**프로그래밍 실습 #7**

2023년 11월 1주차

□ 다음 탐색 알고리즘을 파이썬으로 구현해 보라.

(1) bitskey 클래스

bitskey 클래스에서 비트 단위 연산을 어떻게 수행하는지 확인한다.

def bits(self, k, j):  
 # 알고리즘:  
 # 1. k 만큼 우측 shift 이동 -> k 번째 Bit 가 마지막 비트가 된다.  
 # 2. 11111 을 j 만큼 좌측 shift -> j 만큼 0이 생성.  
 # j = 1 이므로 가장 마지막 비트가 0이 됨.  
 # 3. 11110 을 NOT 연산을 통해 00001 으로 변환.  
 # 4. 00001 과 0000k 를 AND 연산.  
 # k 가 1 이면 1, 0 이면 0을 반환.  
 # 5. 따라서 bits(k, 1) 을 수행 할 시 k번째 비트 값을 알 수 있음.  
 return (self.x >> k) & ~(~0 << j)

(2) 디지털 탐색 트리

Dict 클래스에 check(self, v) 함수를 추가하여 디지털 탐색 트리가 제대로 생성되었는지 검사한다.

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

[1, 19, 5, 18, 3, 26, 9]

key : 1, parents: 1

key : 3, parents: 5

key : 5, parents: 1

key : 9, parents: 5

key : 18, parents: 19

key : 19, parents: 1

key : 26, parents: 19

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

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

maxb = 5

class bitskey:

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

self.x = x

def get(self):

return self.x

def bits(self, k, j):

return (self.x >> k) & ~(~0 << j)

class node:

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

self.key = bitskey(key)

self.left = None

self.right = None

class Dict:

itemMin = bitskey(0)

z = node(itemMin)

head = node(itemMin)

head.left = z

head.right = z

def search(self, v):

v = bitskey(v)

x = self.head.left

b = maxb

self.z.key = v

while v.get() != x.key.get():

b = b - 1

if v.bits(b, 1):

x = x.right

else:

x = x.left

if x == self.z:

return -1

else:

return x.key.get()

def insert(self, v):

v = bitskey(v)

b = maxb-1

x = self.head.left

p = self.head

while x.key.get() != self.z.key.get():

p = x

if v.bits(b, 1):

x = x.right

else:

x = x.left

b -= 1

x = node(self.itemMin)

x.key = v

x.left = self.z

x.right = self.z

if v.bits(b+1, 1):

p.right = x

else:

p.left = x

def check(self, v):

pass

import random, time

N = 7

key = [1, 19, 5, 18, 3, 26, 9]

s\_key = [1, 19, 5, 18, 3, 26, 9]

s\_key.sort()

d = Dict()

for i in range(N):

d.insert(key[i])

print(key)

for i in range(N):

d.check(s\_key[i])

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

(3) 기수 탐색 트라이

Dict 클래스에 check(self, v) 함수를 추가하여 디지털 탐색 트리가 제대로 생성되었는지 검사한다.

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

[1, 19, 5, 18, 3, 26, 9]

1 left left left left

3 left left left right

5 left left right

9 left right

18 right left left right left

19 right left left right right

26 right right

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

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

maxb = 5

class bitskey:

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

self.x = x

def get(self):

return self.x

def bits(self, k, j):

return (self.x >> k) & ~(~0 << j)

class node:

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

if key.get() == 0:

self.key = bitskey(0)

self.external = False

else :

self.key = key

self.external = True

self.left = 0

self.right = 0

class Dict:

itemMin = bitskey(0)

head = 0

head\_check = False

def search(self, v):

v = bitskey(v)

return self.searchR(self.head, v, maxb-1)

def insert(self, v):

v = bitskey(v)

self.insertR(self.head, v, maxb-1)

def insertR(self, h, v, d):

if h == 0:

h = node(v)

if self.head\_check == False:

self.head = h

return h

if h.external:

leaf = node(v)

h = self.split(leaf, h, d)

if self.head\_check == False:

self.head = h

self.head\_check = True

return h

if v.bits(d, 1) == 0:

h.left = self.insertR(h.left, v, d-1)

else:

h.right = self.insertR(h.right, v, d-1)

return h

def split(self, p, q, d):

t = node(self.itemMin)

if ((p.key.bits(d, 1))\*2 + (q.key.bits(d, 1))) == 0:

t.left = self.split(p, q, d-1)

elif ((p.key.bits(d, 1))\*2 + (q.key.bits(d, 1))) == 1:

t.left = p

t.right = q

elif ((p.key.bits(d, 1))\*2 + (q.key.bits(d, 1))) == 2:

t.right = p

t.left = q

elif ((p.key.bits(d, 1))\*2 + (q.key.bits(d, 1))) == 3:

t.right = self.split(p, q, d-1)

return t

def searchR(self, h, v, d):

if h == 0:

return self.itemMin

if v.get() == h.key.get():

return v

if v.bits(d, 1) == 0:

return self.searchR(h.left, v, d-1)

else:

return self.searchR(h.right, v, d-1)

def check(self, v):

pass

import random, time

N = 7

key = [1, 19, 5, 18, 3, 26, 9]

s\_key = [1, 19, 5, 18, 3, 26, 9]

s\_key.sort()

d = Dict()

for i in range(N):

d.insert(key[i])

print(key)

d.head.external = True

for i in range(N):

d.check(s\_key[i])

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

(4) 패트리샤 트리

Dict 클래스에 check(self, v) 함수를 추가하여 패트리샤 트리가 제대로 생성되었는지 검사한다.

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

[1, 19, 5, 18, 3, 26, 9]

key : 1, parents: 3, number: 0

key : 3, parents: 5, number: 1

key : 5, parents: 9, number: 2

key : 9, parents: 19, number: 3

key : 18, parents: 26, number: 0

key : 19, parents: 19, number: 4

key : 26, parents: 19, number: 3

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

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

maxb = 5

class bitskey:

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

self.x = x

def get(self):

return self.x

def bits(self, k, j):

return (self.x >> k) & ~(~0 << j)

class node:

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

self.key = key

self.b = None

self.left = None

self.right = None

class Dict():

itemMin = bitskey(0)

head = node(itemMin)

head.b = maxb

head.left = head.right = head

def search(self, v):

v = bitskey(v)

p = self.head

x = self.head.left

while p.b > x.b:

p = x

if self.bits(v, x.b, 1):

x = x.right

else:

x = x.left

if v.get() != x.key.get(): return self.itemMin

return x.key

def insert(self, v):

v = bitskey(v)

i = maxb

p = self.head

t = self.head.left

while p.b > t.b:

p = t

if self.bits(v, t.b, 1):

t = t.right

else:

t = t.left

if v.get() == t.key.get(): return

while self.bits(t.key, i, 1) == self.bits(v, i, 1):

i -= 1

p = self.head

x = self.head.left

while p.b > x.b and x.b > i:

p = x

if self.bits(v, x.b, 1):

x = x.right

else:

x = x.left

t = node(self.itemMin)

t.key = v

t.b = i

if self.bits(v, t.b, 1):

t.left = x

t.right = t

else:

t.left = t

t.right = x

if self.bits(v, p.b, 1):

p.right = t

else:

p.left = t

def bits(self, item, bit, cmp):

if item.bits(bit, 1) == cmp:

return 1

else:

return 0

def check(self, v):

pass

import random, time

N = 7

key = [1, 19, 5, 18, 3, 26, 9]

s\_key = [1, 19, 5, 18, 3, 26, 9]

#key = [7, 20, 17, 4, 9, 21, 23, 13]

#s\_key = [7, 20, 17, 4, 9, 21, 23, 13]

s\_key.sort()

d = Dict()

for i in range(N):

d.insert(key[i])

print(key)

for i in range(N):

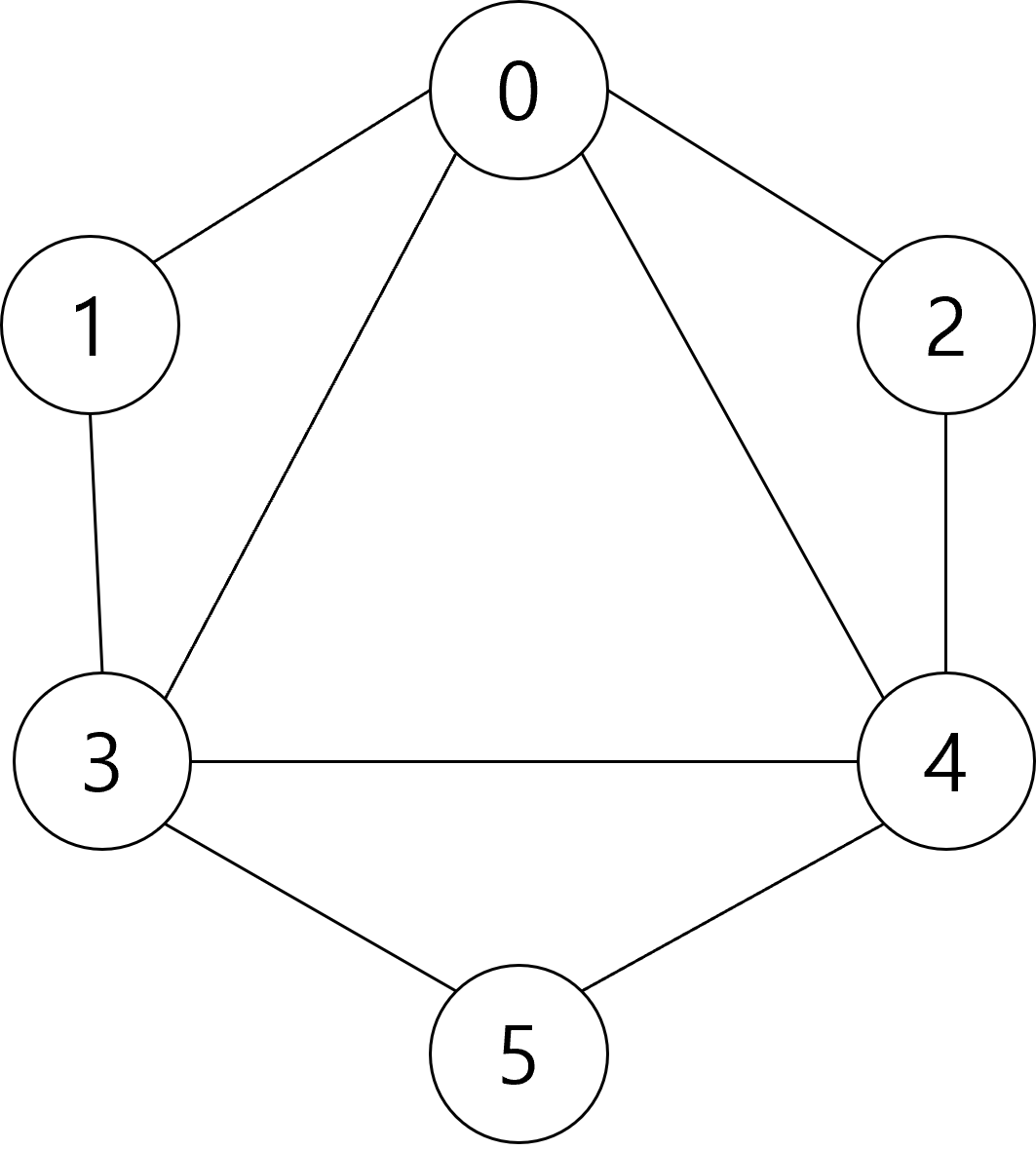
d.check(s\_key[i])

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

□ 그래프 탐색 알고리즘

(1) 깊이 우선 탐색 (Depth First Search: DFS) 알고리즘

다음과 같은 그래프를 파이썬 리스트로 표현하면 다음과 같다.



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

a = [[1, 2, 3, 4, None],

[0, 3, None],

[0, 4, None],

[0, 1, 4, 5, None],

[0, 2, 3, 5, None],

[3, 4, None]]

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

다음과 같은 소스 코드를 사용하여 DFS 알고리즘을 파이썬으로 구현하라.

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

def dfs(v):

n = 6

a = [[1, 2, 3, 4, None], [0, 3, None], [0, 4, None], [0, 1, 4, 5, None], [0, 2, 3, 5, None], [3, 4, None]]

for i in range(n):

visited = [False] \* n

print('dfs(%d) : '%i, end='')

dfs(i)

print()

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

[실행 예]

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

dfs(0) : 0 1 3 4 2 5

dfs(1) : 1 0 2 4 3 5

dfs(2) : 2 0 1 3 4 5

dfs(3) : 3 0 1 2 4 5

dfs(4) : 4 0 1 3 5 2

dfs(5) : 5 3 0 1 2 4

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

(2) 너비 우선 탐색 (Breadth First Search: DFS) 알고리즘

다음과 같은 소스 코드를 사용하여 BFS 알고리즘을 파이썬으로 구현하라. BFS 알고리즘을 구현할 때 queue 라이브러리를 사용하라.

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

def bfs(v):

import queue

q = queue.Queue()

n = 6

a = [[1, 2, 3, 4, None], [0, 3, None], [0, 4, None], [0, 1, 4, 5, None], [0, 2, 3, 5, None], [3, 4, None]]

for i in range(n):

visited = [False] \* n

print('bfs(%d) : '%i, end='')

bfs(i)

print()

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

[실행 예]

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

bfs(0) : 0 1 2 3 4 5

bfs(1) : 1 0 3 2 4 5

bfs(2) : 2 0 4 1 3 5

bfs(3) : 3 0 1 4 5 2

bfs(4) : 4 0 2 3 5 1

bfs(5) : 5 3 4 0 1 2

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