정렬 알고리즘

01. 선택 정렬

```
import random

def selection_sort(list, n) :
    for i in range(0, n-1) :
        least = i                       # 교환 될 위치 지정
        for j in range(i+1, n) :        # 교환 될 위치보다 작은 값 탐색
            if list[j] < list[least] :
                least = j
        list[i], list[least] = list[least], list[i]

list = []

for i in range(0, 20) :
    list.append(random.randrange(1,100))

selection_sort(list, len(list))

print(list)
```

02. 삽입 정렬

```
import random

list = []
for i in range(0, 20) :
    list.append(random.randrange(1,100))

def insertion_sort(list, n) :
    for i in range(1, n) :
        key = list[i]
        # 키의 이전 위치부터 처음 위치까지 키 값보다 작은 값 탐색
        for j in range(i-1, -1, -1) :
            if list[j] <= key :
                break
            # 현재 위치의 값이 키 값보다 크면 다음 위치의 값에 현재 위치 값 복사
            list[j+1] = list[j]

        if list[j] < key :      # 탐색된 위치의 값이 키 값보다 작으면
            list[j+1] = key
        else:
            list[j] = key
```

```
insertion_sort(list, len(list))

print(list)
```

03. 버블 정렬

```
import random
list = []

for i in range(0, 20) :
    list.append(random.randrange(1,100))

def bubble_sort(list, n) :
    for i in range(n-1, -1, -1) :
        for j in range(0, i) :
            if list[j] > list[j+1] :
                list[j], list[j+1] = list[j+1], list[j]

bubble_sort(list, len(list))

print(list)
```

04. 합병 정렬

```
import random
list = []
sorted = []

for i in range(0, 20) :
    list.append(random.randrange(1,100))

for i in range(0, 20) :
    sorted.append(0)

def merge(list, left, mid, right, sorted) :
    i = left        # 중앙을 기준으로 분할된 왼쪽 리스트에 대한 인덱스
    j = mid + 1     # 중앙을 기준으로 분할된 오른쪽 리스트에 대한 인덱스
    k = left        # 정렬되어 저장되는 리스트에 대한 인덱스

    # 분할 정렬된 list의 합병
    while i <= mid and j <= right :
        if list[i] <= list[j] :
            sorted[k] = list[i]
            k += 1
            i += 1
        else:
```

```
            sorted[k] = list[j]
            k += 1
            j += 1

    # 합병을 한 후 정렬되지 않은 나머지 값들 후처리
    if i > mid :    # 오른쪽 리스트가 남은 경우
        for l in range(j, right +1) :
            sorted[k] = list[l]
            k += 1
    else :          # 왼쪽 리스트가 남은 경우
        for l in range(i, mid +1) :
            sorted[k] = list[l]
            k += 1

    # 리스트에 정렬된 리스트 복사
    list[left:right+1] = sorted[left:right+1]


def merge_sort(list, left, right, sorted) :
    if left < right :
        mid = (left + right) // 2
        merge_sort(list, left, mid, sorted)         # 왼쪽 리스트 정렬
        merge_sort(list, mid + 1, right, sorted)    # 오른쪽 리스트 정렬
        merge(list, left, mid, right, sorted)       # 병합

merge_sort(list, 0, 19, sorted)

print(list)
```

05. 퀵 정렬

```
import random
list = []

for i in range(0, 20) :
    list.append(random.randrange(1,100))

def quick_sort(list, left, right) :
    if left < right :
        q = partition(list, left, right)    # 피봇 결정 후 정렬한 뒤 피봇 위치 반환
        quick_sort(list, left, q - 1)       # 피봇 중심 왼쪽 정렬
        quick_sort(list, q + 1, right)      # 피봇 중심 오른쪽 정렬

def partition(list, left, right) :
    low = left
    high = right
    pivot = list[left]

    while low < high :  # 왼쪽이 오른쪽 보다 위치가 작을 때
        # 피봇보다 작은 값(피봇 중심 왼쪽 값)
```

```
        while low <= high and list[low] <= pivot :
            low += 1
        # 피봇보다 큰 값(피봇 중심 오른쪽 값)
        while low <= high and list[high] >= pivot :
            high -= 1
        if low < high :
            list[low], list[high] = list[high], list[low]
    # 중간 위치랑 피봇이랑 위치 바꿈
    list[left], list[high] = list[high], list[left]
    return high

quick_sort(list, 0, 19)

print(list)
```