CHAPTER.21

정렬의 기본과 기본 정렬 알고리즘의 원리와 구현

학습 내용

[1] 정렬의 기본

[2] 기본 정렬 알고리즘의 원리와 구현 (선택 정렬)

학습 목표

- 정렬의 개념을 설명할 수 있다.
- 정렬을 위한 코드의 형식을 설명할 수 있다.
- 기본적인 정렬 방식에 대해 설명할 수 있다.

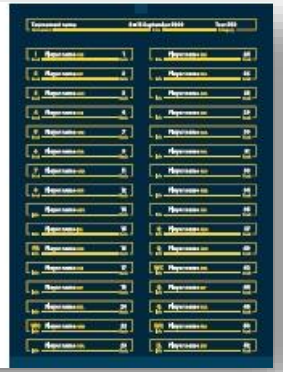
01 정렬의 기본

1) 정렬이란?
2) 정렬의 개념
3) 정렬 알고리즘의 종류

1] 정렬이란?

학교 출석부 또는 종류에 따라 가지런히 놓여 있는 칼들처럼 순서대로 데이터가 나열되어 있는 것

- 원하는 자리에 자유롭게 앉을 수 있는 대학교도 출석부에는 학생의 학번 순서 또는 이름 순서로 학생 명단이 작성되어 있음



1] 정렬이란?

학교 출석부 또는 종류에 따라 가지런히 놓여 있는 칼들처럼 순서대로 데이터가 나열되어 있는 것

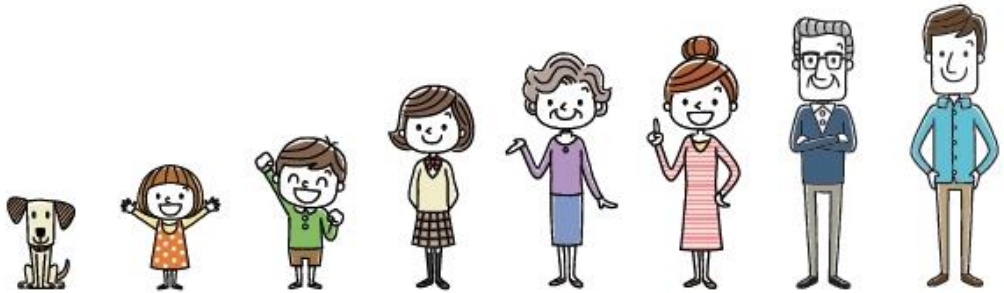
- 가지런히 놓고 사용하면 더 편리한 칼

정렬 (Sort)

중요 알고리즘 중 하나로, 자료들을 일정한 순서대로 나열한 것

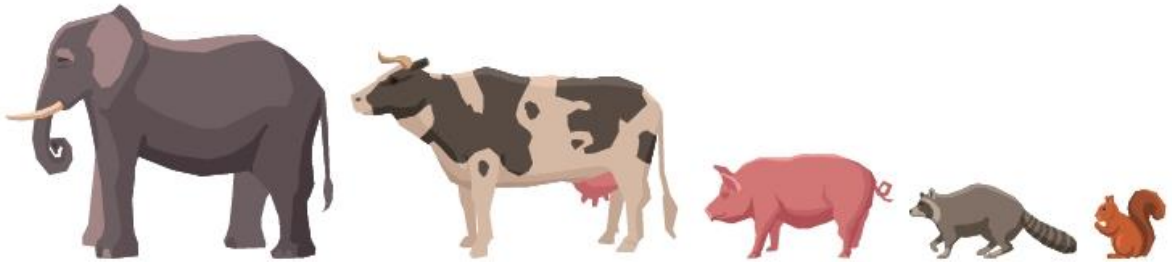
오름차순 정렬과 내림차순 정렬의 예

- 작은 키에서 큰 키 순으로 오름차순 정렬

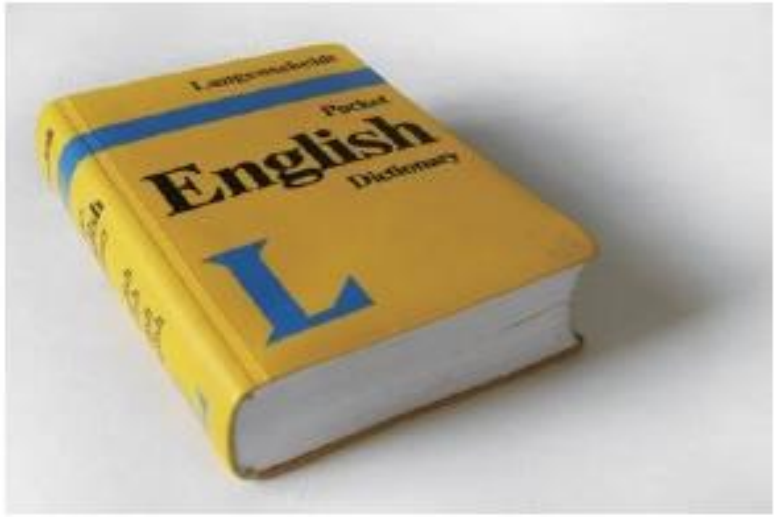


오름차순 정렬과 내림차순 정렬의 예

- 무거운 순에서 가벼운 순으로 내림차순 정렬



실생활 정렬 예



3] 정렬 알고리즘의 종류

오름차순 정렬이든 내림차순 정렬이든
결과의 형태만 다를 뿐이지 같은 방식으로 처리됨

정렬하는 방법에 대한 정렬 알고리즘은 수십 가지

- 선택 정렬 (Selection Sort)
- 삽입 정렬 (Insertion Sort)
- 버블 정렬 (Bubble Sort)
- 퀵 정렬 (Quick Sort)

02

기본 정렬 알고리즘의 원리와 구현

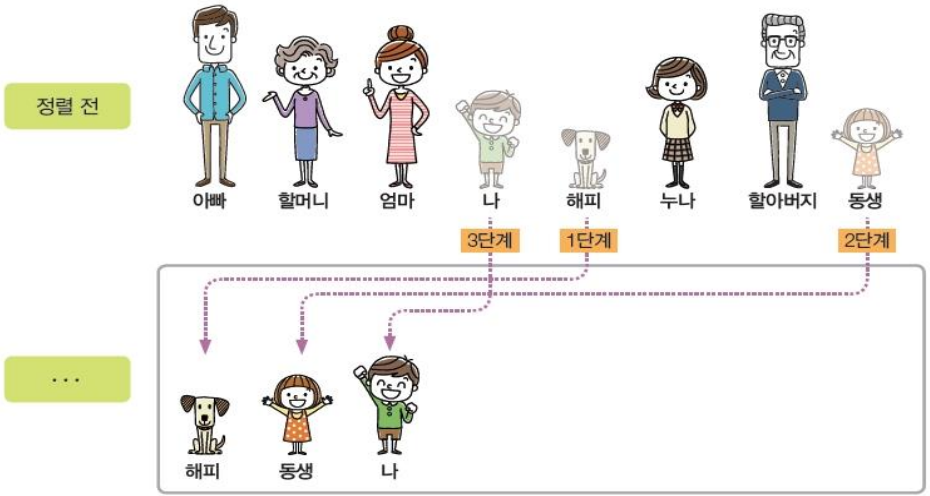
1) 선택 정렬

1] 선택 정렬

선택 정렬의 개념

여러 데이터 중에서 가장 작은 값을 뽑는 작동을 반복하여 값을 정렬

- 가족을 선택 정렬 방법으로 키 순(오름차순)으로 세우는 과정 예



선택 정렬의 개념

여러 데이터 중에서 가장 작은 값을 뽑는 작동을 반복하여 값을 정렬

- 가족을 선택 정렬 방법으로 키 순(오름차순)으로 세우는 과정 예

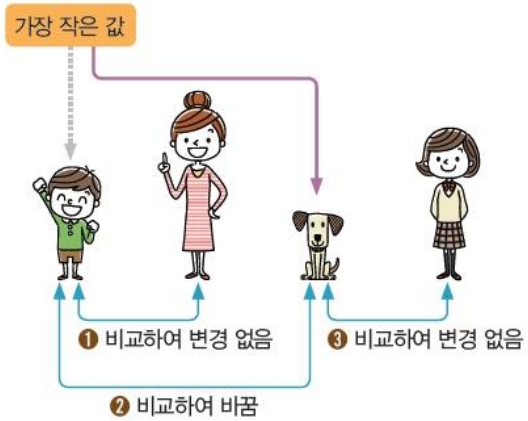
최솟값을 찾는 방법

1 배열의 첫 번째 값을 가장 작은 값으로 지정

최솟값을 찾는 방법

2 가장 작은 값으로 지정한 값을 다음 차례의 값과 비교하여 가장 작은 값을 변경하거나 그대로 두는 방법으로

3 마지막 값까지 비교를 마친 후 현재 가장 작은 값으로 지정된 값을 가장 작은 값으로 결정



최솟값을 찾는 방법

배열에서 최솟값 위치를 찾는 함수

```
1  def findMinIdx(ary) :
2      minIdx = 0 ❶
3      for i in range(1, len(ary)) :
4          if (ary[minIdx] > ary[i]) :
5              minIdx = i
6      return minIdx
7
8  testAry = [55, 88, 33, 77]
9  minPos = findMinIdx(testAry)
10 print('최솟값 -->', testAry[minPos])
```

❷

실행 결과

```
최솟값 --> 33
```

최댓값을 찾는 방법 실습

> 앞쪽의 소스를 수정해서 최댓값 위치를 찾도록 코드를 작성하자.

실행 결과

```
최댓값 --> 88
```

최댓값을 찾는 방법 실습

선택 정렬 구현

크기가 3인 배열을 준비하고 값을 입력하는 방법을 사용했음

```
ary = [None for _ in range(3)]
ary[0] = 100
ary[1] = 200
ary[2] = 300
```

배열 크기를 미리 지정하지 않고 파이썬에서 제공하는 가변 크기의 배열을 사용

```
ary = []
ary.append(100)
ary.append(200)
ary.append(300)
```

선택 정렬 구현

선택 정렬의 구현

```
1  ## 클래스와 함수 선언 부분 ##
2  def findMinIdx(ary) :
3      minIdx = 0
4      for i in range(1, len(ary)) :
5          if (ary[minIdx] > ary[i]) :
6              minIdx = i
7      return minIdx
8
9  ## 전역 변수 선언 부분 ##
10 before = [188, 162, 168, 120, 50, 150, 177, 105]
11 after = []
12
```

선택 정렬 구현

선택 정렬의 구현

```
13 ## 메인 코드 부분 ##
14 print('정렬 전 -->', before)
15 for _ in range(len(before)) :
16     minPos = findMinIdx(before)
17     after.append(before[minPos])
18     del(before[minPos])
19 print('정렬 후 -->', after)
```

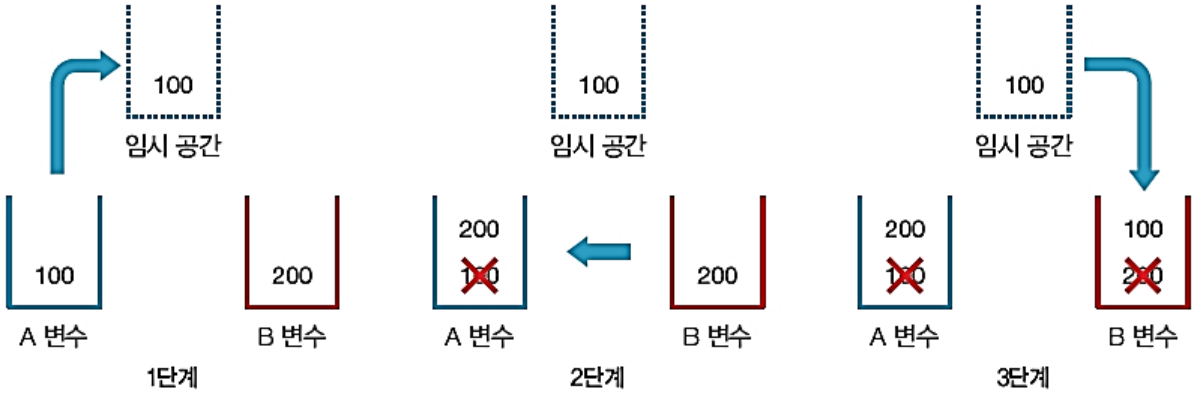
실행 결과

```
정렬 전 --> [188, 162, 168, 120, 50, 150, 177, 105]
정렬 후 --> [50, 105, 120, 150, 162, 168, 177, 188]
```

두 변숫값 교환

알고리즘을 구현할 때는 두 변숫값을 교환해야 하는 경우가 종종 발생

- 원칙적으로 한 번에 두 변수의 값을 교환할 수 없으므로, 임시 공간을 사용해야 함



두 변숫값 교환

알고리즘을 구현할 때는 두 변숫값을 교환해야 하는 경우가 종종 발생

- 원칙적으로 한 번에 두 변수의 값을 교환할 수 없으므로, 임시 공간을 사용해야 함

```
temp = A
A = B
B = temp
```

또는

```
A, B = B, A
```

개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

데이터가 4개이므로 오른쪽과 같이 총 3회의 사이클이 필요

- **선택 정렬 전 초기 상태**

선택 정렬할 데이터

3회 사이클

개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

1 먼저 사이클1 중 맨 앞의 값을 가장 작은 값으로 지정한 후, 나머지 값과 비교해서 제일 작은 값을 찾음

- 선택 정렬 예 – 사이클1

초기설정
(i 값은 0)

개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

1 먼저 사이클1 중 맨 앞의 값을 가장 작은 값으로 지정한 후, 나머지 값과 비교해서 제일 작은 값을 찾음

- 선택 정렬 예 – 사이클1

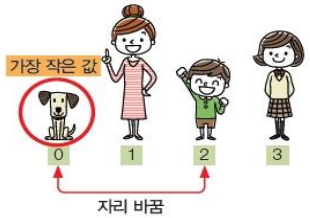
반복
(i+1부터 끝까지 가장작은값과비교)

개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

1 먼저 사이클1 중 맨 앞의 값을 가장 작은 값으로 지정한 후, 나머지 값과 비교해서 제일 작은 값을 찾음

- 선택 정렬 예 – 사이클1

마무리
(가장작은값이
i 위치로



개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

2 사이클1에서 찾은 가장 작은 값을 제외한 사이클2 중
맨 앞의 값을 가장 작은 값으로 우선 지정하고,
나머지 값들과 비교하여 제일 작은 값을 찾음

- 선택 정렬 예 – 사이클2

초기설정
(i 값은 1)

개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

2 사이클1에서 찾은 가장 작은 값을 제외한 사이클2 중
맨 앞의 값을 가장 작은 값으로 우선 지정하고,
나머지 값들과 비교하여 제일 작은 값을 찾음

- 선택 정렬 예 - 사이클2

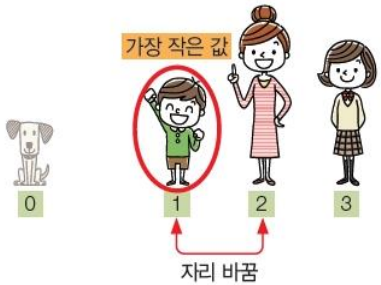
반복
(i+1부터 끝까지
가장작은값과비교)

개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

2 사이클1에서 찾은 가장 작은 값을 제외한 사이클2 중
맨 앞의 값을 가장 작은 값으로 우선 지정하고,
나머지 값들과 비교하여 제일 작은 값을 찾음

- 선택 정렬 예 – 사이클2

마무리
(가장작은값이
i 위치로



개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

3 사이클2에서 찾은 가장 작은 값을 제외한 사이클3 중
맨 앞의 값을 가장 작은 값으로 우선 지정하고,
나머지 값들과 비교하여 제일 작은 값을 찾음

- 선택 정렬 예 – 사이클3

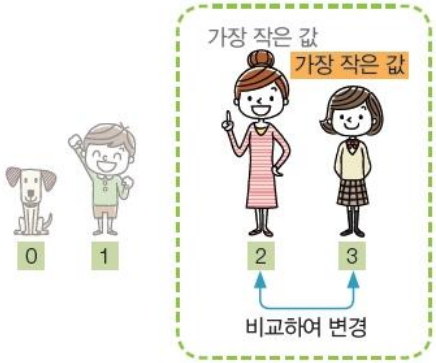
초기설정
(i 값은 2)

개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

3 사이클2에서 찾은 가장 작은 값을 제외한 사이클3 중
맨 앞의 값을 가장 작은 값으로 우선 지정하고,
나머지 값들과 비교하여 제일 작은 값을 찾음

- 선택 정렬 예 – 사이클3

반복
(i+1부터 끝까지
가장작은값과비교)



개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

3 사이클2에서 찾은 가장 작은 값을 제외한 사이클3 중
맨 앞의 값을 가장 작은 값으로 우선 지정하고,
나머지 값들과 비교하여 제일 작은 값을 찾음

- 선택 정렬 예 – 사이클3

마무리
(가장작은값이
i 위치로

개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

4 모든 사이클 완료 후 정렬된 결과

- 선택 정렬 후 데이터

개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

개선된 선택 정렬

```
## 클래스와 함수 선언 부분 ##
def selectionSort(ary) :
    n = len(ary)
    for i in range(0, n-1) :
        minIdx = i
        for k in range(i+1, n) :
            if (ary[minIdx] > ary[k]) :
                minIdx = k
        tmp = ary[i]
        ary[i] = ary[minIdx]
        ary[minIdx] = tmp

    return ary

```

개선된 선택 정렬 구현(데이터 4개를 정렬하는 예)

개선된 선택 정렬

```
15 ## 전역 변수 선언 부분 ##
16 dataAry = [188, 162, 168, 120, 50, 150, 177, 105]
17
18 ## 메인 코드 부분 ##
19 print('정렬 전 -->', dataAry)
20 dataAry = selectionSort(dataAry)
21 print('정렬 후 -->', dataAry)
```

실행 결과

```
정렬 전 --> [188, 162, 168, 120, 50, 150, 177, 105]
정렬 후 --> [50, 105, 120, 150, 162, 168, 177, 188]
```

선택 정렬 성능

정렬에서 중요한 사항 중 하나는 정렬을 완료하는 비교 횟수

Code11-03.py의 7행이 몇 번 수행되었는지 확인하는 예

i 값	k 값	비교 횟수
0	1, 2, 3	3회
1	2, 3	2회
2	3	1회

선택 정렬 성능

Code11-03.py의 7행이 몇 번 수행되었는지 확인하는 예

- 데이터 개수가 4일 때 7행은 이와 같이 반복

```
i가 0일 때 → 3(=4-1)번 수행
i가 1일 때 → 2(=4-2)번 수행
i가 2일 때 → 1(=4-3)번 수행
```

선택 정렬 성능

Code11-03.py의 7행이 몇 번 수행되었는지 확인하는 예

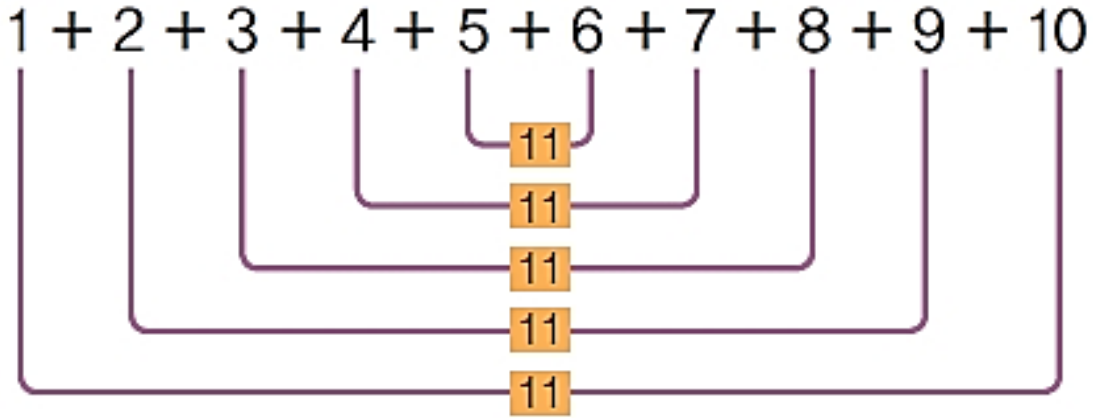
- 데이터 개수가 n개라면 이와 같이 계산됨

```
i가 0일 때 → n-1번 수행
i가 1일 때 → n-2번 수행
i가 2일 때 → n-3번 수행
i가 3일 때 → n-4번 수행
...(중략)...
i가 n-3일 때 → 2번 수행
i가 n-2일 때 → 1번 수행
```

선택 정렬 성능

비교 횟수는 거꾸로 하면 1+2+3+…+(n−1)번이 되는데
이를 수식으로 유도하기 전에 1부터 10까지 합계를 구하는 방법

- 1부터 10까지의 합계



선택 정렬 성능

비교 횟수는 거꾸로 하면 1+2+3+…+(n-1)번이 되는데
이를 수식으로 유도하기 전에 1부터 10까지 합계를 구하는 방법

11 * 5회 = 55

11이 5회 반복되므로 이와 같이 계산됨

(10 + 1) × (10 / 2) = 55

숫자 10을 중심으로 표현

선택 정렬 성능

결국 1부터 n까지 합계는 다음 수식과 같음
(10 대신에 n을 대입)

$$(n+1)\times\frac{n}{2}=\frac{(n+1)\times n}{2}$$

비교 횟수의 합계인 1+2+3+…+(n−1)은 1부터 n−1까지 합계이므로 위 수식에서 n 대신에 n−1을 대입

$$\frac{(n-1+1)\times(n-1)}{2}=\frac{n\times(n-1)}{2}=\frac{1}{2}n^2-\frac{1}{2}n$$

학습평가

Q1 Q2 Q3

Q1 정렬에 대한 기본 개념과 거리가 먼 것은?

1. 순서대로 나열하는 것을 의미한다.
2. 작은 것부터 나열하는 것을 오름차순 정렬이라고 한다.
3. 큰 것부터 나열하는 것을 내림차순 정렬이라고 한다.
4. 영어 사전이나 국어사전은 내림차순 정렬되어 있다.

학습평가

Q1 정렬에 대한 기본 개념과 거리가 먼 것은?

1. 순서대로 나열하는 것을 의미한다.
2. 작은 것부터 나열하는 것을 오름차순 정렬이라고 한다.
3. 큰 것부터 나열하는 것을 내림차순 정렬이라고 한다.
4. ✓ 영어 사전이나 국어사전은 내림차순 정렬되어 있다.

정 답 4 영어 사전이나 국어사전은 내림차순 정렬되어 있다.

해 설 사전은 일반적으로 오름차순으로 정렬되어 있습니다.

학습평가

Q1 Q2 Q3

Q2 정렬 알고리즘의 종류와 거리가 먼 것은?

1. 선택 정렬
2. 재귀 정렬
3. 삽입 정렬
4. 퀵 정렬

학습평가

Q1 Q2 Q3

Q2 정렬 알고리즘의 종류와 거리가 먼 것은?

1 선택 정렬
2 재귀 정렬
3 삽입 정렬
4 퀵 정렬

정 답 2 재귀 정렬

해 설 정렬은 선택 정렬, 삽입 정렬, 퀵 정렬이 있습니다.

학습평가

Q1 Q2 Q3

Q3 선택 정렬에 대한 설명과 거리가 먼 것은?

1. 데이터 중에서 가장 작은 값 또는 가장 큰 값을 뽑는 작동을 반복한다.
2. 키 순서, 이름 순서, 몸무게 순서 등을 정렬할 때 사용할 수 있다.
3. 정렬은 오름차순과 내림차순으로 정렬할 수 있다.
4. 개념은 어렵지만 속도가 가장 빠른 정렬 중 하나이다.

학습평가

Q3 선택 정렬에 대한 설명과 거리가 먼 것은?

1. 데이터 중에서 가장 작은 값 또는 가장 큰 값을 뽑는 작동을 반복한다.
2. 키 순서, 이름 순서, 몸무게 순서 등을 정렬할 때 사용할 수 있다.
3. 정렬은 오름차순과 내림차순으로 정렬할 수 있다.
4. ✓ 개념은 어렵지만 속도가 가장 빠른 정렬 중 하나이다.

정 답 4 개념은 어렵지만 속도가 가장 빠른 정렬 중 하나이다.

해 설 입력 개수가 커질수록 기하급수적으로 비교 횟수(또는 연산 횟수)가 늘어나기에 성능이 좋지 않은 알고리즘입니다.

정렬의 기본

정렬의 개념

- 중요 알고리즘 중 하나인 정렬(Soft)은 자료들을 일정한 순서대로 나열한 것

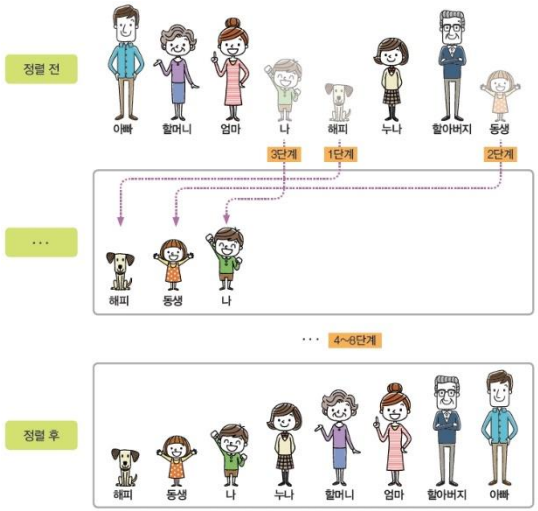
정렬 알고리즘의 종류

- 오름차순 정렬이든 내림차순 정렬이든 결과의 형태만 다를 뿐이지 같은 방식으로 처리됨
- 정렬하는 방법에 대한 정렬 알고리즘은 수십 가지
- 선택 정렬(Selection Sort), 삽입 정렬(Insertion Sort), 버블 정렬(Bubble Sort), 퀵 정렬(Quick Sort)

기본 정렬 알고리즘의 원리와 구현

선택 정렬

- 선택 정렬의 개념 : 여러 데이터 중에서 가장 작은 값을 뽑는 동작을 반복하여 값을 정렬



기본 정렬 알고리즘의 원리와 구현

선택 정렬

- **두 변숫값의 교환**
  - ✓ 알고리즘을 구현할 때는 두 변숫값을 교환해야 하는 경우가 종종 발생
  - ✓ 원칙적으로 한 번에 두 변수의 값을 교환할 수 없으므로, 임시 공간을 사용해야 함

```
temp = A
A = B
B = temp
```

또는

```
A, B = B, A
```