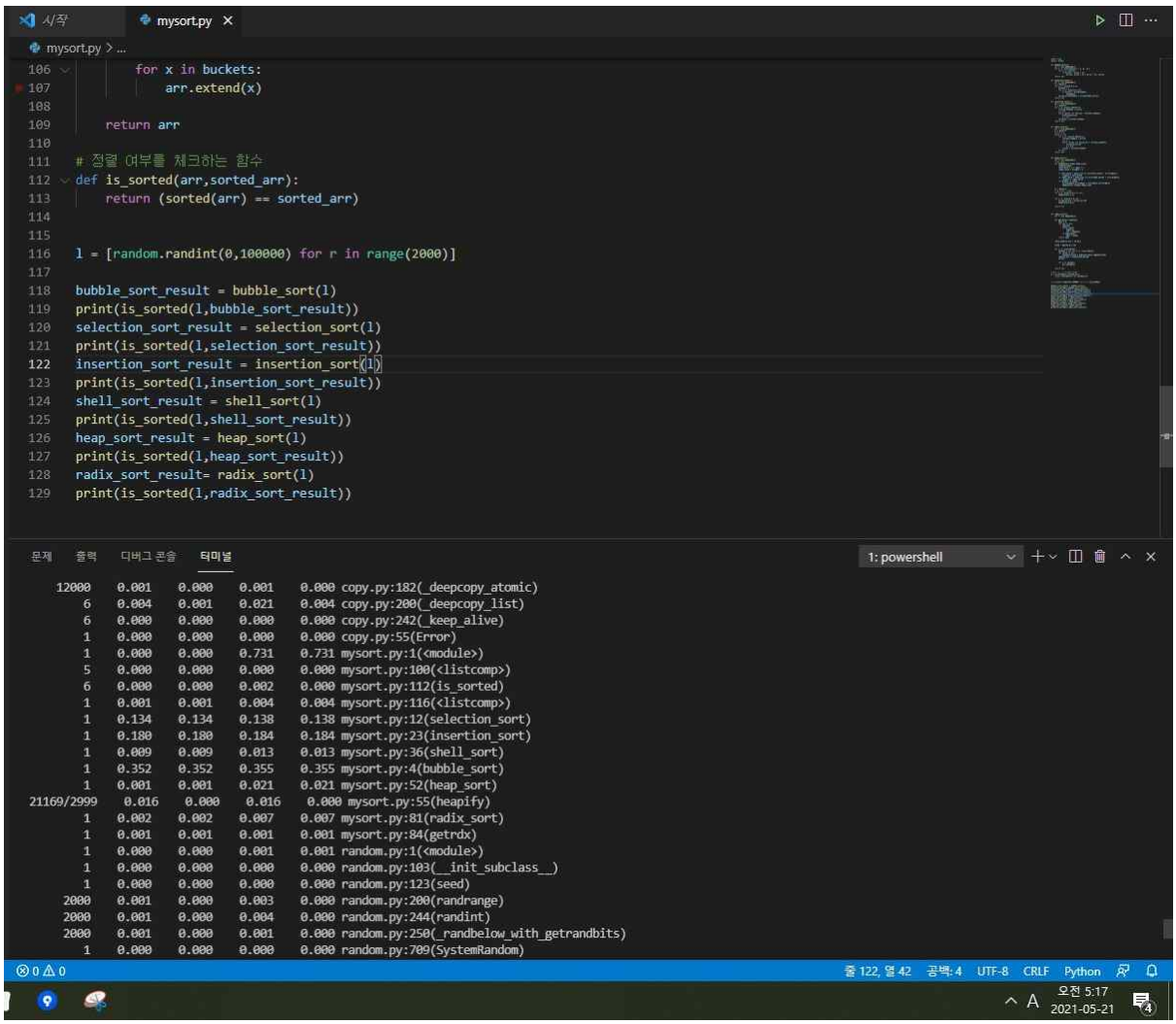
알고리즘과 문제해결 일반과제6

20181061 전하영

1. 실행 스냅샷

- python -m cProfile mysort.py 실행 결과 함수별 실행 시간



```
        for x in buckets:
            arr.extend(x)

    return arr

# 정렬 여부를 체크하는 함수
def is_sorted(arr,sorted_arr):
    return (sorted(arr) == sorted_arr)


l = [random.randint(0,100000) for r in range(2000)]

bubble_sort_result = bubble_sort(l)
print(is_sorted(l,bubble_sort_result))
selection_sort_result = selection_sort(l)
print(is_sorted(l,selection_sort_result))
insertion_sort_result = insertion_sort(l)
print(is_sorted(l,insertion_sort_result))
shell_sort_result = shell_sort(l)
print(is_sorted(l,shell_sort_result))
heap_sort_result = heap_sort(l)
print(is_sorted(l,heap_sort_result))
radix_sort_result= radix_sort(l)
print(is_sorted(l,radix_sort_result))
```

```
문제  출력  디버그 콘솔  터미널                                1: powershell

    12000    0.001    0.000    0.001    0.000 copy.py:182(_deepcopy_atomic)
        6    0.004    0.001    0.021    0.004 copy.py:200(_deepcopy_list)
        6    0.000    0.000    0.000    0.000 copy.py:242(_keep_alive)
        1    0.000    0.000    0.000    0.000 copy.py:55(Error)
        1    0.000    0.000    0.731    0.731 mysort.py:1(<module>)
        5    0.000    0.000    0.000    0.000 mysort.py:100(<listcomp>)
        6    0.000    0.000    0.002    0.000 mysort.py:112(is_sorted)
        1    0.001    0.001    0.004    0.004 mysort.py:116(<listcomp>)
        1    0.134    0.134    0.138    0.138 mysort.py:12(selection_sort)
        1    0.180    0.180    0.184    0.184 mysort.py:23(insertion_sort)
        1    0.009    0.009    0.013    0.013 mysort.py:36(shell_sort)
        1    0.352    0.352    0.355    0.355 mysort.py:4(bubble_sort)
        1    0.001    0.001    0.021    0.021 mysort.py:52(heap_sort)
21169/2999    0.016    0.000    0.016    0.000 mysort.py:55(heapify)
        1    0.002    0.002    0.007    0.007 mysort.py:81(radix_sort)
        1    0.001    0.001    0.001    0.001 mysort.py:84(getrdx)
        1    0.000    0.000    0.001    0.001 random.py:1(<module>)
        1    0.000    0.000    0.000    0.000 random.py:103(__init_subclass__)
        1    0.000    0.000    0.000    0.000 random.py:123(seed)
     2000    0.001    0.000    0.003    0.000 random.py:200(randrange)
     2000    0.001    0.000    0.004    0.000 random.py:244(randint)
     2000    0.001    0.000    0.001    0.000 random.py:250(_randbelow_with_getrandbits)
        1    0.000    0.000    0.000    0.000 random.py:709(SystemRandom)
```

- 리스트가 정렬된 것인지를 체크하는 함수를 작성하고 반환된 리스트마다 체크 함수를 수행한 결과

```
116   l = [random.randint(0,100000) for r in range(2000)]
117
118   bubble_sort_result = bubble_sort(l)
119   print(is_sorted(l,bubble_sort_result))
120   selection_sort_result = selection_sort(l)
121   print(is_sorted(l,selection_sort_result))
122   insertion_sort_result = insertion_sort(l)
123   print(is_sorted(l,insertion_sort_result))
124   shell_sort_result = shell_sort(l)
125   print(is_sorted(l,shell_sort_result))
126   heap_sort_result = heap_sort(l)
127   print(is_sorted(l,heap_sort_result))
128   radix_sort_result= radix_sort(l)
129   print(is_sorted(l,radix_sort_result))
```

```
문제   출력   디버그 콘솔   터미널                    1: powershell
      306    0.000    0.000    0.000    0.000 {method 'rstrip' of 'str' objects}
      209    0.000    0.000    0.000    0.000 {method 'startswith' of 'str' objects}

PS C:\Users\hayoungzzzzz\Dropbox\내 PC (DESKTOP-G22A87B)\Desktop\4학년 1학기\알고리즘과문제
\일반과제 6> python -m cProfile mysort.py
True
True
True
True
True
True
         116418 function calls (86191 primitive calls) in 0.611 seconds
```

2. 정렬 알고리즘들의 실행 시간을 표로 작성

1	0.138	0.138	0.141	0.141	mysort.py:12(selection_sort)
1	0.130	0.130	0.133	0.133	mysort.py:23(insertion_sort)
1	0.006	0.006	0.009	0.009	mysort.py:36(shell_sort)
1	0.288	0.288	0.291	0.291	mysort.py:4(bubble_sort)
1	0.001	0.001	0.016	0.016	mysort.py:52(heap_sort)
21189/2999	0.012	0.000	0.012	0.000	mysort.py:55(heapify)
1	0.002	0.002	0.008	0.008	mysort.py:81(radix_sort)
1	0.001	0.001	0.001	0.001	mysort.py:84(getrdx)

위는 각 정렬 알고리즘들의 실행 시간이다. 정렬 함수 안에 정의해둔 내부 함수의 실행 시간도 포함했다. 다른 함수 호출을 포함하여 함수에서 보낸 시간(cumtime)을 보아야 한다.

정렬 알고리즘	실행 시간(sec)	시간복잡도	비고
버블 정렬(bubble_sort)	0.291	O(n^2)	
선택 정렬(selection_sort)	0.141	O(n^2)	
삽입 정렬(insertion_sort)	0.133	O(n^2)	
셸 정렬(shell_sort)	0.009	O(n^2)	
힙 정렬(heap_sort)	0.291	O(nlogn)	heapify 함수 호출 포함
기수 정렬(radix_sort)	0.008	d * O(n)	getrdx 함수 호출 포함

힙 정렬의 heapify와 기수 정렬의 getrdx의 호출 시간도 포함되어있다.

3. 정렬 알고리즘들의 성능 분석

지금까지 배운 내부정렬 알고리즘들의 성능을 cProfile을 이용해 실행 시간을 기준으로 비교할 수 있었다.

평균 시간복잡도를 기준으로 가장 느린 정렬 알고리즘은 버블 정렬이고, 가장 빠른 정렬 알고리즘은 기수 정렬이다.

실행 시간을 기준으로 정렬이 빠른 순서는 다음과 같다.

기수 정렬 > 셸 정렬 > 힙 정렬 > 삽입 정렬 > 선택 정렬 > 버블 정렬

확실히 비교정렬이 아닌 기수 정렬이 우수한 성능을 보였다.
정수의 개수가 2000개로 적기 때문에, 정수 범위를 늘린다면 확실한 실행 시간 차이를 볼 수 있을 것이다. 궁금해서 요소의 개수를 5000개로 늘리고 프로그램을 다시 돌려보았다.

```
        1    0.753    0.753    0.762    0.762 mysort.py:12(selection_sort)
        1    0.831    0.831    0.842    0.842 mysort.py:23(insertion_sort)
        1    0.020    0.020    0.028    0.028 mysort.py:36(shell_sort)
        1    1.971    1.971    1.979    1.979 mysort.py:4(bubble_sort)
        1    0.002    0.002    0.045    0.045 mysort.py:52(heap_sort)
59724/7499    0.034    0.000    0.034    0.000 mysort.py:55(heapify)
        1    0.005    0.005    0.020    0.020 mysort.py:81(radix_sort)
        1    0.002    0.002    0.002    0.002 mysort.py:84(getrdx)
```

요소의 개수가 늘어나니 버블 정렬과 기수 정렬의 실행 속도 차이가 점점 벌어졌다.
버블 정렬은 1.979초나 걸렸지만, 그에 반해 기수 정렬은 0.002초 우수한 성능을 보여주었다.