**Đề cương ôn tập Thuật toán và ứng dụng**

Dành cho các lớp CNTT1K62, CNTT5K62, CNTT6K62

I. Thời gian 60 phút

II. Hình thức thi viết không sử dụng tài liệu gồm 3 câu

III. Yêu cầu mỗi bài phân tích thuật toán, ví dụ minh họa và code bằng ngôn ngữ Python

IV. Nội dung

1. **Dãy con liên tục khác biệt (Dictionary)**

Cho dãy n phần tử a1, a2, … an tìm một dãy con liên tục nhiều phần tử nhất có thể sao cho trong dãy con đó không có bất kỳ 2 phần tử nào bằng nhau

**Code**

if \_\_name\_\_ == '\_\_main\_\_':

    n = input()

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

    p=0

    res=0

    D={}

    for i,x in enumerate(a):

        if x in D.keys() and D[x]>=p: p = D[x]+1

        res = max(res, i-p)

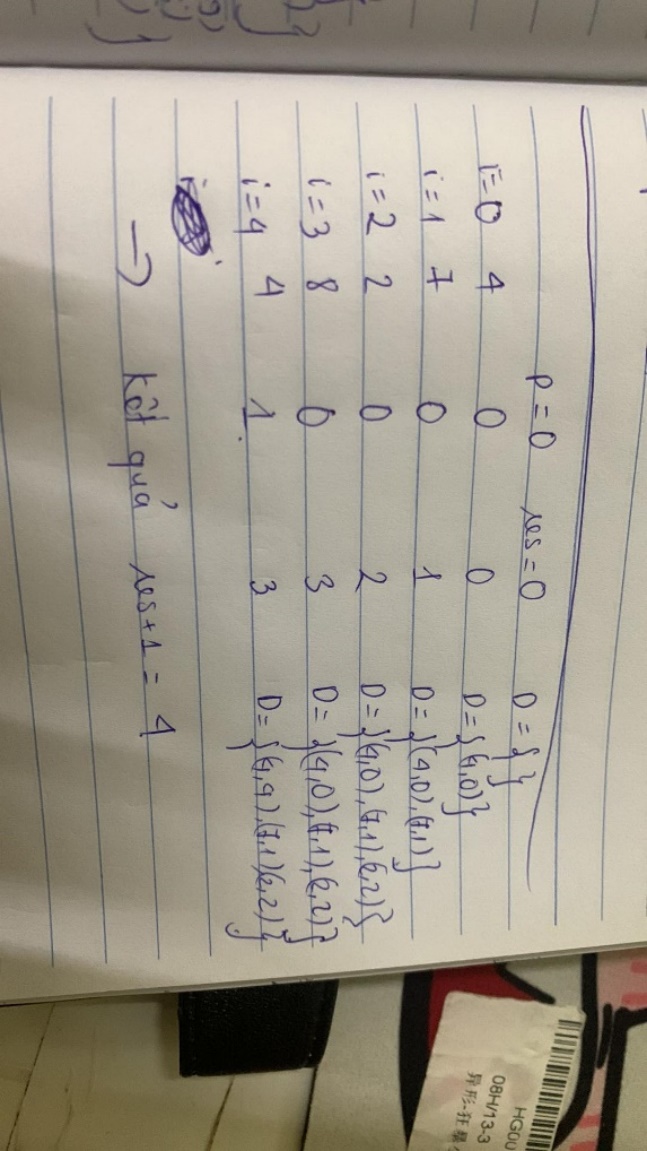
        D[x]=i

    print(res+1)

* **VD**

**N= 5**

**A={ 4, 7, 2, 8, 4}**



1. **Cổ vũ (Dictionary)**

Trong một cuộc thi tài có 2 đối thủ mặc đai xanh và đai đỏ thi đấu với nhau, để cổ vũ cho hai đối thủ thì các cổ động viên đứng xếp thành một hàng dọc cổ vũ cho đối thủ nào thì tay cầm cờ mầu đai của đối thủ đó. Tichpx đi cổ vũ thi đấu thì gặp một bạn cựu sinh viên ngày xưa học thuật toán do Tichpx dạy. Tichpx đố bạn đó tìm một đoạn dài nhất liên tục các cổ động viên sao cho số cổ vũ cho đối thủ đai xanh bằng số cổ vũ cho đối thủ đai đỏ. Bạn hãy giúp bạn cựu sinh viên đó nhé. Bài toán đặt ra là nhập vào một xâu ký tự gồm những ký tự X biểu thị cho mầu Xanh và D biểu thị cho mầu Đỏ chỉ ra độ dài đoạn dài nhất có số ký tự X bằng số ký tự D

* **Code**

if \_\_name\_\_ == '\_\_main\_\_':

    D={0:0}

    res=0

    T=0

    a = input()

    for i,x in enumerate(a,1):

        T+= (1 if x=='X' else -1)

        if T in D.keys(): res = max(res,i-D[T])

        else: D[T] = i

    print(res)

* **VD:**

1. **Quân tượng thú vị (Dictionary)**

Một bàn cờ vua có kích thước 1000x1000, các hàng được đánh số từ 1 đến 1000 lần lượt từ trên xuống dưới, các cột được đánh số từ 1 đến 1000 lần lượt từ trái sang phải, quân tượng là quân có thể di chuyển theo đường chéo và ăn các quân cờ khác. Giả sử trên bàn cờ có n quân tượng, nhiệm vụ của bạn là hãy đếm số quân tượng có thể ăn nhau trên bàn cờ (Hai quân tượng có thể ăn nhau kể cả giữa chúng có một quân tượng khác).

* **Phân tích thuật toán:**
* **Code**

import collections

if \_\_name\_\_ == '\_\_main\_\_':

    C={}

    P={}

    A=[]

    B=[]

    n = int(input())

    for i in range(n):

        x,y = map(int,input().split())

        A.append(x-y)

        B.append(x+y)

    C= collections.Counter(A)

    P= collections.Counter(B)

    res =0

    for v in C.values(): res+=v\*(v-1)//2

    for v in P.values(): res+=v\*(v-1)//2

    print(res)

* **VD**

3

1 1

5 5

9 9

1. **Chăn bò (Dictionary)**

Toto nhận được nhiệm vụ trông nom bò sữa nhà mình đang ăn cỏ trên cánh đồng.

Giả sử cánh đồng là một lưới tọa độ, các con bò đang ở các tọa độ nguyên trên cánh đồng đó và Toto thì đang ở gốc tọa độ. Nếu hai con bò bất kỳ mà thằng hàng (hoặc trùng nhau) cùng phía so với Toto thì Toto chỉ nhìn thấy con ở phía trước mà không nhìn thấy con ở phía sau bị che lấp. Giả sử mắt của Toto rất tốt nếu không có vật cản thì con bò ở xa mấy cũng nhìn thấy.

Nhập vào tọa độ của n con bò, hỏi Toto nhìn thấy bao nhiêu con.

* **Phân tích thuật toán:**
* **Code**

import math

if \_\_name\_\_ == '\_\_main\_\_':

    D={}

    n=int(input())

    for i in range(n):

        x,y = map(int,input().split())

        d = abs(math.gcd(x,y))

        x//=d

        y//=d

        if x\*2000+y not in D.keys():D[x\*2000+y] = True

    print(len(D))

* **VD:**

6

3 3

7 7

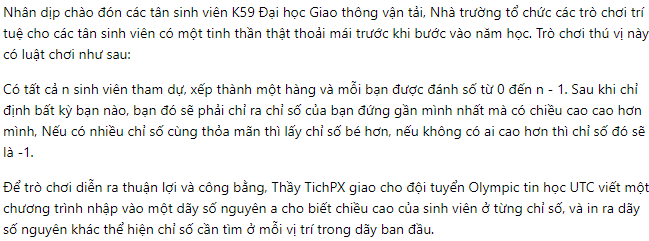
4 6

-1 -1

6 9

3 3

1. **Chào đón k59 (stack)**

****

* **Phân tích thuật toán:**
* **Code**

import math

import queue

def Tim(a):

    S = queue.LifoQueue()

    L=[]

    S.put((math.inf,-1))

    for i , x in enumerate(a):

        while S.queue[-1][0] <x : S.get()

        L.append(S.queue[-1][1])

        S.put((x,i))

    return L

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

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

    L = Tim(a)

    R = Tim(a[::-1])

  R = [n-1-i if i>=0 else -1 for i in R][::-1]

    for i in range(n):

        if L[i] == -1 or R[i] == -1: print(L[i] + R[i] + 1, end =" ")

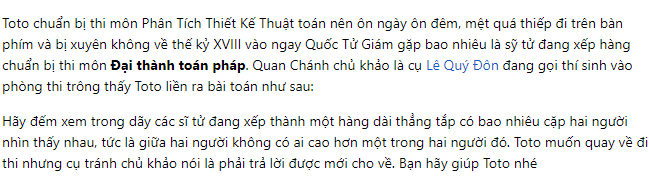
        else :  print(L[i] if i-L[i]<=R[i]-i else R[i], end=" ")

* **VD:**

6

1 4 2 1 7 6

1. **Xếp hàng (stack)**

****

* **Phân tích thuật toán:**
* **Code**

import math

import queue

def Xep(a):

    S= queue.LifoQueue()

    res =0

    for i, x in enumerate(a):

        while not S.empty() and S.queue[-1][0] < x :

            res+= S.get()[1]

        if S.qsize()> 0 and S.queue[-1][0]== x:

             res += S.queue[-1][1] + (S.qsize()>1)

             S.queue[-1][1] +=1

        else :

             res += S.qsize() >0

             S.put([x,1])

    return res

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

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

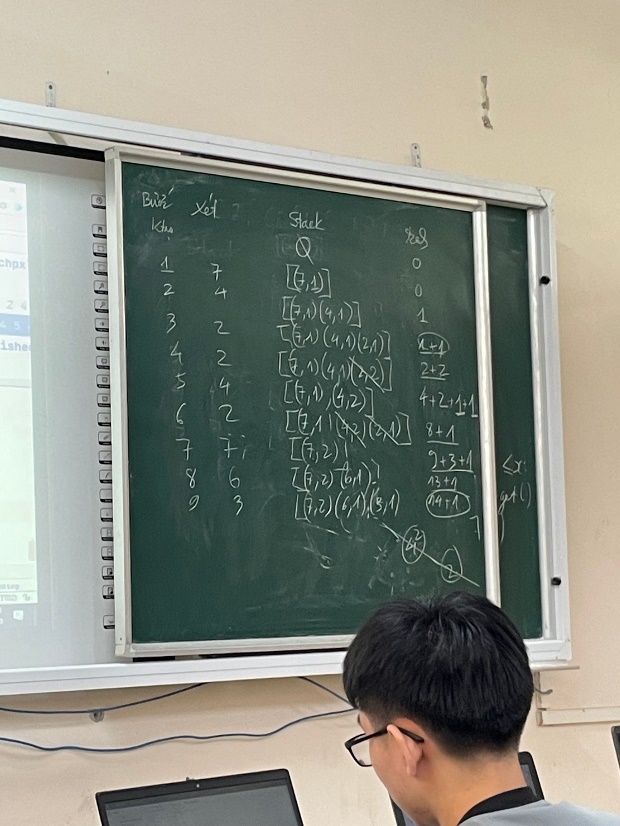
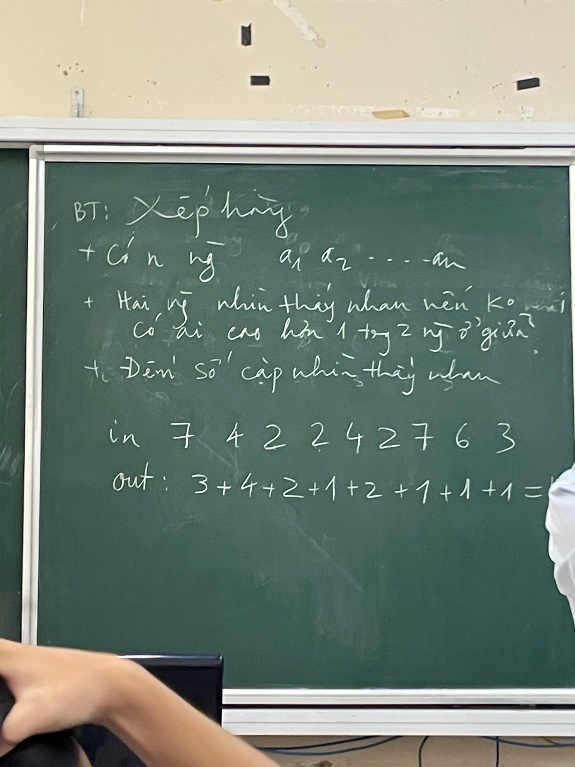
    k= Xep(a)

    print(k)

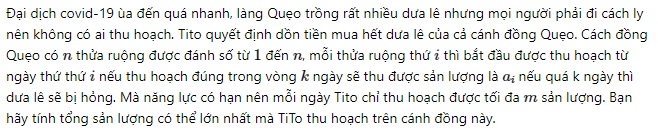
* **VD**

**8**

**4 7 2 8 4 4 4 6**



1. **Buôn dưa lê (queue)**

****

* **Phân tích thuật toán:**
* **Code**

import queue

if \_\_name\_\_ == '\_\_main\_\_':

    n =  int(input())

    k =  int(input())

    m =  int(input())

    a = list(map(int,input().split()))+ [0]\*(k-1)

    res = 0

    Q= queue.Queue()

    for x in a:

        Q.put(x)

        while Q.qsize()>k :

            Q.get()

        t =0

        while Q.qsize()>0 and t+ Q.queue[0]<= m:

            t+= Q.get()

        if(Q.qsize()):

            Q.queue[0]-=m-t

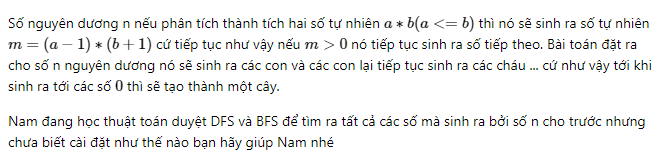
            t=m

        res+=t

    print(res)

* **VD**

1. **Mọi con đường về 0 (BFS-DFS)**

****

* **Phân tích thuật toán:**

**Code**

import queue

def Search(n,s):

    res = {n}

    s.put(n)

    while s.qsize() >0:

        x = s.get()

        m = int(x\*\*0.5) +1

        for a in range(1,m):

            if x%a ==0:

                y = (a-1)\*(x//a+1)

                if y not in res:

                    s.put(y)

                    res.add(y)

    return res

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    S = queue.Queue()

    print(\*Search(n,S))

* **VD**

1. **Nối thanh kim loại (hàng đợi ưu tiên)**

Công việc cơ khí thật là mệt nhọc, muốn nối một thanh kim loại độ dài a với một thanh kim loại độ dài b thì kinh phí để thuê nối tốn mất a+b đơn vị tiền tệ. Hiện nay Tichpx cần nối n thanh kim loại lần lượng có độ dài là a1,a2, ... an thành một đoạn theo bạn Tichpx nên bố trí thế nào để tổng số tiền phải trả là ít nhất

* **Phân tích thuật toán:**
* **Code**

import queue

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

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

    Q=queue.PriorityQueue()

    res =0

    for x in a: Q.put(x)

    while Q.qsize()>= 2:

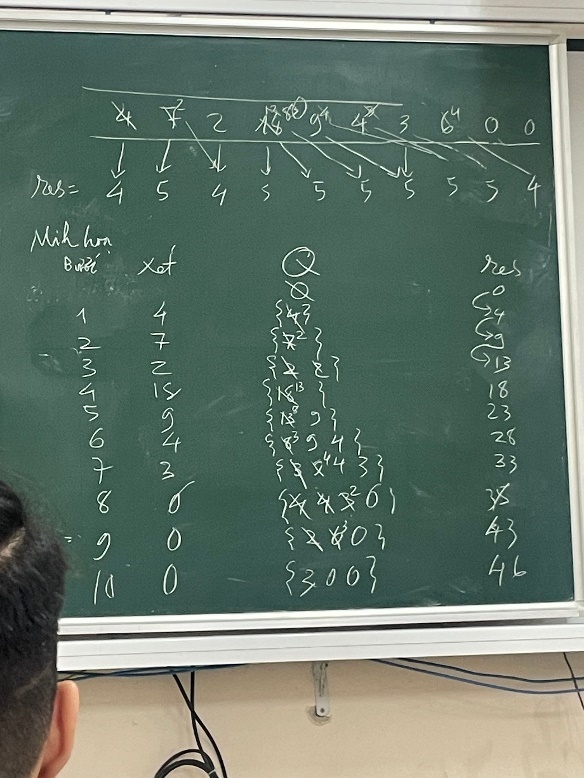
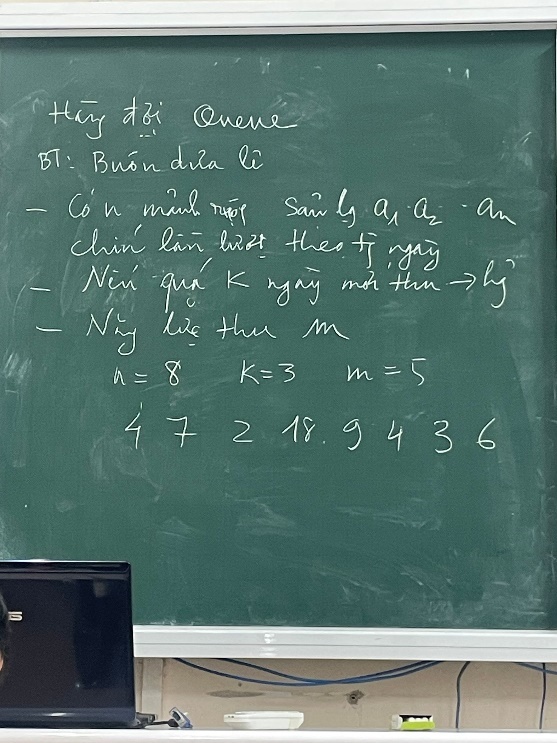
        u = Q.get() + Q.get()

        Q.put(u)

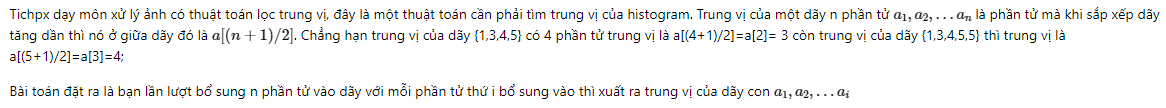
        res += u

    print(res)

* **VD:**



1. **Phần tử trung vị (hàng đợi ưu tiên)**

****

* **Phân tích thuật toán:**
* **Code**

import queue

if \_\_name\_\_ == '\_\_main\_\_':

    n = input()

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

    R=queue.PriorityQueue()

    L= queue.PriorityQueue()

    res =0

    for i,x in enumerate(a,1):

        if i%2==1 :

            L.put(-x)

        else:

            R.put(x)

        if i>= 2 and -L.queue[0] > R.queue[0]:

            L.put(-R.get())

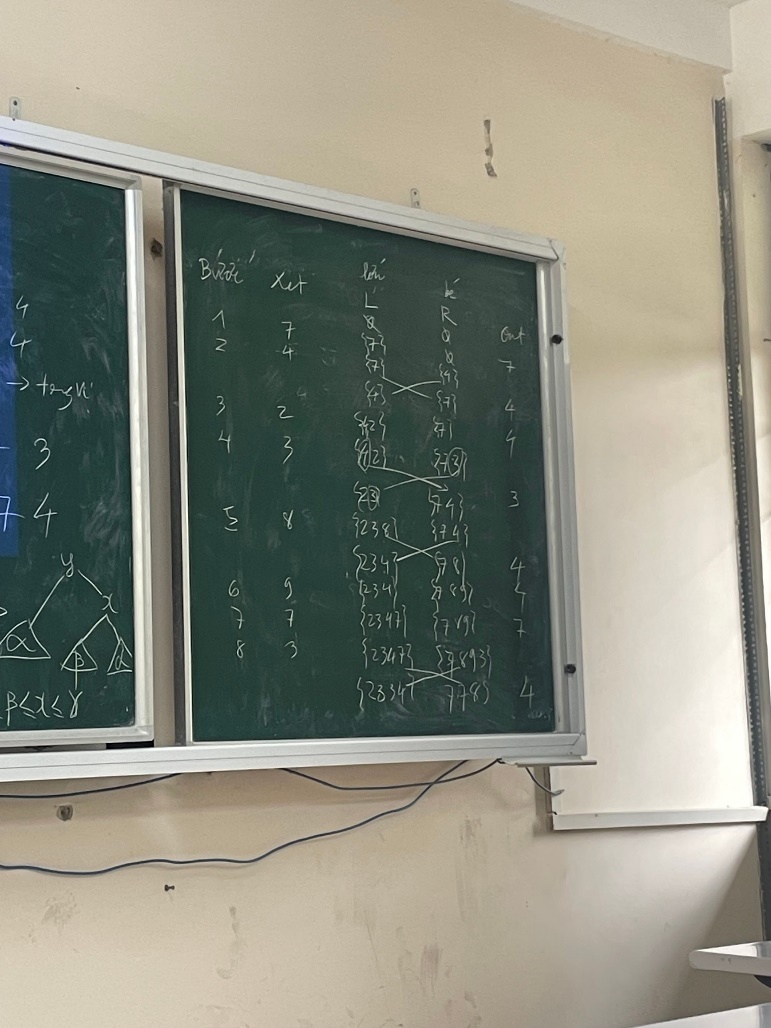
            R.put(-L.get())

        print(-L.queue[0], end=" ")

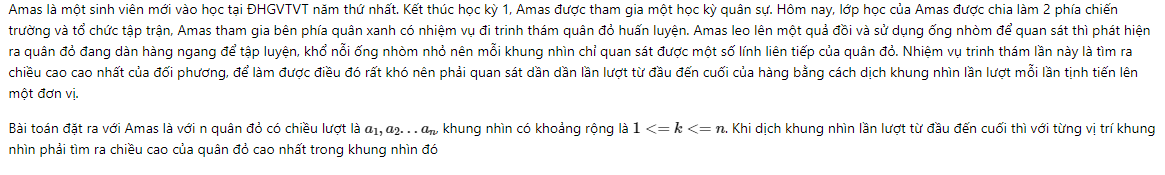
        res = -L.queue[0]

    print(res)

* **VD**



1. **Bài toán trinh thám (hàng đợi ưu tiên)**

****

* **Phân tích thuật toán:**

**Code**

import queue

if \_\_name\_\_ == '\_\_main\_\_':

    n,k = map(int,input().split())

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

    Q=queue.PriorityQueue()

    for i,x in enumerate(a,1):

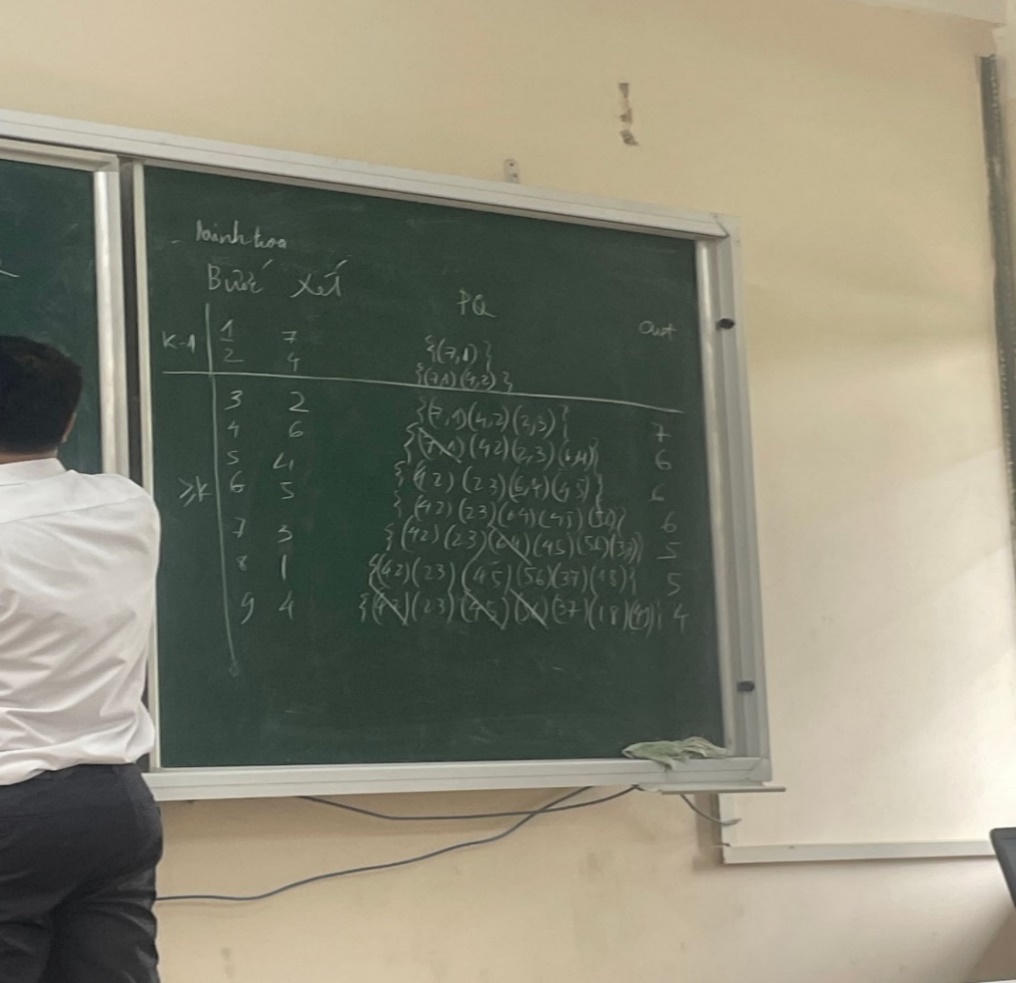
        while Q.qsize() >0 and i-Q.queue[0][1] >= k: Q.get()

        if Q.qsize() > 0 and -Q.queue[0][0]<=x: Q.queue.clear()

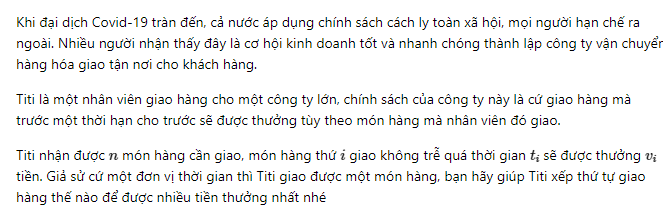
        Q.put((-x,i))

        if i>= k: print(-Q.queue[0][0], end= " ")

* **VD**



1. **Bài toán giao hàng (hàng đợi ưu tiên)**

****

* **Phân tích thuật toán:**
* **Code**

import queue

if \_\_name\_\_ == '\_\_main\_\_':

    A= [[] for i in range(1000005)]

    n= int(input())

    for i in range (n):

        t,v = map(int, input().split())

        A[t].append(v)

    res=0

    Q=queue.PriorityQueue()

    for j in range(1000000, 0,-1):

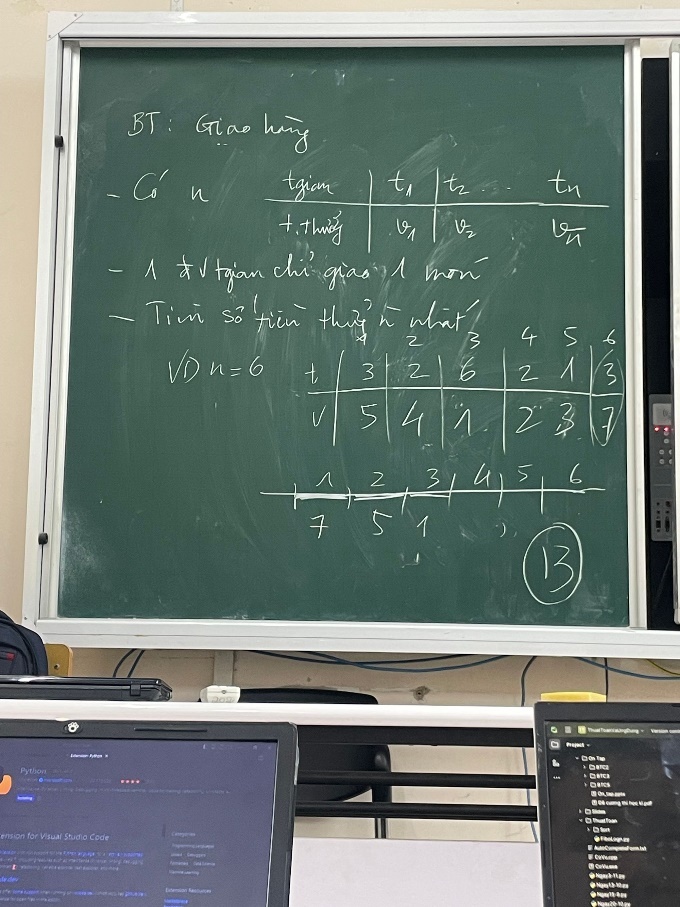
        for x in A[j]: Q.put(-x)

        if Q.qsize():

            res-= Q.get()

    print(res)

* **VD**



1. **Thuật toán Huffman (hàng đợi ưu tiên)**

* **Phân tích thuật toán:**
* **Code**

import queue

import collections

if \_\_name\_\_ == '\_\_main\_\_':

    s = input()

    D= collections.Counter(s)

    Q= queue.PriorityQueue()

    res =0

    for v in D.values():

        Q.put(v)

    while Q.qsize() >=2:

        x= Q.get() + Q.get()

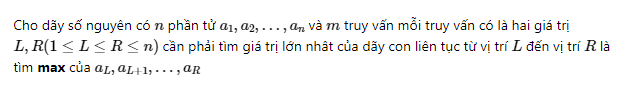
        res+=x

        Q.put(x)

    print(res)

* **VD:**

1. **Truy vấn max (cây IT)**

****

* **Phân tích thuật toán:**
* **Code**

class node:

    def \_\_init\_\_ (self,L,R):

        self.lo,self.hi = L,R

        self.elem =-10\*\*9

        if L+1==R: self.left, self.right=None,None

        else:

            self.left=node(L,(L+R)//2)

            self.right=node((L+R)//2,R)

def update(H,i,x):

    H.elem= max(H.elem,x)

    if H.left!=None:

        update(H.left if i<=H.left.hi else H.right,i,x)

def get(H,L,R):

    if L==H.lo and R==H.hi: return H.elem

    if R<=H.left.hi: return get(H.left,L,R)

    if L>=H.right.lo: return get(H.right,L,R)

    return max(get(H.left,L,H.left.hi), get(H.right,H.right.lo,R))

if \_\_name\_\_ == '\_\_main\_\_':

    n,m = map(int,input().split())

    root= node(1,n+1)

    for i,x in enumerate(map(int,input().split()),1): update(root,i,x)

    for i in range(m):

        L,R=map(int,input().split())

        print(get(root,L,R+1))

* **VD:**