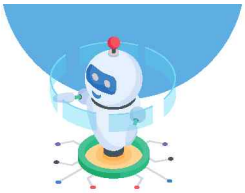


07 정렬과 탐색 알고리즘

✚ **학습 목표** •정렬과 탐색의 개념을 이해하고, 실생활의 문제를 해결할 수 있다.
•다양한 정렬과 탐색 알고리즘의 특징과 효율을 비교·분석할 수 있다.

✚ **학습 요소** •정렬 알고리즘, 탐색 알고리즘

생각 깨우기 **도서관 서가에서 책들을 순서대로 정리하는 이유를 생각해 보자.**

도서관에는 다양한 분야의 책을 한국 십진 분류법과 청구 기호에 따라 분류하고 정리하기 때문에 원하는 책을 쉽게 찾아서 꺼내 볼 수 있다.

Q **무작위로 섞여 있는 책을 순서대로 정리하고 원하는 책을 효율적으로 찾는 방법에는 무엇이 있을지 이야기해 보자.**

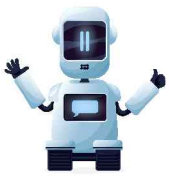
1 | 정렬 알고리즘

도서관에서는 여러 가지 기호를 사용하여 책을 순서대로 정리한다. 이처럼 여러 개의 데이터를 일정한 기준에 따라 순서대로 나열하는 것을 정렬이라고 하며, 데이터를 정렬하기 위한 구체적인 절차나 방법을 정렬 알고리즘이라고 한다. 값이 작은 기호부터 순서대로 정리된 도서관의 책, 작은 번호부터 정리된 시험지는 오름차순으로 정렬된 데이터이며, 메달이 많은 국가부터 적은 국가 순서대로 나열된 올림픽 순위는 내림차순으로 정렬된 데이터다. 데이터가 정렬되어 있다면 원하는 자료를 보다 쉽고 빠르게 찾을 수 있다. 정렬 알고리즘에는 선택 정렬, 퀵 정렬 등이 있다.

정렬 알고리즘

선택 정렬, 퀵 정렬 이외에도 버블 정렬, 삽입 정렬, 합병 정렬 등 다양한 종류의 알고리즘이 있다.

여기서는 오름차순으로 정렬하는 정렬 알고리즘을 설명합니다. 내림차순으로 정렬하기 위해서는 어떻게 하면 좋을지 스스로 생각해 보고, 친구와 의견을 나누어 보세요.



내림차순으로 선택 정렬할 때에는 가장 큰 데이터를 선택해 첫 번째 위치로 옮기고, 남은 데이터 중 가장 큰 데이터를 선택해 두 번째 위치로 옮기는 과정을 반복하며 정렬할 수 있다.

01 선택 정렬

선택 정렬은 정렬되지 않은 데이터 중에서 가장 작은 데이터를 선택하여 알맞은 위치로 옮기는 작업을 반복하며 전체 데이터를 순서대로 정렬하는 방법이다. 전체 데이터 중에서 가장 작은 데이터를 찾아 첫 번째 위치로 옮기고, 남은 데이터 중에서 가장 작은 데이터를 찾아 두 번째 위치로 옮기는 과정을 반복하며 모든 데이터를 정렬한다. 선택 정렬 알고리즘은 다음과 같다.

선택 정렬 알고리즘

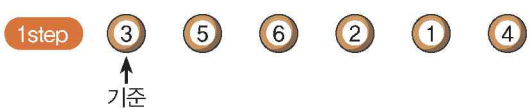
❶ 정렬되지 않은 데이터 중 첫 번째 위치에 있는 데이터를 기준으로 정한다.
❷ 가장 작은 데이터를 찾아 기준이 되는 데이터와 위치를 바꾼다.
❸ 알맞은 위치로 옮겨진 데이터를 제외한 나머지에 대해 ❶~❸을 반복한다.

예제 **다음과 같이 서로 다른 무게의 공 6개와 공의 무게를 비교할 수 있는 양팔 저울이 1개 있을 때, 가장 가벼운 공부터 순서대로 선택 정렬하는 모습을 살펴보자.**

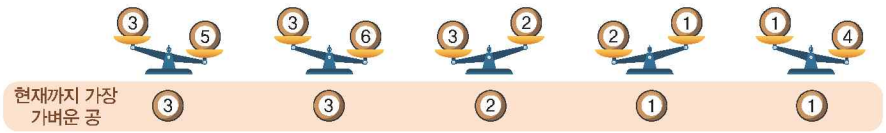
(공의 무게는 공에 적힌 번호와 같으며 양팔 저울을 통해서만 무게를 비교할 수 있다고 가정한다.)

풀이

❶ 정렬되지 않은 공 중 첫 번째 위치에 있는 3번 공을 기준으로 정한다.

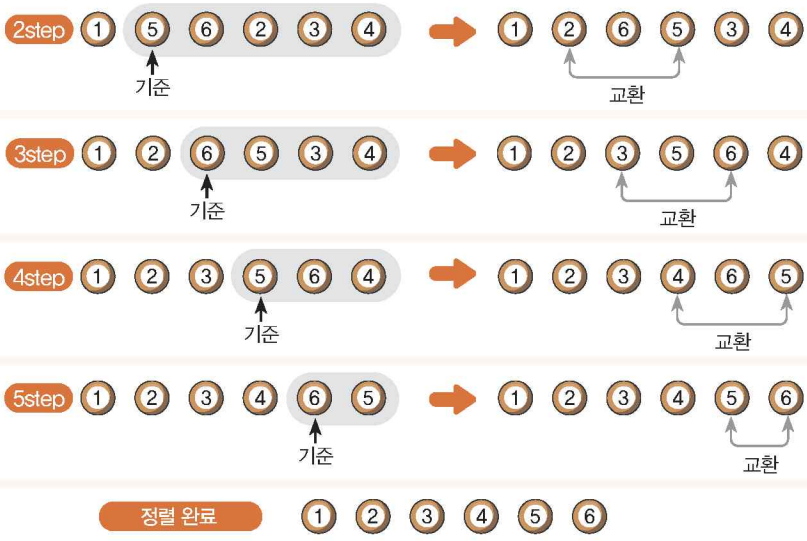


❷ 양팔 저울로 비교하며 정렬되지 않은 공 중 가장 가벼운 공을 찾는다.



❸ 양팔 저울을 다섯 번 사용하여 5회 비교하면 여섯 개의 공 중에서 가장 가벼운 공을 찾을 수 있다. 기준 공과 가장 가벼운 공의 위치를 교환한다.

❹ 정렬이 끝난 1번 공을 제외하고 정렬되지 않은 다섯 개의 공을 위와 같은 과정으로 반복하면 모든 공을 알맞은 위치로 옮겨 오름차순으로 정렬할 수 있다. 정렬되지 않은 공 중 가장 작은 공을 선택하여 알맞은 위치로 옮기며 선택 정렬하는 과정은 다음과 같다.



정렬 과정

구분	양팔 저울 사용 횟수 (비교 횟수)
1step	5
2step	4
3step	3
4step	2
5step	1
총 횟수	15

설명 선택 정렬 알고리즘에 따라 6개의 공을 정렬하기 위해서는 저울을 총 15회(5회+4회+3회+2회+1회) 사용해야 한다.

해 보기 1 선택 정렬 알고리즘으로 정렬하기

다음은 무작위로 섞여 있는 학급의 공책이다. 선택 정렬 알고리즘을 사용하여 번호를 기준으로 오름차순 정렬할 때 총 비교 횟수를 알아보자.

• 비교 횟수:

02 퀵 정렬

퀵 정렬은 기준이 되는 값을 정하여 기준보다 작은 데이터는 기준 값의 왼쪽에, 기준보다 큰 데이터는 기준 값의 오른쪽에 두어 그룹을 나누는 과정을 반복하면서 전체 데이터를 정렬하는 방법이다. 퀵 정렬 알고리즘은 다음과 같다.

내림차순으로 정렬하기 위해서는 어떻게 하면 좋을지 생각해 보고, 친구와 의견을 나누어 보세요.

퀵 정렬 알고리즘

❶ 첫 번째 위치에 있는 데이터를 기준으로 정한다.
❷ 기준보다 작은 데이터는 기준의 왼쪽 그룹에, 기준보다 큰 데이터는 기준의 오른쪽 그룹에 두어 두 개의 그룹으로 나눈다.
❸ 각 그룹 내의 데이터가 한 개가 될 때까지 ❶~❸을 반복한다.

예제 **다음과 같이 서로 다른 무게의 공 6개와 공의 무게를 비교할 수 있는 양팔 저울이 1개 있을 때, 가장 가벼운 공부터 순서대로 퀵 정렬하는 모습을 살펴보자.**

풀이

❶ 첫 번째 위치에 있는 공을 기준으로 정한다.

❷ 양팔 저울로 기준이 되는 3번 공과 비교하여 기준 공보다 가벼운 공의 그룹, 기준 공보다 무거운 공의 그룹으로 나눈다.

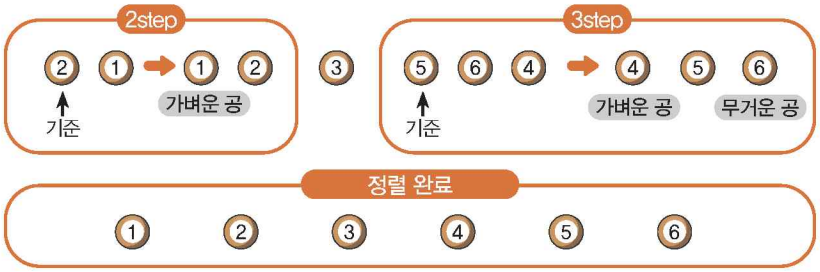
그룹의 첫 번째 데이터, 마지막 데이터, 중간에 위치한 데이터를 퀵 정렬의 기준 값으로 선택할 수 있어요. 이외에 또 어떤 방법이 있을지 생각해 보세요.

❸ 1단계에서 5단계까지 수행하는 동안 양팔 저울을 다섯 번 사용하여 3번 공을 기준으로 두 개의 그룹으로 나눌 수 있다.

정렬 과정

구분	양팔 저울 사용 횟수 (비교 횟수)
1step	5
2step	1
3step	2
총 횟수	8

④ 나누어진 두 개의 그룹에서 각각 기준 공을 정하여 두 개의 그룹으로 나누는 과정을 반복하면 모든 공을 알맞은 위치로 옮겨 오름차순으로 정렬할 수 있다. 각 그룹을 퀵 정렬하는 과정은 다음과 같다.



설명 퀵 정렬 알고리즘에 따라 6개의 공을 정렬하기 위해서는 저울을 총 8회(5회+1회+2회) 사용해야 한다.

해 보기 2 퀵 정렬 알고리즘으로 정렬하기

다음은 무작위로 섞여 있는 학급의 공책이다. 퀵 정렬 알고리즘을 사용하여 번호를 기준으로 오름차순 정렬할 때 총 비교 횟수를 알아보자.

• 비교 횟수:

03 선택 정렬과 퀵 정렬 알고리즘의 분석

선택 정렬은 알고리즘이 간단하지만 정렬할 데이터가 많아질수록 비교 횟수가 많아지므로 작은 규모의 데이터를 정렬할 경우 효과적으로 사용할 수 있다. 예를 들어 10개의 데이터를 선택 정렬한다면 총 45번을 비교해야 하지만, 1,000개의 데이터를 선택 정렬한다면 49만 9,500번을 비교해야 데이터를 정렬할 수 있다.

퀵 정렬은 가장 빠른 알고리즘 중 하나로 일반적으로 성능이 좋지만, 기준이 되는 값을 어떻게 선택하는지에 따라 효율성이 달라질 수 있다. 퀵 정렬의 기준으로 데이터의 가장 작은 값 또는 가장 큰 값이 선택되면, 왼쪽 또는 오른쪽 그룹에 데이터가 치우쳐 불균형하게 나누어지므로 비교 횟수가 많아져 효율성이 떨어진다. 반면, 데이터의 중앙값이 기준이 되면 비교 횟수가 크게 줄어들어 효율성이 높아진다.

Tip 퀵 정렬의 기준과 효율성
리스트 [7, 1, 2, 6, 3, 5, 4]를 퀵 정렬을 사용하여 오름차순으로 정렬할 때 첫 번째 요소를 기준으로 정할 경우의 비교 횟수는 총 21회(6+5+4+3+2+1)이지만, 중앙에 위치한 요소를 기준으로 정할 경우의 비교 횟수는 총 12회(6+4+1+1)가 된다. 따라서 위 예시 데이터의 경우 중앙에 위치한 요소를 기준으로 선택한 경우의 효율성이 더 좋다.

해 보기 3 선택 정렬의 비교 횟수 확인하기

다음은 133쪽의 선택 정렬 예제를 파이썬으로 구현한 코드다.

프로그램

```
# data 리스트에 정렬하고자 하는 값을 입력한다.
data = [3, 5, 6, 2, 1, 4]
def selection_sort(data):
  print('정렬 전 데이터', data)
  count = 0
  for i in range(len(data) - 1):
    min_idx = i
    for j in range(i + 1, len(data)):
      count = count + 1
      if data[j] < data[min_idx]:
        min_idx = j
    data[i], data[min_idx] = data[min_idx], data[i]
  print('정렬 후 데이터', data)
  print('전체 비교 횟수:', count)
selection_sort(data)
```

방법 ❶

1 프로그램을 직접 실행하여 비교 횟수를 확인해 보자.

실행 결과

2 data 리스트에 정렬하고자 하는 값을 다음과 같이 변경해 보고 비교 횟수를 확인해 보자.

```
data = [7, 1, 10, 9, 5, 6, 15, 90, 95, 62, 47, 32, 58, 49, 11, 70, 19]
```

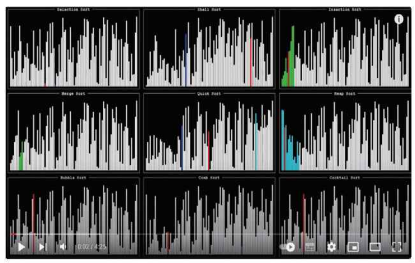
• 비교 횟수:

해 보기 4 정렬 알고리즘의 수행 시간 비교하기

영상을 보고 데이터 집합의 특징에 따라 선택 정렬과 퀵 정렬이 수행되는 시간이 어떻게 달라지는지 시간을 측정하여 기록해 보자.

★영상 주소 및 타임라인★

• 영상 주소 – https://youtu.be/ZZuD6iUe3Pc
• 특정한 몇 개의 값만 존재하는 데이터 – 1:08
• 거의 정렬이 완료된 데이터 – 3:38
• 랜덤한 데이터 – 0:0
• 역순으로 정렬된 데이터 – 2:07



데이터 집합의 특징	랜덤한 데이터	특정한 몇 개의 값만 존재하는 데이터	역순으로 정렬된 데이터	거의 정렬이 완료된 데이터
선택 정렬(Selection Sort)				
퀵 정렬(Quick Sort)				

2 | 탐색 알고리즘

여러 개의 데이터 중에서 원하는 것을 찾아내는 작업을 탐색이라고 하며, 데이터를 탐색하기 위한 구체적인 절차나 방법을 탐색 알고리즘이라고 한다. 탐색 알고리즘을 이용하면 도서관에서 원하는 책 찾기, 컴퓨터에 저장된 파일 찾기, 온라인 쇼핑몰에서 원하는 제품 찾기 등을 할 때 원하는 정보를 빠르고 효과적으로 찾을 수 있다. 대표적인 탐색 알고리즘에는 순차 탐색, 이진 탐색 등이 있다.

01 순차 탐색

순차 탐색이란 데이터의 처음부터 끝까지 순차적으로 하나씩 비교하며 원하는 데이터를 찾는 방법이다. 첫 번째 데이터부터 시작하여 원하는 데이터를 찾을 때까지 비교하며, 마지막 데이터까지 검사하고도 원하는 데이터를 찾지 못하면 찾고자 하는 데이터가 없다고 판단한다. 순차 탐색 알고리즘은 다음과 같다.

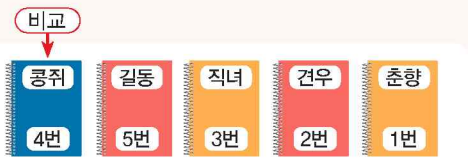
순차 탐색 알고리즘

1. 탐색하려는 데이터와 첫 번째 데이터를 비교한다.
2. 비교한 데이터가 탐색하려는 데이터와 같다면 탐색을 종료하고, 같지 않다면 다음 데이터와 비교한다.
3. 2를 반복하여 마지막 데이터까지 비교한 뒤에도 탐색하려는 데이터를 찾지 못했다면 데이터가 없다고 판단하고 탐색을 종료한다.

예제 **다음과 같이 섞여 있는 공책 중에서 2번 공책을 순차 탐색하는 과정을 알아보자.**

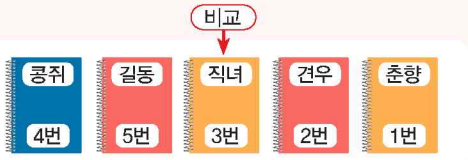
풀이

1. 첫 번째 공책이 2번 공책인지 비교한다. 2번 공책이 아니므로 다음 공책과 비교한다.



2. 두 번째 공책이 2번 공책인지 비교한다. 2번 공책이 아니므로 다음 공책과 비교한다.

3. 세 번째 공책이 2번 공책인지 비교한다. 2번 공책이 아니므로 다음 공책과 비교한다.

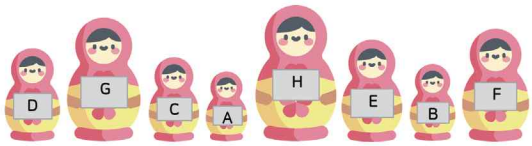


❹ 네 번째 공책이 2번 공책이다. 원하는 공책을 찾았으므로 탐색을 종료한다. 2번 공책은 네 번째 위치에 있다.

설명 순차 탐색 알고리즘에 따라 첫 번째 공책부터 순서대로 비교하며 2번 공책을 찾는다. 여기에서는 총 4회의 비교로 원하는 공책을 찾을 수 있다. 만약 2번 공책이 첫 번째 위치에 있었다면 한 번만 비교하여 찾을 수 있지만, 마지막 위치에 있었다면 총 5회 비교해야 찾을 수 있다.

해 보기 5 순차 탐색 알고리즘으로 탐색하기

다음 마트료시카 인형은 크기가 작은 것부터 A~H까지의 고유한 알파벳 이름을 갖는다. 순차 탐색 알고리즘을 사용하여 인형 'E'를 탐색할 때 총 비교 횟수를 알아보자.



• 비교 횟수:

02 이진 탐색

이진 탐색은 탐색할 데이터의 범위를 반으로 줄여 나가며 원하는 데이터를 찾는 알고리즘으로, 정렬되어 있는 데이터에서만 이진 탐색을 수행할 수 있다. 데이터가 정렬되어 있지 않다면, 데이터를 순서대로 정렬한 후 이진 탐색 알고리즘을 적용해야 한다. 이진 탐색은 순서대로 정렬된 데이터의 중간에 위치한 기준 데이터와 찾고자 하는 데이터를 비교하여 찾는 데이터가 기준 데이터보다 작다면 기준 데이터의 왼쪽 범위에서만 데이터를 탐색하고, 찾는 데이터가 기준 데이터보다 크다면 오른쪽 범위에서만 탐색하는 과정을 반복하면서 탐색 범위를 좁혀 가는 방법이다.

이진 탐색 알고리즘은 다음과 같다.

이진 탐색 알고리즘

❶ 탐색 범위에서 중간에 위치한 데이터를 기준으로 정한다.
❷ 기준 데이터와 탐색하려는 데이터를 비교하여 같다면 탐색을 종료하고, 같지 않다면 ㉮와 ㉯ 중 하나를 수행한다.
- ㉮ 탐색하려는 데이터가 중간에 위치한 기준 데이터보다 작다면, 탐색 범위를 기준 데이터의 왼쪽 그룹으로 좁힌다.
- ㉯ 탐색하려는 데이터가 중간에 위치한 기준 데이터보다 크다면, 탐색 범위를 기준 데이터의 오른쪽 그룹으로 좁힌다.

❸ 더 이상 좁혀 나갈 범위가 없어질 때까지 ❶, ❷를 반복한다.

예제 다음과 같이 섞여 있는 공책 중에서 2번 공책을 이진 탐색하는 과정을 알아보자.

풀이

❶ 이진 탐색을 수행하기 위해서는 데이터가 순서대로 정렬되어 있어야 하므로 공책을 번호 순서대로 나열한 뒤 탐색을 시작한다.

❷ 탐색 범위의 중간에 위치한 기준 공책은 2번 공책이 아니고, 2번이 3번보다 작으므로 탐색 범위를 기준 공책의 왼쪽 그룹으로 좁힌다.

❸ 좁혀진 탐색 범위의 중간에 위치한 기준 공책이 2번 공책이 아니고, 2번이 1번보다 크므로 탐색 범위를 기준 공책의 오른쪽 그룹으로 좁힌다.

❹ 탐색 범위의 중간에 위치한 기준 공책이 2번 공책이므로 탐색을 종료한다.

설명 이진 탐색 알고리즘에 따라 중간에 위치한 기준 공책을 2번 공책과 비교하여 탐색 범위를 좁히며 탐색을 수행하면 총 3회의 비교만으로 2번 공책을 찾을 수 있다.

Tip
탐색 범위의 데이터 개수가 짝수일 때, 중간 데이터는 첫 번째와 마지막 데이터의 위치 값의 평균의 몫을 사용하여 결정할 수 있다. 1번, 2번 공책의 위치 값의 평균은 (1+2)/2=1.5이므로, 1번 공책이 중간 데이터로 선택된다.

해 보기 6 이진 탐색 알고리즘으로 탐색하기

다음 마트료시카 인형은 크기가 작은 것부터 A~H까지의 고유한 알파벳 이름을 갖는다. 이진 탐색 알고리즘을 사용하여 E 인형을 탐색할 때 총 비교 횟수를 알아보자.

초기 상태

정렬을 수행한 상태

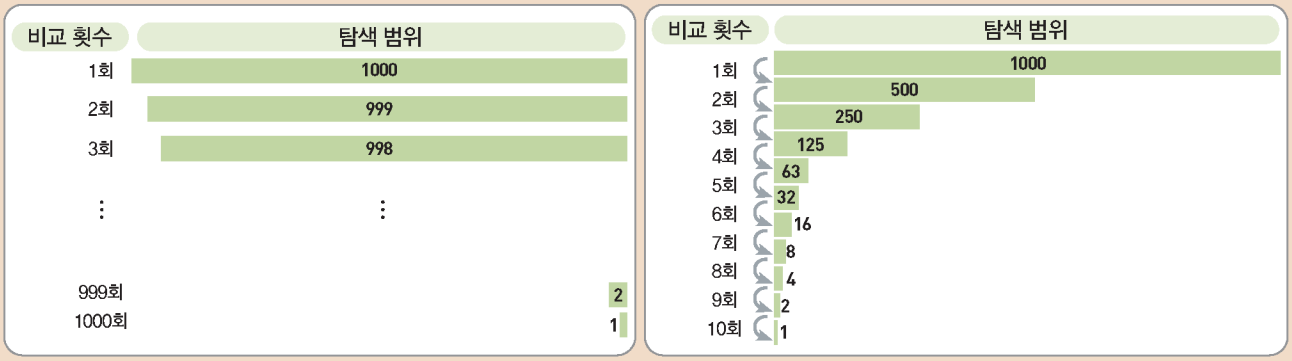
• 비교 횟수:

03 순차 탐색과 이진 탐색 알고리즘의 분석

순차 탐색은 알고리즘이 단순하고 데이터가 정렬되어 있지 않은 경우에도 적용할 수 있지만 탐색할 데이터의 양에 따라 비교 횟수가 크게 달라질 수 있다. 예를 들어 10개의 데이터를 순차 탐색한다면 최악의 경우 10번의 비교가 필요하지만, 1,000개의 데이터를 순차 탐색한다면 최악의 경우 1,000개의 데이터를 모두 비교해야 하기 때문에 순차 탐색은 데이터의 양이 적은 경우에 효과적이다.

이진 탐색은 비교 결과에 따라 탐색 범위가 계속해서 절반으로 줄어들기 때문에 데이터의 수가 많은 경우 훨씬 적은 비교만으로 원하는 데이터를 찾을 수 있다. 1,000개의 데이터에서 이진 탐색을 할 때, 최악의 경우에도 10번의 비교만으로 원하는 데이터를 찾을 수 있기 때문에 순차 탐색에 비해 효율적이다. 단, 이진 탐색을 적용하려면 데이터는 반드시 정렬되어 있어야 한다.

1,000개의 데이터에서 순차 탐색, 이진 탐색을 수행할 때, 최악의 경우 비교 횟수와 탐색 범위



소단원 1분 요약

❶ 데이터를 기준에 따라 순서대로 나열하는 것을 정렬이라고 하며, 대표적인 정렬 알고리즘으로는 선택 정렬과 퀵 정렬이 있다.

❷ 선택 정렬 알고리즘은 정렬되지 않은 데이터 중 가장 작은 데이터를 선택하여 알맞은 위치로 옮기며 정렬하는 방법이며, 작은 규모의 데이터를 정렬할 때 효과적이다. 퀵 정렬 알고리즘은 기준이 되는 값을 정하고 데이터를 두 개의 그룹으로 나누는 과정을 반복하며 정렬하는 방법으로, 기준 값을 어떻게 선택하는지에 따라 효율성이 달라질 수 있다.

❸ 데이터 중에서 원하는 값을 찾아내는 것을 탐색이라고 하며, 대표적인 탐색 알고리즘으로는 순차 탐색과 이진 탐색이 있다.

❹ 순차 탐색 알고리즘은 처음부터 끝까지 순차적으로 비교하며 탐색하는 방법으로, 데이터의 양이 적은 경우 효과적이다. 이진 탐색은 탐색 범위를 반으로 줄여 나가며 탐색하는 방법으로 데이터가 많은 경우 순차 탐색보다 효율적이지만 탐색 전 데이터가 정렬되어 있어야 한다.

소단원 자기 평가

평가 항목	평가 기준		
	잘함	보통	노력
1. [지식·이해] 다양한 정렬 알고리즘의 원리와 수행 과정을 설명할 수 있다.			
2. [지식·이해] 다양한 탐색 알고리즘의 원리와 수행 과정을 설명할 수 있다.			
3. [과정·기능] 데이터를 정렬하는 알고리즘의 특징과 효율을 비교하고 분석할 수 있다.			
4. [과정·기능] 데이터를 탐색하는 알고리즘의 특징과 효율을 비교하고 분석할 수 있다.			
5. [가치·태도] 알고리즘 효율의 가치와 영향력을 인식하고 적극적으로 탐구하려는 태도를 함양할 수 있다.			

정보 탐구 활동

탐색 프로그램 작성하기

다음은 순차 탐색과 이진 탐색 알고리즘을 파이썬으로 구현한 프로그램이다. 파이썬 프로그램을 작성하고 물음에 답해 보자.

프로그램

```
import random
data = []                                  # 비어 있는 data 리스트 생성

# ㉮
for i in range(1000):                      # 데이터 집합의 크기를 1000으로 설정하고 1000개의 데이터 생성
  num = random.randint(1, 1000)            # 1 이상 1000 이하의 랜덤한 수를 변수 num에 대입
  data.append(num)                         # 변수 num을 data 리스트에 추가

count1, count2 = 0, 0                      # 각 정렬 알고리즘의 비교 횟수를 저장할 count1, count2에 0 대입
print(data)                                # 1000개의 수가 저장된 data 리스트 출력
search = int(input('검색할 데이터를 입력하세요: '))
def search1(a, x):                         # search1 함수 정의
  global count1                            # 비교 횟수를 저장할 변수 count1을 수정하기 위한 코드
  n = len(a)                               # a의 길이를 변수 n에 대입
  for i in range(0, n):
    count1 = count1 + 1
    if x == a[i]:
      return '순차 탐색할 경우 비교 횟수는 '+ str(count1) + '회입니다.'  # 변수에 대입
  return '존재하지 않습니다.'                                          # 변수에 대입
def search2(a, x):                         # search2 함수 정의
  a = sorted(a)                            # ㉯
  global count2                            # 비교 횟수를 저장할 변수 count2를 수정하기 위한 코드
  start = 0                                # 변수 start에 첫번째 데이터의 인덱스 값을 대입
  end = len(a) - 1                         # 변수 end에 마지막 데이터의 인덱스 값을 대입
  while start <= end :
    mid = (start + end) // 2
    count2 = count2 + 1
    if x == a[mid]:
      return '이진 탐색할 경우 비교 횟수는 ' + str(count2) + '회입니다.'
    elif x > a[mid] :
      start = mid + 1
    else :
      end = mid - 1
  return '존재하지 않습니다.'
print(search1(data, search))
print(search2(data, search))
```

Tip random은 파이썬에 내장된 난수를 생성해 주는 모듈로, random.randint(1, 10)과 같이 사용할 경우, 1 이상 10 이하 범위의 정수 난수를 생성한다.

1 작성한 프로그램을 실행하여 비교 횟수를 확인해 보자.

검색한 데이터	비교 횟수	
	순차 탐색	이진 탐색

2 ㉮에서 데이터 집합의 크기와 수의 범위를 자유롭게 수정하며 프로그램을 실행해 보고, 탐색 가능 여부와 비교 횟수를 확인해 보자.

데이터 집합의 크기	검색 데이터	비교 횟수	
		순차 탐색	이진 탐색

3 21행의 ㉯ 문장은 데이터 집합을 정렬하는 문장이다. ㉯ 문장을 삭제하여 프로그램을 실행해 보고, 그렇게 실행되는 이유를 생각해 보자.

실행 결과	그렇게 실행된 이유

4 1~3 의 결과를 바탕으로 순차 탐색과 이진 탐색의 특징을 정리해 보고, 알고리즘의 효율성을 비교하고 분석해 보자.

구분	순차 탐색	이진 탐색
알고리즘 특징		
효율성 분석		

탐구 활동 자기 평가	평가 항목	평가 기준		
		잘함	보통	노력
	1. 탐색 알고리즘의 특징을 비교하여 설명할 수 있는가?.			
	2. 탐색 알고리즘에 따른 효율을 비교하여 설명할 수 있는가?			

• 잘함 (내용을 이해하고 설명함) • 보통 (내용을 이해함) • 노력 (내용을 부분적으로 이해함)