



Algorithm

정렬 알고리즘 성능분석

2025-04-04

조윤실



목 차



■ 정렬 알고리즘 성능분석

- 1) 정렬 알고리즘 테스트
- 2) 정렬 알고리즘 성능 비교
- 3) 문자열 데이터 정렬
- 4) 이미지 데이터 정렬

5) Quiz

※ 가천대학교 컴퓨터공학과 '알고리즘' 과정

정렬 알고리즘 성능분석

정렬 알고리즘 (기초) 테스트

알고리즘 테스트 항목

■ 정렬 알고리즘 기초 테스트 조건

- 1) 정수 데이터 정렬 : `[11, 1, 51, 1, 5, 3]`
- 2) 빈 데이터 정렬 : `[]`
- 3) 음수 데이터 정렬 : `[1, 1, -5, 6]`
- 4) `numpy.array` 데이터 정렬 : `np.array([11, -4, 20, 15, 13.5, -20])`
- 5) (이미) 정렬이 된 데이터 정렬 : `np.array(range(50))`
- 6) (역순) 정렬이 된 데이터 정렬 : `np.arange(50, 0, -5)`
- 7) 큰 데이터 정렬 : `np.random.randint(-5000, 5000, size=1000)`

실습문제 : 기초 정렬 알고리즘 테스트

- (앞에서 정의된)정렬 알고리즘 기초 테스트 조건에 맞게 테스트 진행하세요.
 - 1) Bubble Sort
 - 2) Selection Sort
 - 3) Insertion Sort

정렬 알고리즘 성능 비교

정렬 알고리즘 성능 측정

- 데이터 크기(Size)별 정렬 시간 측정

- 1) 데이터 발생 : `random.randint(0, 10000)`
- 2) 시간측정 함수 : `measure_time(정렬 알고리즘, 데이터 크기)`
- 3) 데이터 크기 : `sizes = [100, 500, 1000, 5000]`
- 4) 정렬 알고리즘 : `sort_funcs = bubble_sort`
- 5) 알고리즘별 측정 시간 저장 : `bubble_times = []`

정렬 알고리즘 성능 측정

- 데이터 크기(Size)별 정렬 시간 측정 예 : (Bubble Sort)

```
import time
import random

def measure_time(sort_func, size):
    arr = [random.randint(0, 10000) for _ in range(size)]
    start = time.time() # 시간 측정 시작
    sort_func(arr, len(arr))
    end = time.time() # 시간 측정 종료
    return end - start

# 데이터 크기
sizes = [100, 500, 1000, 5000]
for size in sizes:
    print(f"Size {size}: Bubble Sort = {measure_time(bubble_sort, size):.6f} sec")
```

그래프로 표현

```
import time
import random
import matplotlib.pyplot as plt
```

```
def measure_time(sort_func, sizes):
    times = []
    for size in sizes:
        arr = [random.randint(0, 10000) \
                for _ in range(size)]
        start = time.time()
        sort_func(arr, len(arr))
        end = time.time()
        times.append(end - start)
    return times
```

데이터 크기

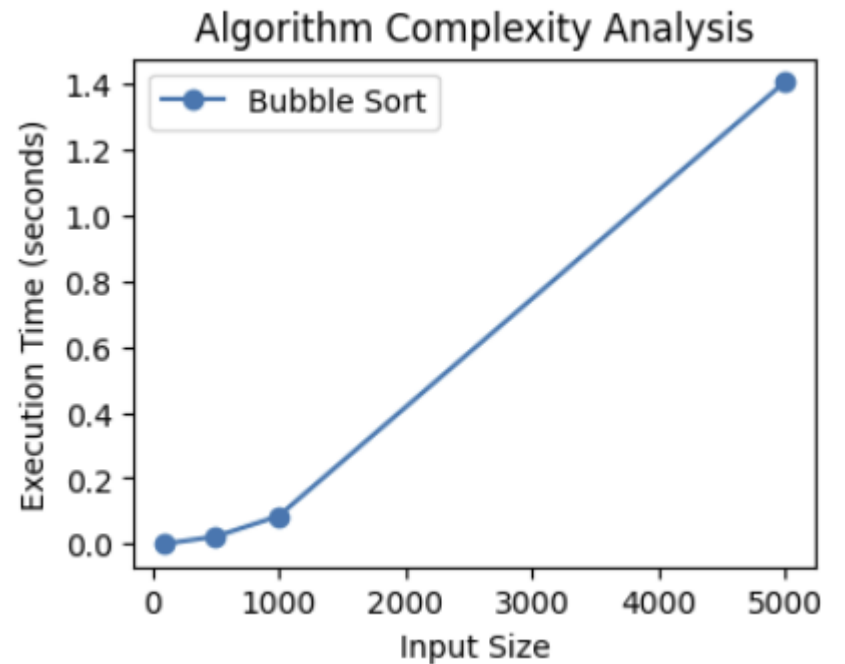
```
sizes = [100, 500, 1000, 5000]
```

시간 측정하여 리스트로 담기

```
bubble_times = measure_time(bubble_sort, sizes)
```

그래프 그리기

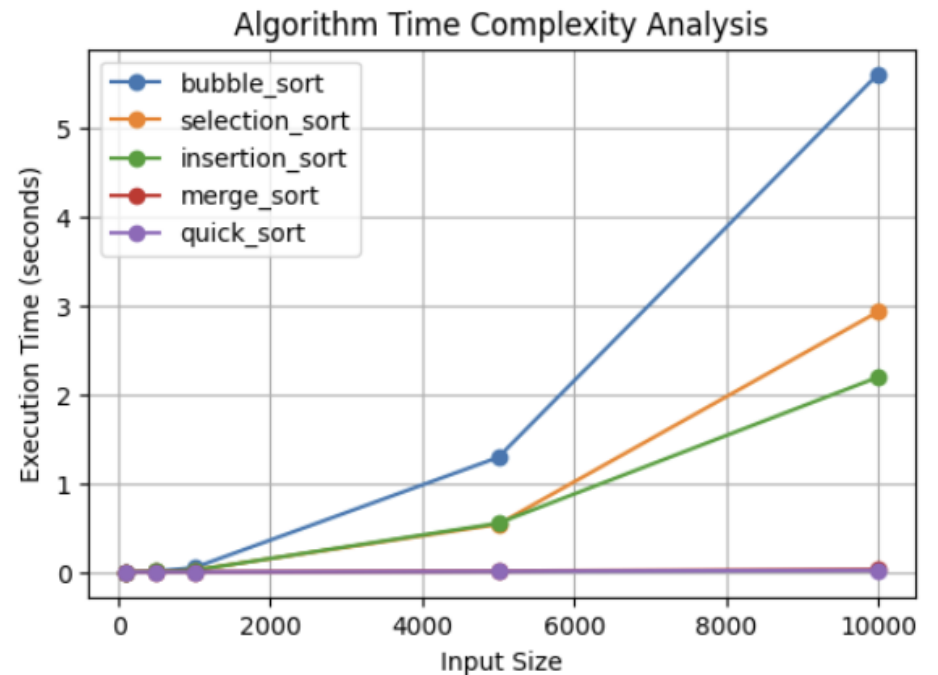
```
plt.figure(figsize=(4, 3))
plt.plot(sizes, bubble_times,
         label="Bubble Sort", marker='o')
plt.xlabel("Input Size")
plt.ylabel("Execution Time (seconds)")
plt.legend()
plt.title("Algorithm Complexity Analysis")
plt.show()
```



실습문제 : 여러 개 알고리즘 시간 성능 측정하기

- 앞에서 작성한 시간 성능 측정 방법을 수정하여 여러 개 정렬 알고리즘의 시간 성능을 비교하여 그래프로 시각화 하세요.

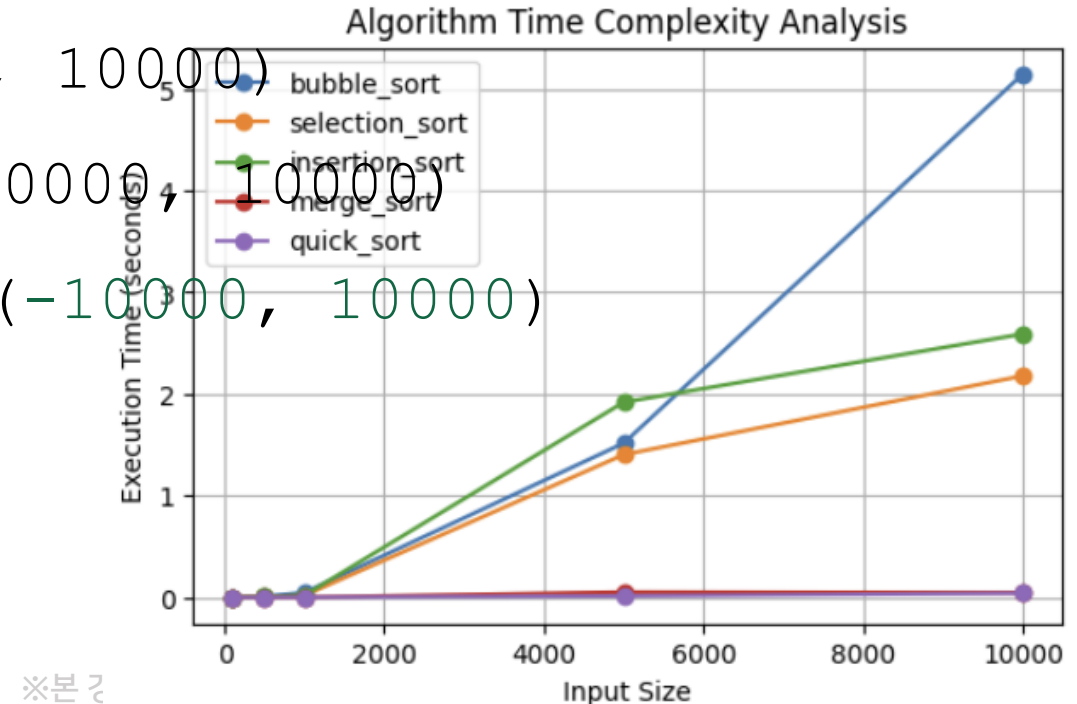
1) `sort_funcs = [bubble_sort, selection_sort, insertion_sort, merge_sort, quick_sort]`



실습문제 : 다양한 종류 데이터 사용한 시간 성능 측정하기

- 앞에서 작성한 시간 성능 측정 방법을 수정하여 다양한 종류의 데이터를 사용하여 정렬 알고리즘의 성능을 측정해 보세요.

- 1) 정수(양의 정수) : `random.randint(0, 10000)`
- 2) 정수(음수포함) : `random.randint(-10000, 10000)`
- 3) `np.array` 정수 : `np.random.randint(-10000, 10000)`



문자열 데이터 정렬

숫자 vs 알파벳 vs 한글

■ 숫자 & 알파벳

- ASCII <https://ko.wikipedia.org/wiki/ASCII>
- 총 128개 문자 정의 (0 ~ 127까지) → 확장 ASCII(총 256개 문자(0 ~ 255))
 - 7비트로 표현 (최대 1byte)

■ 한글

- 유니코드 <https://ko.wikipedia.org/wiki/유니코드>
- 완성형 한글 음절(U+AC00 ~ U+D7A3)은 초성 × 중성 × 종성의 조합으로 생성
 - 유니코드 최대 4bytes로 구성
 - 유니코드는 저장/전송 시 UTF-8, UTF-16, UTF-32 같은 인코딩 방식 사용

임의의 문자열 데이터 생성

- 방법 1: nltk 라이브러리 corpus(말뭉치) 사용

```
import random
import nltk                # NLTK :: Natural Language Toolkit

nltk.download('words')    # 단어 리스트 다운로드 (처음 실행 시에만 필요)

from nltk.corpus import words

def generate_random_words(n=10000):
    word_list = words.words()
    # 너무 긴 단어를 제외하고 짧고 일반적인 단어 위주로 선택 (예: 3~8자)
    filtered_words = [w for w in word_list if 3 <= len(w) <= 8 and w.isalpha()]
    return random.sample(filtered_words, n)

# 실행
alphabet_words = generate_random_words()
print(alphabet_words)
# alphabet_words = ['braiding', 'bervie', 'uptable', 'awane', 'Cradock']
```

임의의 문자열 데이터 생성

- 방법 2: 한글 초성 × 중성 × 종성의 조합을 생성

'''

유니코드 : 시작위치 (0xAC00:가) , 초성 (19개) , 중성 (21개) , 종성 (28개)

ex: '강' 의 구성: ㄱ (초성) + ㅏ (중성) + ㅇ (종성)

인덱스: 초성 'ㄱ' = 0,
 중성 'ㅏ' = 0,
 종성 'ㅇ' = 0

code = 0xAC00 + (0 * 21 * 28) + (0 * 28) + 21
 = 0xAC00 + 21
 = 0xAC15 → chr(0xAC15) == '강'

'''

임의의 문자열 데이터 생성

- 방법 3: 자주 쓰이는 한국어 낱말(의미 있는 한글 단어 사용)

https://ko.wiktionary.org/wiki/부록:자주_쓰이는_한국어_낱말_5800

실습문제 : 한글 단어 리스트 만들기

- 앞에서 사용한 "자주 쓰이는 한국어 낱말 5800" 사이트에서 데이터 가져와 단어 5000개를 리스트에 지정하세요.
 - 1) 사이트에서 한글 단어 복사하기
 - 2) 메모장을 열어 복사하고 `korean_words.txt`로 저장
 - 3) 코드에서 `txt` 파일 읽어오기
 - 4) 임의의 1000개 리스트 `korean_words` 변수에 지정하기

실습문제 : 문자열 데이터 정렬하기

- 앞에서 만든 아래 데이터를 test_case로 만들어 정렬해 보세요..

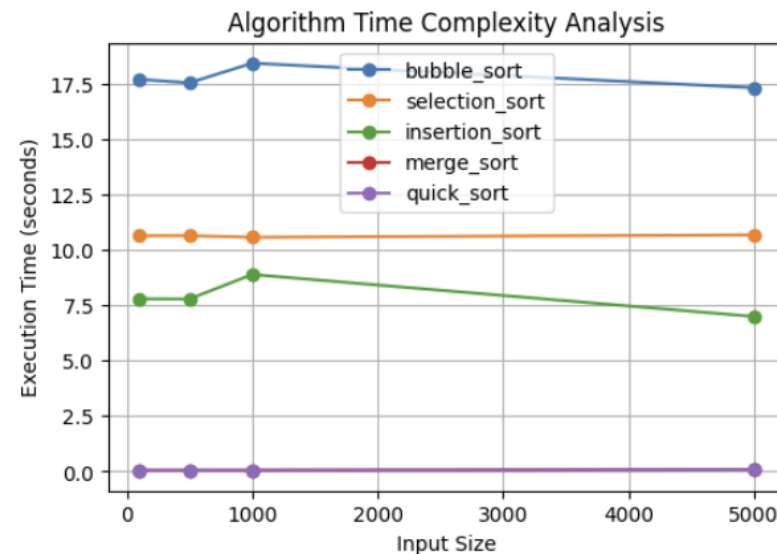
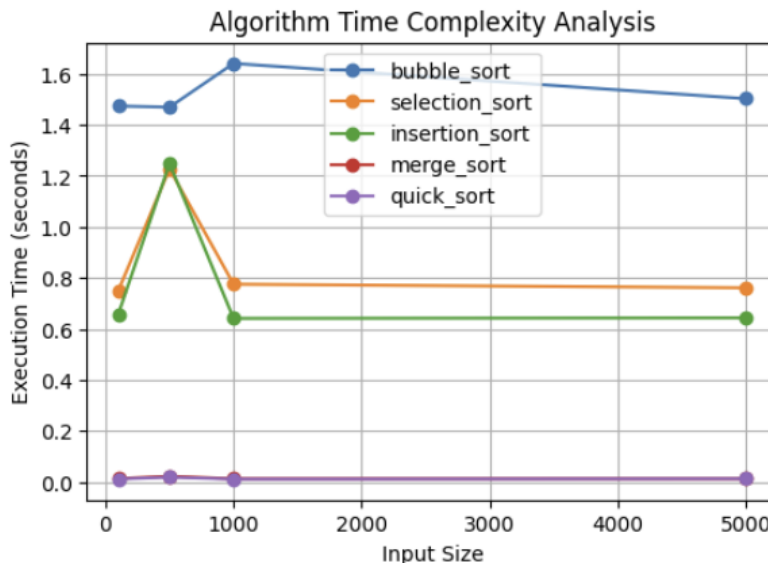
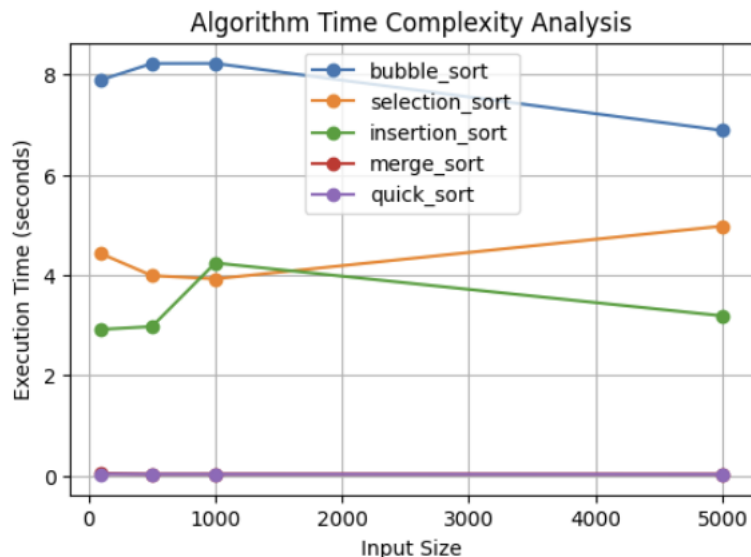
```
test_cases = [ (alphabet_words, 'alphabet_words'),  
               (korean_words, 'korean_words'),  
               (mixed_words, 'mixed_words') ]
```

실습문제 : 다량의 문자열 데이터 시간 성능 측정하고 시각화하기

- 앞에서 만든 아래 데이터를 test_case로 만들어 정렬해 보세요.

1) 데이터 크기 : `sizes = [100, 500, 1000, 5000]`

➤ `alphabet_words`, `korean_words`, `mixed_words`



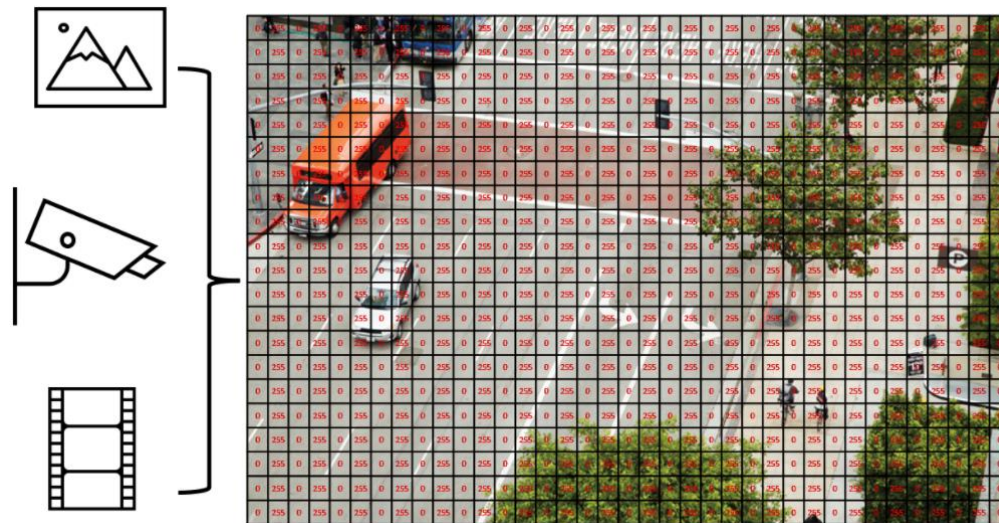
이미지 데이터 정렬

이미지 데이터는 어떤 것을 기준으로 정렬할까?

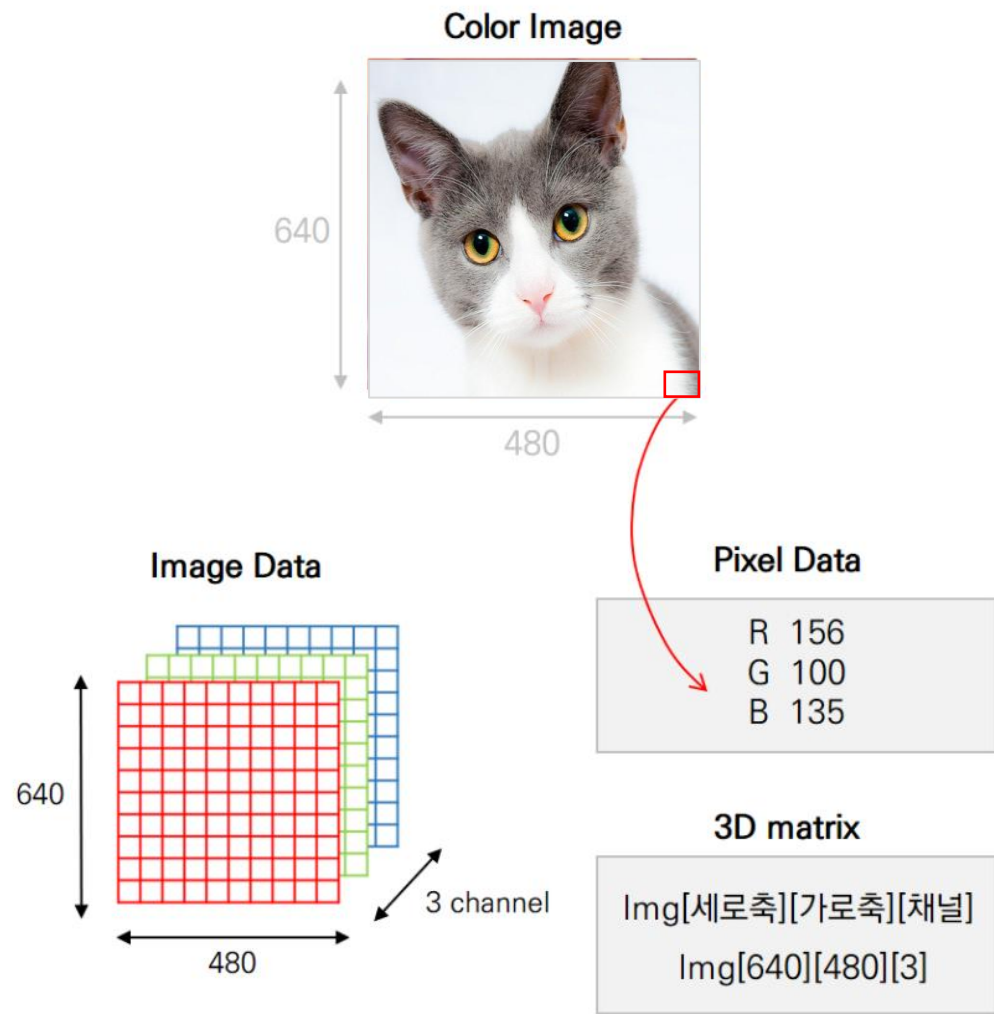
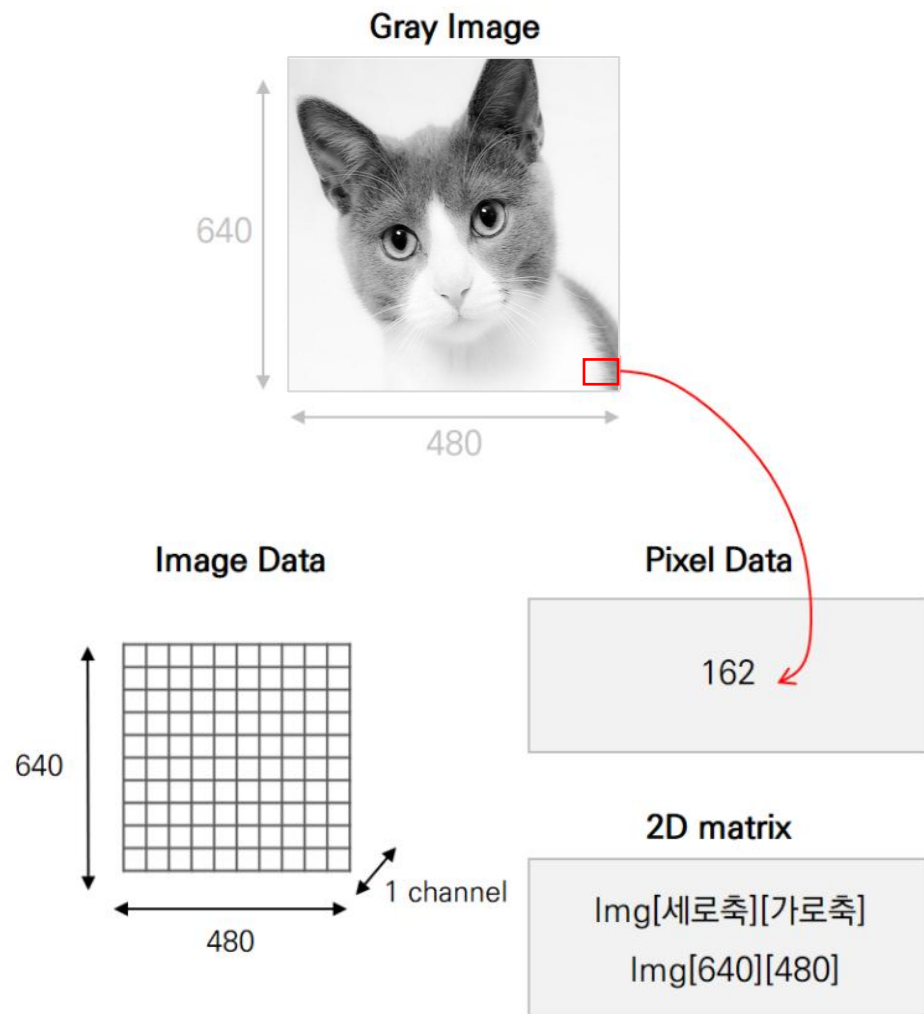
이미지 데이터

■ 픽셀(Pixel)이란?

- 이미지(영상)를 구성하는 가장 작은 단위 (화소)
- 하나의 픽셀은 색상 정보를 담고 있는 데이터 단위
- 픽셀의 크기는 컬러 표현 방식(비트 덱스, 비트 깊이)에 따라 달라진다



이미지 데이터



※본 강의 자료는 본인 학습용으로만 사용 가능하며 무단 복제/배포를 금지합니다.

이미지 데이터

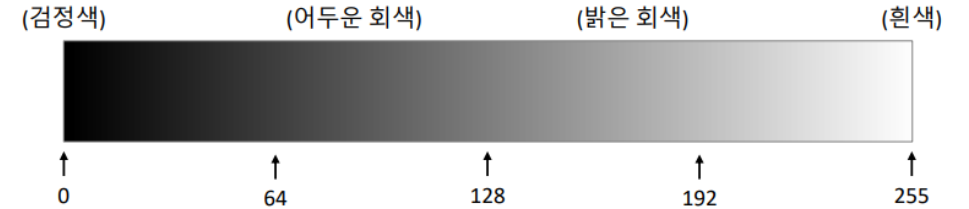
■ Grayscale

- 밝기 성분을 0 ~ 255 범위의 정수로 표현
- 1 Byte 사용 (2^8 만큼 표현 : 0~255)



187	187	187	194	197	173	77	25	19	19
190	187	190	191	158	37	15	14	20	20
187	182	180	127	32	16	13	16	14	12
184	186	172	100	20	13	15	18	13	18
186	190	187	127	18	14	15	14	12	10
189	192	192	148	16	15	11	10	10	9
192	195	181	37	13	10	10	10	10	10
189	194	54	14	11	10	10	10	9	8
189	194	19	16	11	11	10	10	9	9
192	88	12	11	11	10	10	10	9	9

픽셀



이미지 데이터

- 컬러 모드별 픽셀 크기 예 :

컬러 모드	설명	비트 수	바이트 수
1-bit (흑백)	0 or 1 (흑/백만)	1비트	1/8 바이트
L (Grayscale)	밝기만 표현 (0~255)	8비트	1 바이트
RGB	"R, G, B 각 채널이 8비트"	24비트	3 바이트
RGBA	RGB + Alpha(투명도)	32비트	4 바이트
CMYK	"인쇄용 색상(Cyan, Magenta, Yellow, Black)"	32비트	4 바이트
16-bit grayscale	고해상도 흑백 (0~65535)	16비트	2 바이트
HDR 이미지 (예: float32 RGB)	고정밀 이미지 표현	96비트	12 바이트

이미지 데이터

■ 이미지 데이터 확인

```
from PIL import Image
import numpy as np

img = Image.open("cat.jpg")
print("모드:", img.mode) # 예: 'RGB'
print("해상도:", img.size) # (width, height)

# 픽셀 수
width, height = img.size
pixels = width * height

# 총 바이트 수 (RGB 기준)
total_bytes = pixels * 3 # RGB는 픽셀당 3바이트
print(f"총 바이트 수 (추정): {total_bytes:,} Bytes")
```


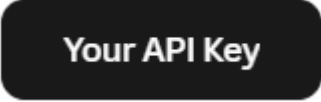


모드: RGB
해상도: (640, 425)
총 바이트 수 (추정): 816,000 Bytes

실습문제 : 이미지 데이터 준비하기

- 아래와 같이 정렬을 위한 다량의 이미지 데이터를 준비하세요

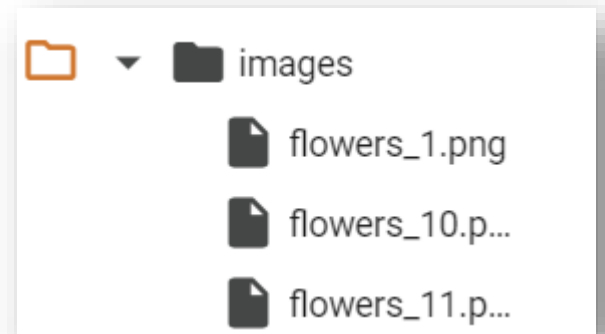
1) <https://www.pexels.com/api/> 회원가입

2)  "Get Started" 클릭 후 →  API Key 복사
(배포된 코드에 입력하기)

3) 자신의 API Key 입력

4) SEARCH_QUERY 입력 후 실행 : `cats(jpg)`, `dogs(jpg)`, `cars(gif)`, `flowers(png)`

코랩에서 코드를 실행하면
images 폴더에 해당 이미지가
다운로드 된 것을 확인할 수 있음



실습문제 : 이미지 데이터 속성 추출하기

- 앞에서 준비한 이미지 파일의 속성을 추출하여 표로 나타내 보세요.

	파일명	파일크기(Bytes)	해상도	모드	평균밝기	R평균	G평균	B평균
0	flowers_51.png	55727	940x627	RGB	179.04	199.23	176.53	139.04
1	flowers_82.png	47010	940x627	RGB	178.75	185.23	176.30	174.38
2	flowers_9.png	53233	940x628	RGB	182.94	232.29	159.73	172.72
3	flowers_64.png	24822	433x650	RGB	131.60	187.99	104.75	121.85
4	flowers_31.png	131671	940x613	RGB	107.15	107.74	107.78	102.40
...
95	flowers_7.png	141548	867x650	RGB	96.77	127.52	89.46	53.84
96	flowers_46.png	33353	433x650	RGB	92.44	97.66	90.83	87.03
97	flowers_69.png	50843	867x650	RGB	116.66	120.53	121.84	79.76
98	flowers_61.png	48535	940x627	RGB	216.47	223.01	213.54	214.26

실습문제 : 이미지 속성별 정렬하기

- 이미지의 밝기값 기준으로 정렬해 보세요. (배포코드 참고)

Original 1



Grayscale 28.5



Original 2



Grayscale 54.1



Original 3



Grayscale 79.1



[Quiz]

[Quiz1]

- 동일한 기계에서 삽입 정렬과 병합 정렬의 구현 결과를 비교한다고 가정하자. n 개의 입력에 대해 삽입 정렬은 $8n^2$ 번을, 병합 정렬은 $64n\log n$ 번을 계산하고 각각 종료한다. n 값이 얼마일 때까지 삽입 정렬이 병합 정렬보다 빠를까? (단, 파이썬에서 $\log n$ 은 \log_2 로 계산)

[Quiz2]

- 동일한 기계에서 수행 시간이 $100n^2$ 인 알고리즘이 수행 시간이 2^n 인 알고리즘보다 빨라지는 n 의 최솟값은 얼마인가?

[Quiz3]

- 배열 [33,19,20,15,13,10,2,13,16,12]는 최대힙(Max-heap)인가?

[Quiz4]

- 은행은 주로 계좌의 거래 내역을 거래 시간 순으로 기록하지만 많은 사람이 수표가 번호순으로 되어 있는 고객 내역서를 받고 싶어한다. 사람들은 보통 수표를 수표 번호순으로 사용하고 상인들은 비교적 신속하게 이를 현금화한다. 따라서 거래 시간 순서를 수표 번호 순서로 바꾸는 문제는 거의 정렬된 입력을 정렬하는 문제다. Insertion sort가 Quick sort보다 이 문제를 잘 풀지를 논하라.

[Quiz5]

- 다음 영어 단어 목록을 Radix-sort 정렬 알고리즘으로 정렬하여라.

단어 목록 : COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB,
BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX

Q & A

Next Topic

- 탐색 알고리즘

Keep learning, see you soon!