

A1.

```
1 # 이진 검색 작동확인
2 A = Data(size=128)
3 print(A.data)
4 Loc, k = A.binary_search(10)
5 print(Loc, k)
```

[1, 1, 2, 2, 3, 6, 7, 8, 8, 8, 9, 9, 10, 11, 11, 12, 14, 15, 16, 16, 16, 20, 21, 21, 21, 23, 23, 25, 12 6

위 사진과 같이, 객체지향방법으로 구현한 이진검색이 제대로 동작하는 것을 확인할 수 있다.

(관찰)

```
1 ##### A1 #####
2
3 C = [] # n값에 따른 1000개의 문제 p의 평균 데이터 비교 횟수
4
5 for n in [128, 256, 512]:
6     S = Data(size=n)
7     c = 0
8
9     for i in range(1000):
10        x = randrange(1, n+1) # 하나의 문제 pi를 생성
11        loc, ki = S.binary_search(x) # x를 찾을때까지의 비교 횟수 ki를 계산
12        c += ki
13
14    c = int(c/1000) # 평균 데이터 비교횟수 c
15    C.append(c)
16
17 print(C) # n=128, n=256, n=512일때의 c값
```

[5, 6, 7]

1000개의 문제 p에 대한 평균 비교횟수 c는

n = 128일 때 5

n = 256일 때 6

n = 512일 때 7이다.

n과 c와의 관계는  $n = 2^k$  ( $k \geq 3$  인 자연수) 일 때  $c = 2^{k-2}$  인 것을 관찰할 수 있다. 다만 n의 수가 작을 때는 생성되는 S의 데이터 분포에 따라 오차가 있을 수

있다.

```
✓ 0초 ▶ 1 ##### A1 #####
2
3 C = [] # n값에 따른 1000개의 문제 p의 평균 데이터 비교 횟수
4
5 for n in [8, 16, 32, 64, 128, 256, 512, 1024]:
6     S = Data(size=n)
7     c = 0
8
9     for i in range(1000):
10         x = randrange(1, n+1) # 하나의 문제 pi를 생성
11         loc, ki = S.binary_search(x) # x를 찾을때까지의 비교 횟수 ki를 계산
12         c += ki
13
14     c = int(c/1000) # 평균 데이터 비교횟수 c
15     C.append(c)
16
17 print(C) # n=128, n=256, n=512일때의 c값
```

[1, 2, 3, 4, 5, 6, 7, 8]

n이 2^3부터 2^10까지의 실행결과

A2.

```
✓ 초 ▶ 1 ##### A2 #####
2
3 # merge_sort 검증
4 data = [8, 3, 15, 2, 9, 1, 5, 7, 4, 16, 10, 11, 12, 13, 6, 14]
5 extra_mem_size = merge_sort(data)
6 print(data)
7 # 추가로 필요한 메모리의 크기
8 print(extra_mem_size)
```

➡ [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]  
31

문제 2에서 주어진 데이터를 올바르게 정렬하고, 추가로 필요한 메모리의 크기 ( $\approx 2n$ )까지 계산한 결과.

(공간복잡도가  $2n$ 이 되는 merge sort 알고리즘을 사용함. py파일 참조)



```
1 # merge_sort 검증2
2
3 data2 = [randrange(1, 101) for i in range(100)]
4 print(data2)
5 ems = merge_sort(data2)
6 print(data2)
7 print(ems)
```

[97, 97, 34, 23, 66, 2, 29, 96, 22, 52, 51, 97, 88, 41, 3, 32, 57, 71, 49, 46, 90, 79, 2, 82, 76, 45, 93, 60, 37, 85, 93, 95, 67, 94, 1, 25, 71, 30, 69, 73, 10, 19, 54, 36, 49, 1, 2, 2, 2, 2, 3, 4, 5, 5, 7, 7, 9, 10, 12, 16, 16, 18, 19, 20, 22, 23, 23, 24, 25, 28, 28, 29, 29, 30, 30, 31, 32, 34, 36, 37, 37, 41, 43, 43, 43, 45, 46, 46, 46, 47, 47, 48, 202]

두 번째 merge sort 검증 결과. 랜덤으로 생성한 리스트에 대해서도 비내림차순으로 정렬된 결과를 보여준다.