선택정렬

모두의 알고리즘 with 파이썬 - 문제 08

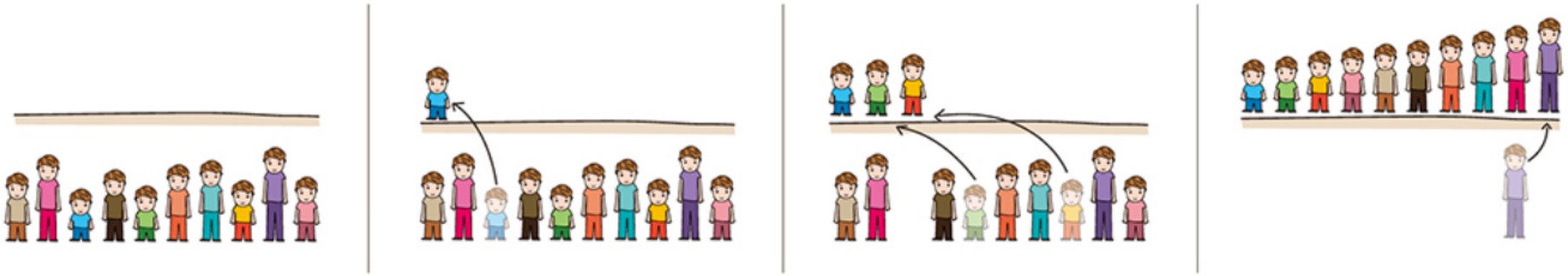
20.05.10 장예훈

🌸 정렬 Sort 🌼

수 많은 정렬 알고리즘 중 ...

- 선택 정렬 (Selection Sort)
- 삽입 정렬 (Insertion Sort)
- 병합 정렬 (Merge Sort)
- 퀵 정렬 (Quick Sort)
- 거품 정렬 (Bubble Sort)

선택 정렬로 줄 세우기



‘키 순서대로 줄 세우기’는 대표적인 정렬 문제의 예시

🔽

학생의 키 라는 자료값을 **작은 것 부터 큰 순서**대로 나열

쉽게 설명한 정렬 알고리즘

vs

일반적인 정렬 알고리즘

정렬 알고리즘

쉽게 설명한 정렬 알고리즘

효율성을 크게 고려하지 않고 정렬 방식이 어떤지를 단순히 보여 주기 위해 만든 참고용 프로그램

일반적인 정렬 알고리즘

효율적으로 정렬 알고리즘을 정식으로 구현한 프로그램

쉽게 설명한 선택 알고리즘

```python
def find_min_idx(a):
    n = len(a)
    min_idx = 0
    for i in range(1, n):
        if a[i] < a[min_idx]:
            min_idx = i
    return min_idx

def sel_sort(a):
    result = []
    while a:
        min_idx = find_min_idx(a)
        value = a.pop(min_idx)
        result.append(value)
    return result
```

```python
d = [2, 4, 5, 1, 3]
print(sel_sort(d))
```

```
[1, 2, 3, 4, 5]
```

입력: 리스트 a / 출력: 정렬된 새 리스트

find_min_idx

주어진 리스트에서 최솟값의 위치를 돌려주는 함수

sel_sort

입력된 리스트를 정렬해주는 함수

쉽게 설명한 선택 알고리즘

```python
def find_min_idx(a):
    n = len(a)
    min_idx = 0
    for i in range(1, n):
        if a[i] < a[min_idx]:
            min_idx = i
    return min_idx

def sel_sort(a):
    result = []
    while a:
        min_idx = find_min_idx(a)
        value = a.pop(min_idx)
        result.append(value)
    return result
```

```python
d = [2, 4, 5, 1, 3]
print(sel_sort(d))
```

```
[1, 2, 3, 4, 5]
```

‼ 리스트의 **pop** 메서드 ‼

```python
a = [1, 2, 3, 4, 5]
a.pop()
```

```
5
```

```python
a
```

```
[1, 2, 3, 4]
```

```python
a.pop(1)
```

```
2
```

```python
a
```

```
[1, 3, 4]
```

쉽게 설명한 선택 알고리즘

```
 1  def find_min_idx(a):
 2      n = len(a)
 3      min_idx = 0
 4      for i in range(1, n):
 5          if a[i] < a[min_idx]:
 6              min_idx = i
 7      return min_idx
 8
 9  def sel_sort(a):
10      result = []
11      while a:
12          min_idx = find_min_idx(a)
13          value = a.pop(min_idx)
14          result.append(value)
15      return result
```

```
d = [2, 4, 5, 1, 3]
print(sel_sort(d))
```

```
[1, 2, 3, 4, 5]
```

입력: 리스트 a / 출력: 정렬된 새 리스트

line 10

새 리스트를 만들어 정렬된 값을 저장

line 12

리스트에 남아 있는 값 중 최솟값의 위치

line 13

찾은 최솟값을 빼내어 value에 저장

line 14

value를 결과 리스트 끝에 추가

쉽게 설명한 선택 알고리즘

```python
def find_min_idx(a):
    n = len(a)
    min_idx = 0
    for i in range(1, n):
        if a[i] < a[min_idx]:
            min_idx = i
    return min_idx

def sel_sort(a):
    result = []
    while a:
        min_idx = find_min_idx(a)
        value = a.pop(min_idx)
        result.append(value)
    return result
```

```python
d = [2, 4, 5, 1, 3]
print(sel_sort(d))
```

```
[1, 2, 3, 4, 5]
```

```
** 1 loop **
a: [2, 4, 5, 1, 3]
min_idx: 3
value: 1
result: [1]
------------------
** 2 loop **
a: [2, 4, 5, 3]
min_idx: 0
value: 2
result: [1, 2]
------------------
** 3 loop **
a: [4, 5, 3]
min_idx: 2
value: 3
result: [1, 2, 3]
------------------
```

```
** 4 loop **
a: [4, 5]
min_idx: 0
value: 4
result: [1, 2, 3, 4]
------------------
** 5 loop **
a: [5]
min_idx: 0
value: 5
result: [1, 2, 3, 4, 5]
------------------
```

일반적인 선택 정렬 알고리즘

리스트 a 안에서 직접 자료의 위치를 바꾸면서 정렬

```
1  def sel_sort_2(a):
2      n = len(a)
3      for i in range(n-1):
4          min_idx = i
5          for j in range(i+1, n):
6              if a[j] < a[min_idx]:
7                  min_idx = j
8          a[i], a[min_idx] = a[min_idx], a[i]
```

```
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

입력: 리스트 a / 출력: 없음

line 3

두 값의 비교를 위해 기준이 되는 값은 n-1개

line 4 ~ 7

i+1번째 값부터 마지막 자료값 중 최솟값을 찾음

line 8

최솟값과 i번째 위치 바꿈

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[2, 4, 5, 1, 3]

a[min_idx]

i = 0 min_idx = 0 j = 1

a[i] = 2 a[min_idx] = 2 a[j] = 4

a[j] **>** a[min_idx]

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[2, 4, 5, 1, 3]

a[min_idx]

i = 0	min_idx = 0	**j = 2**
a[i] = 2	a[min_idx] = 2	**a[j] = 5**

a[j] **>** a[min_idx]

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[2, 4, 5, 1, 3]

a[min_idx]

i = 0	min_idx = 0	**j =3**
a[i] = 2	a[min_idx] = 2	**a[j] = 1**

a[j] **<** a[min_idx]

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[2, 4, 5, 1, 3]

a[min_idx]

i = 0 **min_idx = 3** j =3

a[i] = 2 **a[min_idx] = 1** a[j] = 1

a[j] **<** a[min_idx]

▶️ min_idx = j

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[2, 4, 5, 1, 3]

a[min_idx]

i = 0	min_idx = 3	**j =4**
a[i] = 2	a[min_idx] = 1	**a[j] = 5**

a[j] **>** a[min_idx]

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[**1**, 4, 5, **2**, 3]

a[min_idx]

i = 0 min_idx = 3 j =4

a[i] = 1 **a[min_idx] = 2** a[j] = 5

a[i], a[min_idx] = a[min_idx], a[i]

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[1, 4, 5, 2, 3]

a[min_idx]

i = 1 **min_idx = 1** **j =2**

a[i] = 4 **a[min_idx] = 4** **a[j] = 5**

a[j] **>** a[min_idx]

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[1, 4, 5, 2, 3]

a[min_idx]

i = 1	min_idx = 1	**j =3**
a[i] = 4	a[min_idx] = 4	**a[j] = 2**

a[j] **<** a[min_idx]

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[1, 4, 5, 2, 3]

a[min_idx]

i = 1 **min_idx = 3** j =3

a[i] = 4 **a[min_idx] = 2** a[j] = 2

a[j] **<** a[min_idx]

▶️ min_idx = j

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[1, 4, 5, 2, 3]

a[min_idx]

i = 1	min_idx = 3	**j =4**
a[i] = 4	a[min_idx] = 2	**a[j] = 3**

a[j] **>** a[min_idx]

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

a[i] a[j]

[1, **2**, 5, **4**, 3]

a[min_idx]

i = 1 min_idx = 3 j =4

a[i] = 2 **a[min_idx] = 4** a[j] = 3

a[i], a[min_idx] = a[min_idx], a[i]

일반적인 선택 정렬 알고리즘

```python
def sel_sort_2(a):
    n = len(a)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if a[j] < a[min_idx]:
                min_idx = j
        a[i], a[min_idx] = a[min_idx], a[i]
```

```python
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[1, 2, 3, 4, 5]
```

```
** 1 loop **
a: [2, 4, 5, 1, 3]
a: [1, 4, 5, 2, 3]
--------------------
** 2 loop **
a: [1, 4, 5, 2, 3]
a: [1, 2, 5, 4, 3]
--------------------
** 3 loop **
a: [1, 2, 5, 4, 3]
a: [1, 2, 3, 4, 5]
--------------------
** 4 loop **
a: [1, 2, 3, 4, 5]
a: [1, 2, 3, 4, 5]
--------------------
```

알고리즘 분석

자료를 크기 순서대로 정렬하기 위해 **반드시 두 수의 크기를 비교**해야 하기 때문에

정렬 알고리즘의 계산 복잡도는 보통 **비교 횟수**를 기준으로 함

계산 복잡도는 $O(n^2)$ ➡️ 입력 크기의 제곱에 비례하여 복잡도 증가

연습 문제 8-1

일반적인 선택 정렬 알고리즘을 사용해서 리스트[2, 4, 5, 1, 3]을 정렬하는 과정을 적어보세요.

➡️ 앞에 !! 😄

연습 문제 8-2

프로그램 8-1과 8-2의 정렬 알고리즘은 숫자를 작은 수에서 큰 수 순서로 나열하는 오름차순 정렬이었습니다. 이 알고리즘을 큰 수에서 작은 수 순서로 나열하는 내림차순 정렬로 바꾸려면 프로그램의 어느 부분을 바꿔야 할까요?

```
1  def sel_sort_2(a):
2      n = len(a)
3      loop = 0
4      for i in range(n-1):
5          loop+=1
6          min_idx = i
7          for j in range(i+1, n):
8              if a[j] > a[min_idx]:
9                  min_idx = j
10         a[i], a[min_idx] = a[min_idx], a[i]
```

```
d = [2, 4, 5, 1, 3]
sel_sort_2(d)
print(d)
```

```
[5, 4, 3, 2, 1]
```

감사합니다 😊