(1) 순차 탐색 알고리즘

--------------------------------------------------------------------

sequentialSearch(a[], search\_key, n)

i ← 0;

while (i < n and a[i].key ≠ search\_key) do i ← i + 1;

if (i = n) then return –1;

else return i;

end sequentialSearch()

--------------------------------------------------------------------

□ 파이썬 소스 코드

--------------------------------------------------------------------

class node:

def \_\_init\_\_(self, key=None):

self.key = key

class Dict:

def \_\_init\_\_(self):

Dict.a = []

def search(self, search\_key, n):

i = 0

while i < n and Dict.a[i].key != search\_key:

i += 1

if i == n:

return -1

else:

return i

def insert(self, v):

Dict.a.append(node(v))

N = 8

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

print('키 리스트: ', key)

print()

d = Dict()

for i in range(N):

d.insert(key[i])

s\_key = int(input('탐색 키 (종료시 999) : '))

while s\_key != 999:

res = d.search(s\_key, N)

if res == -1:

print('탐색 키 없음')

else: print('%d 번째 원소'%(res+1))

print()

s\_key = int(input('탐색 키 (종료시 999) : '))

--------------------------------------------------------------------

(2) 이진 탐색 알고리즘

--------------------------------------------------------------------

binarySearch(a[], search\_key, n)

left ← 0; right ← n - 1;

while (left ≤ right) do {

mid ← (left + right) / 2;

if (a[mid].key == search\_key) then return mid;

if (a[mid].key > search\_key) then right ← mid – 1;

else left ← mid + 1;

}

return -1;

end binarySearch()

--------------------------------------------------------------------

□ 파이썬 소스 코드

--------------------------------------------------------------------

class node:

def \_\_init\_\_(self, key=None):

self.key = key

class Dict:

def \_\_init\_\_(self):

Dict.a = []

def search(self, search\_key, n):

left = 0

right = n - 1

while right >= left:

mid = (left + right) // 2

if Dict.a[mid].key == search\_key:

return mid

if Dict.a[mid].key > search\_key:

right = mid - 1

else:

left = mid + 1

return -1

def insert(self, v):

Dict.a.append(node(v))

N = 8

key = [1, 3, 5, 7, 9, 11, 13, 15]

print('키 리스트: ', key)

print()

d = Dict()

for i in range(N):

d.insert(key[i])

s\_key = int(input('탐색 키 (종료시 999) : '))

while s\_key != 999:

res = d.search(s\_key, N)

if res == -1:

print('탐색 키 없음')

else: print('%d 번째 원소'%(res+1))

print()

s\_key = int(input('탐색 키 (종료시 999) : '))

--------------------------------------------------------------------

(3) 이진 트리 탐색 알고리즘

--------------------------------------------------------------------

binaryTreeSearch(T, search\_key)

x ← T;

while (x ≠ null) do {

if (x.key = search\_key) then return x.key;

if (x.key > search\_key) then x ← x.left;

else x ← x.right;

}

return -1;

end binaryTreeSearch()

--------------------------------------------------------------------

□ 파이썬 소스 코드

--------------------------------------------------------------------

class node:

def \_\_init\_\_(self, key=None, left=None, right=None):

self.key = key

self.left = left

self.right = right

class Dict:

x = p = node

z = node(key=0, left=0, right=0)

z.left = z

z.right = z

head = node(key=0, left=0, right=z)

def search(self, search\_key):

x = self.head.right

while x != self.z:

if x.key == search\_key:

return x.key

if x.key > search\_key:

x = x.left

else:

x = x.right

return -1

def insert(self, v):

x = p = self.head

while (x != self.z):

p = x

if x.key == v:

return

if x.key > v:

x = x.left

else:

x = x.right

x = node(key=v, left=self.z, right=self.z)

if p.key > v:

p.left = x

else:

p.right = x

def check(self, search\_key):

x = p = self.head.right

while (x != self.z):

if x.key == search\_key:

print('key : ', x.key, ', parents : ', p.key)

p = x

if x.key > search\_key:

x = x.left

else:

x = x.right

N = 8

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

print('키 리스트: ', key)

print()

d = Dict()

for i in range(N):

d.insert(key[i])

s\_key = int(input('탐색 키 (종료시 999) : '))

while s\_key != 999:

res = d.search(s\_key)

if res == -1:

print('탐색 실패')

else: print('탐색 성공')

print()

s\_key = int(input('탐색 키 (종료시 999) : '))

--------------------------------------------------------------------

(4) 이진 탐색 트리에 대한 정확성 검사

□ 파이썬 소스 코드

--------------------------------------------------------------------

class node:

def \_\_init\_\_(self, key=None, left=None, right=None):

self.key = key

self.left = left

self.right = right

class Dict:

x = p = node

z = node(key=0, left=0, right=0)

z.left = z

z.right = z

head = node(key=0, left=0, right=z)

def search(self, search\_key):

x = self.head.right

while x != self.z:

if x.key == search\_key:

return x.key

if x.key > search\_key:

x = x.left

else:

x = x.right

return -1

def insert(self, v):

x = p = self.head

while (x != self.z):

p = x

if x.key == v:

return

if x.key > v:

x = x.left

else:

x = x.right

x = node(key=v, left=self.z, right=self.z)

if p.key > v:

p.left = x

else:

p.right = x

def check(self, search\_key):

x = p = self.head.right

while (x != self.z):

if x.key == search\_key:

print('key : ', x.key, ', parents : ', p.key)

p = x

if x.key > search\_key:

x = x.left

else:

x = x.right

N = 8

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

s\_key = list(range(1, N + 1))

print('키 리스트: ', key)

d = Dict()

for i in range(N):

d.insert(key[i])

for i in range(N):

d.check(s\_key[i])

--------------------------------------------------------------------

**<코딩 테스트 연습 #3> 최소값 찾기**

□ 파이썬 소스 코드 (히프를 사용한 경우)

--------------------------------------------------------------------

import heapq

N, L = map(int, input().split())

A = list(map(int, input().split()))

hq = []

answer = []

for i in range(N):

heapq.heappush(hq, (A[i], i))

while hq[0][1] < i - L + 1:

heapq.heappop(hq)

answer.append(hq[0][0])

print(" ".join(map(str, answer)))

--------------------------------------------------------------------

□ 파이썬 소스 코드 (데크를 사용한 경우)

--------------------------------------------------------------------

from collections import deque

N, L = map(int, input().split())

A = list(map(int, input().split()))

q = deque()

i = 0

while i < N:

while q and q[-1][1] > A[i]:

q.pop()

q.append((i, A[i]))

i += 1

if i - q[0][0] > L:

q.popleft()

print(q[0][1], end=' ')

--------------------------------------------------------------------