PDF 문서별 주요 파이썬 코드 요약 및 설명 정리

# 📘 관련 문서: DFS(깊이우선탐색) 자료.pdf

## 📄 코드 파일: DFSFirst250522.py — 재귀 기반 깊이 우선 탐색 구현\*\*\*\*

input = sys.stdin.readline # 변수 선언/초기화

n, m= map(int, input().split()) # 사용자 입력

A= [ [] for \_ in range(n+1)] # 변수 선언/초기화

visited=[False]\* (n+1) # 변수 선언/초기화

def DFS(v): # 함수 정의  
 visited[v]=True  
 for i in A[v]:  
 if not visited[i]:  
 DFS(i)  
 for \_ in range(m):  
 s,e = map(int, input().split() )  
 A[s].append(e)  
 A[e].append(s)  
 print('A=',A)  
 count= 0  
 for i in range (1,n+1):  
 if not visited[i]:  
 count+=1  
 DFS(i)  
 print("연결요소의 갯수=",count)

## 📄 코드 파일: DFS TEst.py — DFS 실행 테스트 및 결과 출력\*\*\*\*\*\*\*

def go\_dfs(g, cur, check): # 함수 정의  
 check[cur] = True  
 print(cur, end=' ')  
 for nxt in sorted(g[cur]):  
 if not check[nxt]:  
 go\_dfs(g, nxt, check)

def go\_bfs(g, start, check): # 함수 정의  
 q = [start]  
 check[start] = True  
 idx = 0 # pop(0)을 대체  
 while idx < len(q):  
 cur = q[idx]  
 idx += 1  
 print(cur, end=' ')  
 for nxt in sorted(g[cur]):  
 if not check[nxt]:  
 q.append(nxt)  
 check[nxt] = True  
 n, m, s = map(int, input().split())  
 g = {i: [] for i in range(1, n + 1)}  
 for \_ in range(m):  
 u, v = map(int, input().split())  
 g[u].append(v)  
 g[v].append(u)  
 v1 = [False] \* (n + 1)  
 go\_dfs(g, s, v1)  
 print()  
 v2 = [False] \* (n + 1)  
 go\_bfs(g, s, v2)

# 📘 관련 문서: 너비우선 탐색\_설명(250522).pdf

## 📄 코드 파일: Connected component.py — BFS 기반 연결 요소 탐색

def dfs(node, visited, graph): # 함수 정의  
 visited[node] = True  
 for neighbor in graph[node]:  
 if not visited[neighbor]:  
 dfs(neighbor, visited, graph)  
 # 입력  
 n, m = map(int, input().split())  
 graph = [[] for \_ in range(n + 1)]  
 visited = [False] \* (n + 1)  
 for \_ in range(m):  
 u, v = map(int, input().split())  
 graph[u].append(v)  
 graph[v].append(u)  
 # 연결 요소 개수 계산  
 count = 0  
 for i in range(1, n + 1):  
 if not visited[i]:  
 dfs(i, visited, graph)  
 count += 1  
 print(count)

# 📘 관련 문서: 이진탐색 설명1(250529).pdf / 순차탐색과 이진탐색.pdf / ch08\_이진 트리\_True.pdf

## 📄 코드 파일: bstFirst0529.py — 기초 이진 탐색 트리 구현 및 탐색

def binSearch(ary,fData): # 함수 정의  
 global count  
 start =0  
 end = len(ary)-1  
 while (start<= end):  
 count+=1  
 mid=(start+end)//2  
 if fData == ary[mid]:  
 return mid  
 elif fData>ary[mid]:  
 start=mid+1  
 elif fData<ary[mid]:  
 end=mid-1  
 ##전역변수 선언  
 dataAry, sortedAry=[],[]  
 findData=0  
 count=0  
 num=[]  
 keyNum=0  
 findDataIndex=0  
 #main부분  
 if \_\_name\_\_=="\_\_main\_\_":  
 print("정수 10개를 공백으로 구분해 입력:")  
 for i in range(10):  
 print("%d번째 정수 입력" % (i+1))  
 val=int(input(""))  
 num.append(val)  
 print("입력된 정수들=",num)  
 keyNum=int(input("검색하고자 하는 키 값 입력="))  
 findDataIndex=binSearch(num,keyNum)  
 print("검색된 키값의 index=%d"% findDataIndex)  
 print("검색된 키값=%d"%num[findDataIndex])

## 📄 코드 파일: bstTreeSecond0529.py — 이진 탐색 트리 삽입/삭제 등 연산 포함

class TreeNdoe: # 클래스 정의

def \_\_init\_\_(self): # 함수 정의  
 self.left=None  
 self.data = None  
 self.right=None  
 #전역변수 선언  
 memory=[]  
 root = None  
 nameAry=['블랙핑크','레드벨벳','미미미누','에이핑크','걸스데이','트와이스']  
 #main  
 if \_\_name\_\_=="\_\_main\_\_":  
 node=TreeNdoe()  
 node.data=nameAry[0]  
 root=node  
 memory.append(node)  
 for name in nameAry[1: ]:  
 node=TreeNode()  
 node.data=name  
 current=root  
 while True:  
 if name<current.data:S

## 📄 코드 파일: bstTreeSecond0605.py — 이진 트리 최종 개선 버전

class TreeNode: # 클래스 정의

def \_\_init\_\_(self): # 함수 정의  
 self.left=None  
 self.data = None  
 self.right=None  
 #전역변수 선언  
 memory=[]  
 root = None  
 nameAry=['블랙핑크','레드벨벳','미미미누','에이핑크','걸스데이','트와이스']

def inorder\_traversal(node): # 함수 정의  
 if node is not None:  
 inorder\_traversal(node.left)#왼쪽 서브트리 방문  
 print(node.data, end="")  
 inorder\_traversal(node.right)  
 #main  
 if \_\_name\_\_=="\_\_main\_\_":  
 node=TreeNode()  
 node.data=nameAry[0]  
 root=node  
 memory.append(node)  
 for name in nameAry[1: ]:  
 node=TreeNode()  
 node.data=name  
 current=root  
 while True:  
 if name<current.data:  
 if current.left==None:  
 current.left=node  
 break  
 current=current.left  
 else:  
 if current.right==None:  
 current.right=node  
 break  
 current=current.right  
 memory.append(node)  
 print("이진 탐색 트리 구성완료")  
 print("memory 리스트에 저장된 노드 데이터드")  
 for node in memory:  
 print(node.data)  
 print()  
 print("이진탐색트리의 모든 노드값 출력(중위순):")  
 inorder\_traversal(root)

29번문제 # 입력

N = int(input()) # 수의 개수 저장

A = list(map(int, input().split())) # A 리스트 저장

A.sort() # 정렬

M = int(input()) # 탐색할 수 개수 저장

target\_list = list(map(int, input().split())) # 탐색할 수 리스트 저장

# 이진 탐색 함수

def binary\_search(target, data):

start = 0 # 시작 인덱스

end = len(data) - 1 # 종료 인덱스

while start <= end:

midi = (start + end) // 2 # 중간 인덱스

midv = data[midi] # 중앙값

if midv > target:

end = midi - 1

elif midv < target:

start = midi + 1

else:

return True # 찾았음

return False # 못 찾음

# 탐색

for target in target\_list:

if binary\_search(target, A):

print(1)

else:

print(0)