Quay lui

Chia để trị

Quy hoạt động

Tham lam

Giả sử rằng cần sắp xếp mảng A có N phần tử kiểu longint, và sắp xếp theo thứ tự **tăng dần** (khi muốn sắp xếp theo thứ tự **giảm dần**, chỉ cần đổi chiều các dấu so sánh trong các câu lệnh).

1. **Sắp xếp nổi bọt:** thường được dùng khi **N nhỏ** (khoảng vài trăm trở lại)   
   *Duyệt với i từ 1 đến N-1*

*Duyệt với j từ i+1 đến N*

*Nếu A[i] > A[j] thì đổi chỗ A[i], A[j].*

1. **Sắp xếp nhanh:** thường được dùng khi **N lớn** (khoảng vài trăm trở lên)  
   Thuật toán sắp xếp nhanh thường được cài đặt dưới dạng **đệ quy** QuickSort(ll, rr) trong đó ll là chỉ số của phần tử đầu vùng, rr là chỉ số của phần tử cuối vùng.   
   Thuật toán chia để trị:

*Thủ tục QuickSort( ll, rr)*

*Tính x := A[(ll + rr) div 2] {Phần tử ở vị trí chính giữa vùng}*

*Đặt i = ll, j = rr;*

*Lặp lại cho đến khi i > j*

*Tăng dần i để tìm i nhỏ nhất sao cho A[i] > x*

*Giảm dần j để tìm j lớn nhất sao cho A[j] < x*

*if i <= j thì*

*Đổi chỗ A[i], A[j]*

*Tăng i, giảm j*

*Nếu i < rr thì gọi QuickSort(i, rr) {Sắp xếp phần bên phải}*

*Nếu j > ll thì gọi QuickSort(ll, j) {Sắp xếp phần bên trái}*

1. **Đếm phân phối**: thường được dùng khi **N lớn** nhưng **miền giá trị** của các A[i] **không lớn** (khoảng vài trăm trở lại). Ở dạng sắp xếp này, người ta thường lấy ra giá trị của phần tử, kèm theo đó là số lượng phần tử có giá trị đó.

Ví dụ:

12 (5), 15 (7)

{*Có 5 phần tử với giá trị là 12 và 7 phần tử với giá trị là 15*}.  
Thuật toán đếm phân phối gồm các bước sau:

**+ Đếm:**

*Khởi tạo mảng SL với mọi giá trị đều = 0, gtmin = maxlongint, gtmax = 0.*

*Duyệt với i từ 1 đến N*

*SL[A[i]] = SL[A[i]] + 1*

*Nếu gtmax < A[i] thì gtmax = A[i]*

*Nếu gtmin > A[i] thì gtmin = A[i]*

**+ Đưa ra tần xuất xuất hiện các giá trị của mảng A:**

*Duyệt với i từ gtmin đến gtmax*

*Ghi ra i, tiếp theo là SL[i] để trong cặp dấu ngoặc tròn ().*

**+ Phân phối dãy tăng dần:**

*Duyệt với i từ* ***gtmin*** *đến* ***gtmax***

*Trong khi SL[A[i]] > 0 thì:*

*Ghi ra i*

*SL[A[i]] = SL[A[i]] – 1*

**+ Phân phối dãy giảm dần:**

*Duyệt với i từ gtmax đến gtmin*

*Trong khi SL[A[i]] > 0 thì:*

*Ghi ra i*

*SL[A[i]] = SL[A[i]] – 1*

**----------------------------------------------------------------**

**Kết nối số KETNOI**

Cho N số nguyên dương a[1], a[2],..a[N], mỗi số không vượt quá 109. Từ các số nguyên dương đó, người ta tạo ra các số nguyên dương mới bằng cách kết nối (viết liên tiếp nhau) tất cả các số đã cho.

Ví dụ: với N = 4 và các số 12, 34, 567, 890, ta có thể tạo ra các số mới như sau: 1234567890, 3456789012, 8905673412,... Dễ thấy rằng có tất cả 4! = 24 số được tạo ra theo cách đó. Trong 24 số mới tạo thành đó, số 9805673412 là số lớn nhất.

***Yêu cầu:*** Cho biết số N (1<N<=100) và N số a[1], a[2],..a[N], tìm số lớn nhất tạo thành bằng cách kết nối theo cách đã nêu.

***Dữ liệu vào:*** đọc từ tệp **KETNOI.INP** gồm số N, tiếp sau là N số a[1], a[2],..a[N] các số cách nhau bởi dấu cách.

***Dữ liệu ra:*** ghi trong tệp **KETNOI.OUT** ghi độ dài DD của số kết nối lớn nhất tìm được, tiếp đó, nếu DD<=20 thì ghi ra số đó, trái lại, nếu DD > 20 thì ghi ra 10 chữ số đầu, dấu ba chấm (...), và 10 chữ số cuối của SKN, để trong cặp ngoặc vuông.

***Ví dụ:***

|  |  |
| --- | --- |
| **KETNOI.INP** | **KETNOI .OUT** |
| 4 12 34 567 890 | 10 [8905673412] |
| 9 12 34 567 890 321 123 45 54 678 | 23 [8906785675...3432112312] |

**Hướng dẫn:**

* Đọc dữ liệu từ file KETNOI.INP. n, list a
* Số kết nối lớn nhất được tạo thành nhờ các bước sau:
* *Sắp xếp các phần tử của mảng A theo* ***thứ tự giảm dần*** *của các xâu ký tự (theo định nghĩa ai lớn hơn aj khi aiaj lớn hơn ajai).*

*Ví dụ: ‘9’ và ‘90’ vì ‘990’>’909’ thì suy ra ‘9’ lớn hơn xâu ‘90’*

* *Ghép các số nhận được với nhau.*
* Tính độ dài DD.
* Khi độ dài DD của số kết nối lớn hơn 20, để đưa kết quả ra theo yêu cầu, cần làm như sau:

Tạo XauDau chỉ chứa 10 chữ số đầu của số kết nối

* *Khởi tạo s = ‘’, i = 0*
* *Lặp lại khi độ dài của s còn nhỏ hơn 10:  
   i = i + 1  
   s = s + a[i]  
  Nhớ lại 10 chữ số đầu của s vào XauDau.*

Tạo XauCuoi chỉ chứa 10 chữ số cuoi của số kết nối

* *Khởi tạo s = ‘’, i = N+1*
* *Lặp lại khi độ dài của s còn nhỏ hơn 10:*

*i = i – 1*

*s = a[i] + s*

*Nhớ lại 10 chữ số cuối của s vào XauCuoi.*

* Ghi kết quả ra tệp KETNOI.OUT

**Code:**

a = ['12','34','567','890','9','90','999000001']

def qs (d,c):

i = d

j = c

m = a[(i+j)//2]

while i <= j:

while a[i]+m > m+a[i]:#(a[i] > m):

i = i + 1

while a[j]+m < m+a[j]:#(a[j] < m):

j = j - 1

if i <= j:

a[i],a[j] = a[j],a[i]

i = i + 1

j = j - 1

if d < j:

qs(d,j)

if i < c:

qs(i,c)

**Đoạn rời DOANROI**

Cho N đoạn thẳng với các điểm đầu ai và điểm cuối bi là những số nguyên trong khoảng [-1000,1000] , ai<bi. Liệt kế số lượng tối đa K đoạn thẳng không giao nhau. Hai đoạn thẳng [a,b] và [c,d] được coi là không giao nhau nếu chúng trên cùng một trục số, chúng không có điểm chung. Điều kiện này đòi hỏi: b<c hoặc d<a.

***Dữ liệu vào:*** đọc từ tệp **DOANROI.INP:**

* Dòng đầu tiên: số tự nhiên N,1<N≤1000.
* Dòng thứ i trong N dòng tiếp theo, mỗi dòng chứa hai số nguyên ai, bi cách nhau dấu cách, biểu thị điểm đầu và điểm cuối của đoạn thứ i, i=1…N.

***Dữ liệu ra:*** ghi trong tệp **DOANROI.OUT:**

* Dòng đầu tiên: số tự nhiên K.
* K dòng tiếp theo, mỗi dòng một số tự nhiên v thể hiện chỉ số của các đoạn rời nhau tìm được.

***Ví dụ:***

|  |  |
| --- | --- |
| **DOANROI.INP** | **DOANROI.OUT** |
| 8  2 3  4 5  10 12  13 15  1 9  2 5  6 8  7 15 | 5  1  2  7  3  4 |

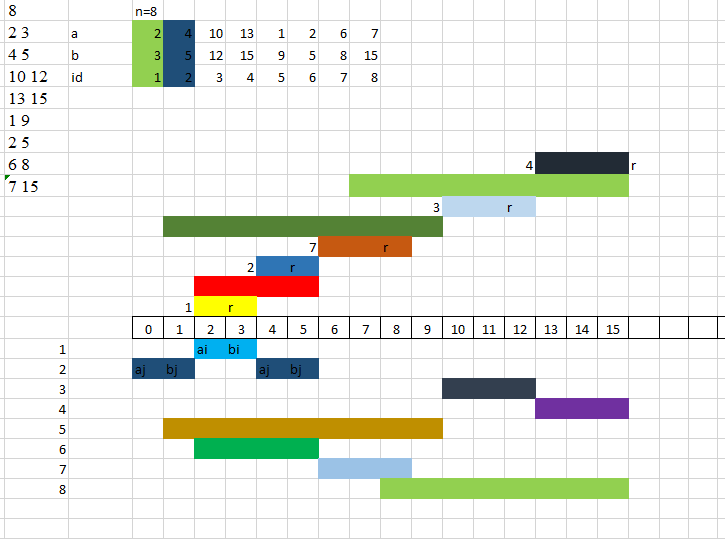
**Hướng dẫn:**

Sử dụng phương pháp tham lam (Greedy Algorithm).

* Đọc dữ liệu từ tệp DOANROI.INP: số n, và list a, list b.
* Tạo list id lưu chỉ số ban đầu của các đoạn thẳng.
* Sắp xếp các đoạn tăng theo đầu phải b, nếu b bằng nhau thì sắp theo a tăng dần.
* Khởi tạo: lấy đoạn 1, đặt r = b1 là đầu phải của đoạn này. k = 0, v = []
* Với mỗi đoạn i: i=2..n tiếp theo ta xét:

Nếu đầu trái của đoạn i, ai > r thì lấy đoạn i đưa vào kết quả (k=k+1,v=v+id[i]) và chỉnh r là đầu phải của đoạn i. r = bi.

* Ghi k và v ra tệp DOANROI.OUT



**Code:**

* Đọc dữ liệu từ tệp DOANROI.INP: số n, và list a, list b.
* Tạo list id lưu chỉ số ban đầu của các đoạn thẳng.

fin = open("DOANROI.INP","r")

fout = open("DOANROI.OUT","w")

li = fin.readline()

n = int(li)

a = []

b = []

id = []

s = 0

for li in fin:

ls = li.split()

a.append(int(ls[0]))

b.append(int(ls[1]))

id.append(s+1)

s += 1

* Sắp xếp các đoạn tăng theo đầu phải b, nếu b bằng nhau thì sắp theo a tăng dần.

def qs (d,c):

i = d

j = c

bm = b[(i+j)//2]

am = a[(i+j)//2]

while i <= j:

while (b[i] < bm) or ((b[i] == bm) and (a[i] < am)):

i = i + 1

while (b[j] > bm) or ((b[i] == bm) and (a[j] > am)):

j = j - 1

if i <= j:

a[i],a[j] = a[j],a[i]

b[i],b[j] = b[j],b[i]

id[i],id[j] = id[j], id[i]

i = i + 1

j = j - 1

if d < j:

qs(d,j)

if i < c:

qs(i,c)

qs(0,n-1)

* Khởi tạo: lấy đoạn 1, đặt r = b1 là đầu phải của đoạn này. k = 1, v = [id[0]]

r = b[0]

k = 1

v = [id[0]]

* Với mỗi đoạn i: i=1..n-1 tiếp theo ta xét:

Nếu đầu trái của đoạn i, ai > r thì lấy đoạn i đưa vào kết quả (k=k+1,v=v+id[i]) và chỉnh r là đầu phải của đoạn i. r = bi.

for i in range(1,n):

if a[i] >= r:

k += 1

v.append(id[i])

r = b[i]

* Ghi k và v ra tệp DOANROI.OUT

print(k, file = fout)

for i in v:

print(i,file = fout)

fin.close()

fout.close()