색

```
14 ## 전역 변수 선언 부분 ##
15 dataAry = [188, 150, 168, 162, 105, 120, 177, 50]
16 findData = 0
17
18 ## 메인 코드 부분 ##
19 findData = int(input('* 찾을 값을 입력하세요. --> '))
20 print('배열 -->', dataAry)
21 position = seqSearch(dataAry, findData)
22 if position == -1 :
23     print(findData, '(이)가 없네요.')
24 else :
25     print(findData,' (은)는 ', position, '위치에 있음')
```

## 정렬되지 않은 데이터의 순차 검색

실행 결과

```
* 찾을 값을 입력하세요. --> 105
배열 --> [188, 150, 168, 162, 105, 120, 177, 50]
## 비교한 데이터 ==> 188 150 168 162 105
105 (은)는  4 위치에 있음

* 찾을 값을 입력하세요. --> 100
배열 --> [188, 150, 168, 162, 105, 120, 177, 50]
## 비교한 데이터 ==> 188 150 168 162 105 120 177 50
100 (이)가 없네요.
```

## 순차 검색 실습

앞쪽의 소스를 수정해서 중복된 데이터가 여러 개 있을 때, 위치 여러 개를 만들어서 반환하도록 코드를 변경하자. 코드 중 dataAry와 findData는 다음과 같이 변경한다.

datAry = [188, 50, 150, 168, 50, 162, 105, 120, 177, 50]

findData = 50

**실행 결과**

```
배열 --> [188, 50, 150, 168, 50, 162, 105, 120, 177, 50]
50 (은)는  [1, 4, 9] 위치에 있음
```

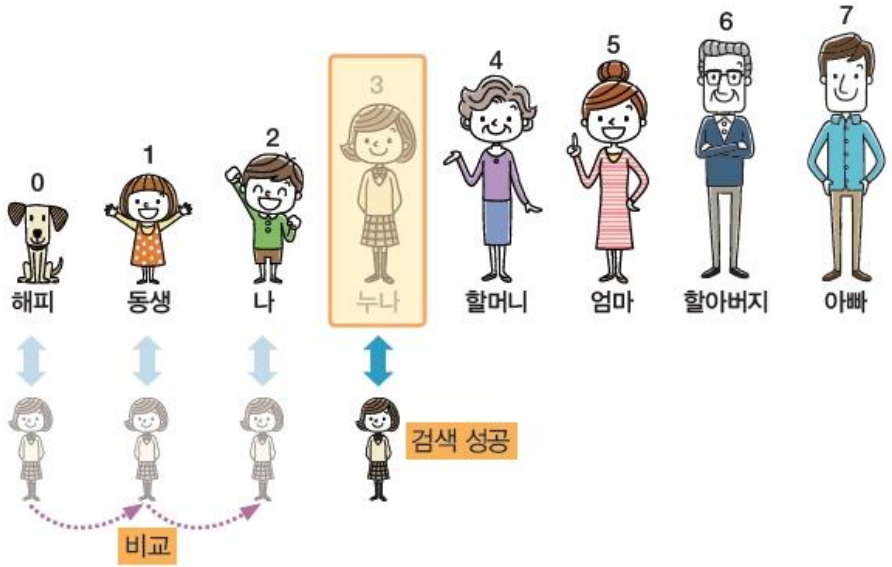
## 순차 검색 실습

## 정렬된 집합의 순차 검색 원리와 구현

- 키 순으로 정렬된 집합

## 정렬된 집합의 순차 검색 원리와 구현

- 검색에 성공하는 경우



## 정렬된 집합의 순차 검색 원리와 구현

- 검색에 실패하는 경우

## 정렬된 데이터의 순차 검색

```
1  ## 클래스와 함수 선언 부분 ##
2  def seqSearch(ary, fData) :
3      pos = -1
4      size = len(ary)
5      print('## 비교한 데이터 ==> ', end = ' ')
6      for i in range(size) :
7          print(ary[i], end = ' ')
8          if ary[i] == fData :
9              pos = i
10             break
11         elif ary[i] > fData :
12             break
```

## 정렬된 데이터의 순차 검색

```
13      print()
14      return pos
15
16 ## 전역 변수 선언 부분 ##
17 dataAry = [188, 150, 168, 162, 105, 120, 177, 50]
18 findData = 0
19
20 ## 메인 코드 부분 ##
21 dataAry.sort()
22 findData = int(input('* 찾을 값을 입력하세요. --> '))
23 print('배열 -->', dataAry)
24 position = seqSearch(dataAry, findData)
25 if position == -1 :
26     print(findData, '(이)가 없네요.')
```

## 정렬된 데이터의 순차 검색

```
27 else :
28     print(findData, '(은)는 ', position, '위치에 있음')
```

실행 결과

```
* 찾을 값을 입력하세요. --> 150
배열 --> [50, 105, 120, 150, 162, 168, 177, 188]
## 비교한 데이터 ==> 50 105 120 150
150 (은)는  3 위치에 있음

* 찾을 값을 입력하세요. --> 70
배열 --> [50, 105, 120, 150, 162, 168, 177, 188]
## 비교한 데이터 ==> 50 105
70 (이)가 없네요.
```

## 순차 검색의 시간 복잡도

- 정렬되지 않은 집합의 순차 검색은 데이터 개수가 $n$개라면 시간 복잡도는 $O(n)$으로 표현됨
- 정렬된 집합의 순차 검색은 검색할 데이터가 없는 경우에도 데이터의 크기가 작다면 앞쪽만 검색한 후 검색 실패를 효율적으로 확인할 수 있음(시간 복잡도는 $O(n)$으로 표현)

# 학습평가

Q1 | Q2 | Q3 | Q4

## Q1

다음 중 검색의 종류가 아닌 것은?

1. 순차 검색
2. 이진 검색
3. 트리 검색
4. 스택 검색

# 학습평가

Q1 | Q2 | Q3 | Q4

## Q1 다음 중 검색의 종류가 아닌 것은?

1 순차 검색
2 이진 검색
3 트리 검색
4 스택 검색

정 답 4 스택 검색

해 설 검색에는 순차 검색, 이진 검색, 트리 검색이 있습니다.

# 학습평가



## Q2 정렬되지 않은 집합을 검색하는 방식은?

1. 순차 검색
2. 이진 검색
3. 트리 검색
4. 스택 검색

# 학습평가

Q1 | Q2 | Q3 | Q4

## Q2 정렬되지 않은 집합을 검색하는 방식은?

1. 순차 검색
2. 이진 검색
3. 트리 검색
4. 스택 검색

**정 답** 1 순차 검색

**해 설** 순차 검색은 검색할 집합이 정렬되어 있지 않은 상태일 때 사용합니다.

# 학습평가

Q1 Q2 Q3 Q4

## Q3 검색에 실패할 경우 일반적으로 반환하는 값은?

1. -2
2. -1
3. 0
4. 1

# 학습평가

Q1 Q2 Q3 Q4

## Q3 검색에 실패할 경우 일반적으로 반환하는 값은?

1 -2
2 -1
3 0
4 1

정 답 2 -1

해 설 검색에 실패할 경우 일반적으로 -1을 반환합니다.

## 검색의 기본

### 검색의 개념

- 어떤 집합에서 원하는 것을 찾는 것으로, 탐색이라고도 함
- 검색에는 순차 검색, 이진 검색, 트리 검색 등이 있음
- 검색에 실패하면 -1을 반환하는 것이 일반적임

## 검색의 기본

### 검색 알고리즘의 종류

- 순차 검색
  - ✓ 검색할 집합이 정렬되어 있지 않은 상태일 때
  - ✓ 처음부터 차례대로 찾아보는 것으로, 쉽지만 비효율적임
  - ✓ 집합의 데이터가 정렬되어 있지 않다면 이 검색 외에 특별한 방법 없음
- 이진 검색
  - ✓ 데이터가 정렬되어 있다면 이진 검색도 사용 가능
  - ✓ 순차 검색에 비해 월등히 효율적이라 데이터가 몇 천만 개 이상이어도 빠르게 찾아낼 수 있음

## 검색의 기본

### 검색 알고리즘의 종류

- 트리 검색
  - ✓ 데이터 검색에는 상당히 효율적이지만 트리의 삽입, 삭제 등에는 부담

## 순차 검색 알고리즘의 원리와 구현

### 순차 검색의 시간 복잡도

- 정렬되지 않은 집합의 순차 검색은 데이터 개수가 $n$개라면 시간 복잡도는 $O(n)$으로 표현됨