Đề cương môn thuật toán và ứng dụng cho lớp CNTT1 & CNTT3 K59

Thời gian 90 phút

Yêu cầu ngôn ngữ lập trình Python:

- list và các thao tác trên list: tạo danh sách, lát cắt, sum, map, zip, filter, reduce ,sort, chặt nhị phân bisect, enumerate, product
- hàm, lambda, namedtuple, lớp
- Cấu trúc dữ liệu: stack, queue, heapq, PriorityQueue, deque
- Thuật toán: Quay lui, tham lam, quy hoạch động, chia để trị, BFS, DFS

Yêu cầu bài làm:

- Nêu ý tưởng thuật toán, nêu công thức (nếu có) không cần chứng minh
- Tính toán bằng tay ví dụ minh họa
- Code bằng Python

Cấu trúc Đề thi gồm 4 câu:

Trong đó 3 câu nằm trong nội dung dưới đây được 9 điểm
Một câu không có trong đề cương chiếm 1 điểm

Nội dung: Các bài trên trang laptrinhonline.club

1. Cấu trúc dữ liệu queue, stack, heapq, PriorityQueue áp dụng cho thuật toán BFS và DFS
   - Dãy con đơn điệu tăng dài nhất

```
from bisect import bisect_left as chat
if __name__ == '__main__':
    n=input()
    a=list(map(int,input().split())) #nhap day tren 1 dong
    A=[]
    for x in a:
        if A==[] or A[-1]<x: A.append(x)
        else:A[chat(A,x)]=x
    print(len(A))
```

   - Trinh thám

```
from collections import deque
if __name__ == '__main__':
    n,k=map(int,input().split())
    a=list(map(int,input().split()))
    d=deque([])
    for i,x in enumerate(a):
        if i<k-1:
            while len(d) and d[-1][0]<x: d.pop()
            d.append((x,i))
        else:
```

```
            while len(d) and d[-1][0]<x: d.pop()
            d.append((x,i))
            while i-d[0][1]+1>k: d.popleft()
            print(d[0][0],end=' ')
```

- In tất cả các đường đi trong bài mọi con đường về 0

```
from math import floor,sqrt
class zero:
    def __init__(self,v):
        self.n=v
        self.p=[0]*v+[1]
    def DFS(self,s):
        for a in range (1,floor(sqrt(s))+1):
            if s%a==0:
                t=(a-1)*(s//a+1)
                if self.p[t]==0:
                    self.p[t]=1
                    self.DFS(t)
    def sol(self):
        self.DFS(self.n)
        for i in range (0,self.n+1):
            if self.p[i]: print(i,end=" ")

if __name__ == '__main__':
    n=int(input("n = "))
    Z=zero(n)
    Z.sol()
```

- Ốc sên: Cho mê cung là các giá trị 0,1 nhập vị trí ốc sên tại một ô chứa số 0 hỏi nó đi được bao nhiêu ô chứa số 0 mà mỗi bước chỉ có 4 láng giềng

```
class ocsen:
    m=[]
    count=0
    def __init__(self,fname):
        f=open(fname,"r")
        for line in f.read().split("\n"):
            a=list(map(int,line.split()))
            self.m.append([1]+a+[1])
        f.close()
        n=len(self.m[0])
        m=[1]*n+self.m+[1]*n
        #print(*self.m,sep="\n")
    def dfs(self,u,v):
        self.m[u][v]=1
        self.count+=1
        hh=[1,-1,0,0]
        hc=[0,0,1,-1]
        for z,t in zip(hh,hc):
            if self.m[u+z][v+t]==0: self.dfs(u+z,v+t)
    def sol(self):
        self.count=0
```

```
        u,v=map(int,input("toa do oc sen : ").split())
        self.dfs(u,v)
        print("So o di duoc ",self.count)
```

- Liệt kê các xâu con

```
D={}
def enuma(s):
    D[s]=1
    if len(s)>1 :
        if s[:-1] not in D.keys():enuma(s[:-1])
        if s[1:] not in D.keys():enuma(s[1:])

if __name__ == '__main__':
    s=input()
    enuma(s)
    k=[x for x in D.keys()]
    k=sorted(k)
    print(*k,sep="\n")
```

2. Thuật toán quay lui
    - Phương trình nghiệm nguyên #phuong trinh x1+..+xn=M

```
dem=0
def TRY(x,k,T,n,M):
    global dem
    if k==n-1:
        x.append(M-T)
        dem = dem +1
        print(dem,M," = ",end=" ")
        print(*x,sep="+")
    else:
        for t in range(M-T+1): TRY(x+[t],k+1,T+t,n,M)

if __name__ == '__main__':
    TRY([],0,0,5,10)
```

    - Đổi tiền

```
class money:
    def nhap(self):
        self.a=list(map(int,input().split()))
        self.M=int(input())
        self.n=len(self.a)
        self.res = self.M+1
    def TRY(self,x,k,T):  #gia su da co tu x0...x[k-1]
        if len(x)==self.n-1:
            if (self.M-T)%self.a[-1]==0:
                t=(self.M-T)//self.a[-1]
```

```
                k=sum(x)+t
                if self.res>k: self.res=k
                print(*(x+[t]),sep=" ")
        else:
            for t in range((self.M-
T)//self.a[k]+1):self.TRY(x+[t],k+1,T+t*self.a[k])
    def sol(self):
        self.nhap()
        self.TRY([],0,0)
        if self.res==self.M+1 : print("khong doi duoc")
        else:print("so to tien it nhat la ",self.res)
if __name__ == '__main__':
    mon=money()
    mon.sol()
```

- Tám hậu

```
def ve(x,n):
    for i,u in enumerate(x): print('-'*(u-1)+'*'+'-'*(n-u))
    print("\n")

class Hau:
    def nhap(self):
        self.n=int(input())
        self.A={}    #danh dau duong cheo song song duong cheo chinh
        self.B={}    #danh dau duong cheo song song duong cheo phu
        self.C={}
    def TRY(self,x,k):
        if len(x)==self.n: ve(x,self.n)# print(*x,sep=" ")
        else:
            for t in range (1,self.n+1):
                if (t not in self.C.keys() or self.C[t]==False) \
                        and (t-k not in self.A.keys() or self.A[t-
k]==False) \
                        and (t+k not in self.B.keys() or
self.B[t+k]==False):
                    self.C[t]=self.A[t-k]=self.B[t+k]=True
                    self.TRY(x+[t],k+1)
                    self.C[t] = self.A[t - k] = self.B[t + k] = False

if __name__ == '__main__':
    H=Hau()
    H.nhap()
    H.TRY([],1)
```

- Hoán vị lặp

```
class hvl:
    dem=0
    def TRY(self,x,n):
        if len(x)==n:
            self.dem+=1
            print(self.dem,"".join(x))
```

```
        for t in self.D.keys():
            if self.D[t]>0:
                self.D[t]-=1
                self.TRY(x+[t],n)
                self.D[t]+=1
    def sol(self):
        xau=input()
        self.D={}
        for t in xau:
            if t not in self.D.keys(): self.D[t]=1
            else:self.D[t]+=1
        self.TRY([],len(xau))
if __name__ == '__main__':
    H=hvl()
    H.sol()
```

-

3. Thuật toán tham lam

- Nối thanh kim loại

```
from queue import PriorityQueue
if __name__ == '__main__':
    Q=PriorityQueue()
    for x in list(map(int,input().split())): Q.put(x)
    res=0
    while Q.qsize()>1:
        u=Q.get()
        u+=Q.get()
        res+=u
        Q.put(u)
    print(res)
```

- Lập lịch (tổ chức sự kiện)

```
class event:
    def __init__(self,u=0,v=0):
        self.start,self.finish=u,v

if __name__ == '__main__':
    n=int(input())
    E=[]
    for _  in range(n):
        u,v=map(int,input().split())
        E.append(event(u,v))
    E.sort(key= lambda x: x.finish) #sap xep tang dan
    time=E[0].start-1
    counter=0
    for e in E:
        if e.start>time:
            counter+=1
            time=e.finish
    print("The number event is organised ",counter)
```

- Búp bê Nga

```
from queue import Queue
if __name__ == '__main__':
    k,*a=list(map(int,input().split()))
    a.sort(reverse=True)
    Q=Queue()
    for x in a:
        if Q.qsize()>0 and x+k<=Q.queue[0]: Q.get()
        Q.put(x)
    print("So bup be",Q.qsize())
```

- Thuật toán Kruskal

```
class edge:
    def __init__(self,L,R,W):
        self.u,self.v,self.w=L,R,W
    def __str__(self):
        return "("+str(self.u)+"->"+str(self.v)+" : "+str(self.w)+")"

class Graph:
    def Read(self):
        f=open("g.txt","r")
        self.n,self.m=map(int,f.readline().split())
        self.A=[]
        for _ in range(self.m):
            u,v,w=map(int,f.readline().split())
            self.A.append(edge(u,v,w))
        f.close()
        self.D=[0]*(self.n+5)
    def getroot(self,x):
        if self.D[x]==0: return x
        return self.getroot(self.D[x])
    def Kruskal(self):
        self.A.sort(key=lambda x:x.w)
        res=0
        print("Chon cac canh : ")
        for e in self.A:
            x=self.getroot(e.u)
            y = self.getroot(e.v)
            if x!=y:
                res+=e.w
                self.D[x]=y
                print(e)
        print("Trong so nho nhat ",res)
    def Print(self):
        for i in range(self.m): print(self.A[i])
if __name__ == '__main__':
    G=Graph()
    G.Read()
    G.Kruskal()
```

- Làm bóng tuyết

```
from queue import PriorityQueue
if __name__ == '__main__':
    n=int(input())
    V=[int(x) for x in input().split()]
    T=[int(x) for x in input().split()]
    Q=PriorityQueue()
    z=0
    for v,t in zip(V,T):
        Q.put(z+v)
        s=0
        while Q.qsize() and Q.queue[0]-z<=t: s+=Q.get()-z
        s+=Q.qsize()*t
        z= z+t if Q.qsize() else 0
        print(s, end=" ")
```

- Giao hàng

```
class elem:
    def __init__(self,v): self.value=v
    def __lt__(self, other): return self.value>other.value
from queue import PriorityQueue
if __name__ == '__main__':
    Q=PriorityQueue()
    n=int(input())
    a=[]
    for i in range (int(1e6)+1): a.append([])
    for _ in range(n):
        t,v=map(int,input().split())
        a[t].append(v)
    res=0
    for i in range(1000000,0,-1):
        for x in a[i]: Q.put(elem(x))
        if(Q.qsize()): res+=Q.get().value
    print(res)
```

4. Quy hoạch động
- Cánh cửa thần kỳ

```
from queue import Queue
if __name__ == '__main__':
    a =
['dangdungcntt','tienquanutc','quang123','maianh','nguyenminhduc2820']
    for _ in range(int(input())):
        n = int(input())
        Q = Queue()
        for i in [0,1,2,3,4]: Q.put((i,1))
        while n>Q.queue[0][1]:
```

```
            x = Q.get()
            n-=x[1]
            Q.put((x[0],x[1]*2))
        print(a[Q.get()[0]])
```

- Đổi tiền

```
class money:
    C,k=[],[]
    def thuan(self):
        self.a=[0]+[int(x) for x in input("menh gia : ").split()]
        self.M=int(input("so tien muon doi : "))
        for _ in range(len(self.a)): self.C.append([int(1e9)]*(self.M+1))
        self.C[0][0]=0
        for i in range(1,len(self.a)):
            for j in range(0,self.M+1):
                if j<self.a[i]: self.C[i][j]=self.C[i-1][j]
                else: self.C[i][j]=min(self.C[i-1][j],1+self.C[i][j-
self.a[i]])
        if self.C[-1][self.M]==int(1e9): print("Khong doi duoc")
        else:
            print("So to it nhat ",self.C[-1][self.M])
            self.nghich(len(self.a)-1,self.M)
            print(self.M,"=",end=" ")
            print(*self.k[::-1],sep="+")
    def nghich(self,u,v):
        if(self.C[u][v]==0): return
        while self.C[u][v]==self.C[u-1][v]: u-=1
        self.k+=[self.a[u]]
        self.nghich(u,v-self.a[u])
if __name__ == '__main__':
    M=money()
    M.thuan()
```