과제 #6 정렬 함수 구현 및 성능 측정

알고리즘과 문제해결

20180018 강 승아

1. 실행 스냅 샷

(1) 버블정렬

```
Terminal:  Local  +
C:\Users\강승아\PycharmProjects\pythonProject4> python -m cProfile bubbleSort.py
[:10] : [4, 5, 6, 12, 25, 27, 33, 38, 46, 52]
 가독성을 위한 생략
 [-10:] :  [9935, 9936, 9936, 9938, 9941, 9950, 9961, 9969, 9979, 9995]
오름차순 정렬이 올바르게 되어 있습니다.
         16321 function calls (16288 primitive calls) in 0.915 seconds
```

소요 시간 : 0.915 sec

(2) 선택정렬

```
Terminal:  Local  +
C:\Users\강승아\PycharmProjects\pythonProject4> python -m cProfile selectionSort.py
 [:10] : [4, 5, 6, 12, 25, 27, 33, 38, 46, 52]
 가독성을 위한 생략
 [-10:] :  [9935, 9936, 9936, 9938, 9941, 9950, 9961, 9969, 9979, 9995]
오름차순 정렬이 올바르게 되어 있습니다.
         16321 function calls (16288 primitive calls) in 0.412 seconds
```

소요 시간 : 0.412 sec

(3) 삽입정렬

```
Terminal:  Local  +
C:\Users\강승아\PycharmProjects\pythonProject4> python -m cProfile insertSort.py
 [:10] : [4, 5, 6, 12, 25, 27, 33, 38, 46, 52]
 가독성을 위한 생략
 [-10:] :  [9935, 9936, 9936, 9938, 9941, 9950, 9961, 9969, 9979, 9995]
오름차순 정렬이 올바르게 되어 있습니다.
         14322 function calls (14289 primitive calls) in 0.432 seconds
```

소요 시간 : 0.432 sec

(4) 셸정렬

```
Terminal:  Local  +
C:\Users\강승아\PycharmProjects\pythonProject4> python -m cProfile shellSort.py
 [:10] : [4, 5, 6, 12, 25, 27, 33, 38, 46, 52]
 가독성을 위한 생략
 [-10:] :  [9935, 9936, 9936, 9938, 9941, 9950, 9961, 9969, 9979, 9995]
오름차순 정렬이 올바르게 되어 있습니다.
         14330 function calls (14297 primitive calls) in 0.050 seconds
```

소요 시간 : 0.050 sec

(5) 힙정렬

```
Terminal:  Local  +
C:\Users\강승아\PycharmProjects\pythonProject4> python -m cProfile heapSort.py
 [:10] : [4, 5, 6, 12, 25, 27, 33, 38, 46, 52]
 가독성을 위한 생략
 [-10:] :  [9935, 9936, 9936, 9938, 9941, 9950, 9961, 9969, 9979, 9995]
오름차순 정렬이 올바르게 되어 있습니다.
         20321 function calls (20288 primitive calls) in 0.041 seconds
```

소요 시간 : 0.041 sec

(6) 기수정렬

```
Terminal:  Local  +
C:\Users\강승아\PycharmProjects\pythonProject4> python -m cProfile radixSort.py
 [:10] : [4, 5, 6, 12, 25, 27, 33, 38, 46, 52]
 가독성을 위한 생략
 [-10:] :  [9935, 9936, 9936, 9938, 9941, 9950, 9961, 9969, 9979, 9995]
오름차순 정렬이 올바르게 되어 있습니다.
         22366 function calls (22333 primitive calls) in 0.059 seconds
```

소요 시간 : 0.059 sec

2. 알고리즘 실행 시간 표

정렬 방법	소요 시간(seconds)
버블정렬	0.915
선택정렬	0.412
삽입정렬	0.432
셸정렬	0.050
힙정렬	0.041
기수정렬	0.059

3. 정렬 알고리즘들의 성능 분석

- ▶ 0~10000사이의 값으로 생성된 난수 리스트 RandomData[2000]은 main 함수 import를 통해 각 python 파일로부터 정렬되었다.
- ▶ main 함수에 고정된 random.seed(100)값에 따라 RandomData는 항상 같은 값을 유지하므로 여섯 개의 정렬 알고리즘 모두 같은 리스트의 정렬을 시도하였다.

* 효율성

힙정렬 > 셸정렬 > 기수정렬 >>> 선택정렬 > 삽입정렬 >>> 버블정렬

힙정렬이 0.040초대의 가장 훌륭한 성능을 보이고 셸정렬, 기수정렬은 0.050초 정도의 비슷한 시간을 소모하였으며, 선택정렬 삽입정렬은 0.400초대의 비슷한 성능, 버블정렬은 유일한 0.900초대로 가장 나쁜 성능을 보였다. 같은 결과를 도출하면서도 확연히 다른 소모 시간을 보이며 좋은 알고리즘의 중요성을 확인할 수 있다.