1번 문제: 교재의 소스를 수정하여 실행하게 해보세요.

5번 문제: 알고리즘 표현을 글이나 언어(파이썬,C등)로 작성하세요.

8번 문제

11번 문제

23번 문제

**1. 리스트 A, L, G, O, R, I, T, H, M을 선택 정렬을 이용해 오름차순으로 정렬하라.**

def selection\_sort(A) :                 # 선택 정렬

    n = len(A)                          # n: 리스트 A의 길이

    for i in range(n-1) :                   # 0, 1, 2, ... n-2 [외부루프]

        least = i

        for j in range(i+1, n) :            # i+1,...,n-1 [내부루프]

            if (A[j]<A[least]) :            # 비교연산

                least = j                   # 최소항목 갱신

        A[i], A[least] = A[least], A[i]     # 배열 항목 교환

        printStep(A, i + 1);                # 중간 과정 출력용 문장

def printStep(arr, val) :                   # 중간 과정 출력용 함수

    print("  Step %2d = " % val, end='')    # 현재 단계를 출력

    print(arr)                          # 리스트의 내용을 출력

data = [ 'A', 'L', 'G', 'O', 'R', 'I', 'T', 'H', 'M' ]

print("Original  : ", data)

selection\_sort(data)

print("Selection : ", data)

Original : ['A', 'L', 'G', 'O', 'R', 'I', 'T', 'H', 'M']

Step 1 = ['A', 'L', 'G', 'O', 'R', 'I', 'T', 'H', 'M']

Step 2 = ['A', 'G', 'L', 'O', 'R', 'I', 'T', 'H', 'M']

Step 3 = ['A', 'G', 'H', 'O', 'R', 'I', 'T', 'L', 'M']

Step 4 = ['A', 'G', 'H', 'I', 'R', 'O', 'T', 'L', 'M']

Step 5 = ['A', 'G', 'H', 'I', 'L', 'O', 'T', 'R', 'M']

Step 6 = ['A', 'G', 'H', 'I', 'L', 'M', 'T', 'R', 'O']

Step 7 = ['A', 'G', 'H', 'I', 'L', 'M', 'O', 'R', 'T']

Step 8 = ['A', 'G', 'H', 'I', 'L', 'M', 'O', 'R', 'T']

Selection : ['A', 'G', 'H', 'I', 'L', 'M', 'O', 'R', 'T']

**5. n개의 검은 구슬과 n개의 흰 구슬이 무작위로 일렬로 나열되어 있다. 이 구슬들을 흰 구슬이 모두 먼저 나오고 이후에 검은 구슬에 나오도록 정렬하려고 한다. 이때, 구슬의 이동은 인접한 두 구슬을 교환하는 것만 가능하다. 이 문제를 해결하기 위한 알고리즘을 작성하라.**

def bubble\_sort(arr):

    for i in range(len(arr)-1, 0, -1):

        for j in range(i):

            if arr[j] == '흑' and arr[j+1] == '백':

                print(arr)

                arr[j], arr[j+1] = arr[j+1], arr[j]

    return arr

A = ['흑', '백', '백', '흑', '흑', '흑', '백', '흑', '백', '백']

print(bubble\_sort(A))

**8. 100개의 0으로 구성된 이진 텍스트(binary text)에서 다음의 패턴을 찾으려고 한다. 알고리즘 3.3과 같은 억지 기법을 사용한다면 각각 몇 번의 비교 연산이 필요한가?**

(1) 96\*5 480 (2) 96\*1 96 (3) 96\*4 384

**11. 입력으로 주어진 문자열에서 A로 시작하고 B로 끝나는 부분 문자열(substring)의 개수를 구하는 문제가 주어졌다. 예를 들어, 문자열 ADBAAEDBA에는 4개의 부분 문자열이 있다. ADBAAEDBA, ADBAAEDBA, ADBAAEDBA, ADBAAEDBA이다.**

(1) 이 문제에 대한 억지 기법 알고리즘을 설계하고, 시간 복잡도를 계산하라.

def count\_substr(str, A, B):

    count = 0

    n = len(str)

    for i in range(n-1):

        if str[i] == 'A':

            for j in range(i+1, n):

                if str[j] == 'B':

                    count += 1

    return count

A = "ADBAAEDBA"

print(count\_substr(A, 'A', 'B'))

* O(n^2)

(2) 이 문제에 대한 더 효율적인 알고리즘을 찾아보라.

def count\_substr2(str, A, B):

    count = 0

    sum = 0

    n = len(str)

    for i in range(n):

        if str[i] == 'A':

            count += 1

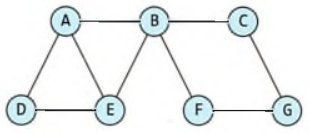
        elif str[i] == 'B':

            sum += count

    return sum

* O(n)

**23. 다음과 같은 그래프가 주어졌다. 물음에 답하라.**



**(1) 이 그래프를 인접 행렬로 표현하라.**

인접 행렬 = [ [0, 1, 0, 1, 1, 0, 0],

[1, 0, 1, 0, 1, 1, 0],

[0, 1, 0, 0, 0, 0, 1],

[1, 0, 0, 0, 1, 0, 0],

[1, 1, 0, 1, 0, 0, 0],

[0, 1, 0, 0, 0, 0, 1],

[0, 0, 1, 0, 0, 1, 0] ]

**(2) 이 그래프를 인접 리스트로 표현하라.**

인접 리스트 = { ‘A’: {‘B’, ‘D’, ‘E’},

‘B’: {‘A’, ‘C’, ‘E’, ‘F’},

‘C’: {‘B’, ‘G’},

‘D’: {‘A’, ‘E’},

‘E’: {A’, ‘B’, ‘D’},

‘F’: {‘B’, ‘G’},

‘G’: {‘C’, ‘F’} }

**(3) 정점 A를 출발 정점으로 하고 깊이 우선 방식으로 탐색하였을 때 방문하는 정점을 순서대로 나열하다. 단, 인접한 정점들 중에서는 알파벳으로 먼저 나오는 정점을 먼저 탐색한다고 가정하라.**

A - B – C – G – F – E - D

**(4) 위 문제에서 더 이상 방문하지 않은 인접 정점이 없어 첫 번째로 되돌아가는 정점을 찾아라.**

F

**(5) 정점 A를 출발 정점으로 하고 너비 우선 방식으로 탐색하였을 때 방문하는 정점을 순서대로 나열하라. 단, 인접한 정점들 중에서는 알파벳으로 먼저 나오는 정점을 큐에 먼저 넣는다고 가정하라.**

A – B – D – E – C – F - G