

# Thuật toán ứng dụng buổi 3

Tham lam không phải lúc nào cũng cho kết quả tối ưu  
↳ Ở bài này nó là tối ưu (✓ better algo)

## Bài 1. Disjoint Segment

Tìm tập các khoảng thời gian không giao nhau lớn nhất

- Cho đầu vào là tập  $X$  gồm  $n$  khoảng thời gian  $X = \{(a_1, b_1), \dots, (a_n, b_n)\}$  với  $a_i < b_i$
- Hãy tìm và đưa ra tập con lớn nhất của  $X$  chứa các khoảng thời gian này sao cho không có 2 khoảng nào giao nhau

### Input

- Dòng 1 là giá trị của  $n$  ( $1 \leq n \leq 100000$ )
- Các dòng tiếp theo là các khoảng thời gian  $(a_i, b_i)$  ( $1 \leq a_i \leq b_i \leq 1000000$ )

### Output

- Số lượng phần tử của tập con lớn nhất

stdin	stdout
6	4
0 10	
3 7	
6 14	
9 11	
12 15	
17 19	

① Chọn cặp kết thúc sớm nhất

② Tìm xem đoạn nào ở chớm vào, lấy (ở thì thì, b<sub>o</sub>)

③ Tiếp tục xét các mốc kết thúc mới, xem cái nào sớm nhất

④ Lặp lại bước ①

### Ý tưởng

- Hai khoảng  $(a_1, b_1)$  và  $(a_2, b_2)$  không giao nhau nếu  $b_1 \leq a_2$
- Sắp xếp các đoạn theo thứ tự tăng dần thời gian, hoặc độ dài khoảng
- Tham lam 1. Chọn khoảng bắt đầu sớm nhất
- Tham lam 2. Chọn khoảng ngắn nhất
- Tham lam 3. Chọn khoảng kết thúc sớm nhất
- Vết cặn: Duyệt hết tập con → không khả thi

### Câu hỏi

Câu 1. Trong các thuật toán tham lam 1,2 trên hãy chỉ ra ít nhất 1 phản ví dụ của nó?

Câu 2. Liệu bạn có thể chứng minh tính chính xác của thuật toán tham lam 3?

Câu 3. Độ phức tạp trong trường hợp tối nhất của thuật toán 3?

## Bài 2. MAX-DISTANCE SUB-SEQUENCE

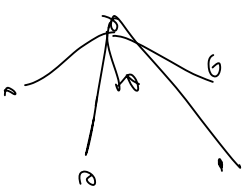
Cho 1 tập gồm  $N$  điểm trong không gian 1 chiều (đường thẳng). **Độ kết nối** của cụm (tập con) gồm  $C$  phần tử được tạo ra từ tập ban đầu được tính bằng khoảng cách của 2 điểm gần nhau nhất trong cụm

VD. Cụm gồm  $C=5$  điểm là 12,30,15,10,37 thì độ kết nối của cụm sẽ là 2, chính là khoảng cách nhỏ nhất giữa 12 và 10

stdin	stdout
1 5 3 1 2 8 4 9	3

5 ptử, xét  $\forall$  tập con có size = 3 từ 5 ptử này

Naive: duyệt hết!  
keep track of min



- Dòng đầu tiên sẽ là số lượng test trong bộ test  $T$  ( $1 \leq T \leq 20$ )

Cách 1. Tìm tất cả các tập con có  $C$  ptử; rồi xét  $\bigcirc$  những tập con đó độ kết nối nào là lớn nhất

Cách 2. Tìm klc lớn nhất  $\rightarrow$  kiểm tra xem solution ấy có tồn tại hay  $\circ$

Ví dụ hình vẽ: 1 2 8 4 9 → Tìm xem liệu có tập con: t/ñ hiệu giữa 2 ph<sup>3</sup> bất kỳ  $\geq 1$  → Idea: Sort dãy các đ<sup>2</sup> 1 2 4 8 9

- Các dòng tiếp theo sẽ là chi tiết các test. Mỗi test tương ứng gồm 2 giá trị là N và C, trong đó N là số lượng phần tử và C là kích thước tập con. Trong 1 bộ dữ liệu thì giá trị N sẽ là giống nhau trong các test.
- Các dòng tiếp sau sẽ lần lượt là giá trị tọa độ của các điểm trong tập N điểm
- Giá trị N ( $2 \leq N \leq 100,000$ ) và tọa độ các điểm  $x_1, \dots, x_n$  ( $0 \leq x \leq 1,000,000,000$ )
- Đầu ra sẽ là độ kết nối lớn nhất của tập con với kích thước C tương ứng (với các test). Nếu có T test thì sẽ có T giá trị đầu ra, với mỗi kết quả đầu ra được in trên 1 dòng.

Gợi ý.

- Nếu dùng vét cạn để tìm khoảng cách nhỏ nhất giữa 2 điểm trong không gian 1D thì sẽ mất thời gian  $O(n^2)$ . Ta có thể sắp xếp dãy trước (chi phí sắp xếp là  $O(n \log n)$  hoặc  $O(n^2)$ ), rồi tìm khoảng cách nhỏ nhất với thời gian chỉ  $O(n)$ .
- Nếu vét cạn hết các tập con có thể thì thời gian quá lớn ( $2^n$ ), ta cần cách tiếp cận thông minh hơn.
- Nếu xuất phát từ tập con 2 điểm, thì tập con có độ kết nối lớn nhất sẽ tạo bởi việc chọn 2 điểm xa nhất có thể. Khi thêm điểm tiếp theo thì chọn điểm sao cho nó xa 2 điểm kia nhất (điểm gần giữa 2 điểm đã chọn trước).
- Cách tiếp cận tham lam?
  - Sắp xếp trước danh sách N điểm
  - Chọn tập đầu là 2 điểm mút ( 2 điểm xa nhất)
  - Lần lượt chọn các điểm tiếp theo (cho tới khi đủ C) là điểm giữa dãy (đoạn con)

VD. Các điểm sau khi sắp xếp là 1,10,12,15,21,34

Và C = 3

Tập điểm 2 ban đầu là {1,34}, vậy điểm tiếp sẽ là 15 với độ kết nối là 14

Nếu C = 4

Tập điểm 2 ban đầu là {1,34}, điểm tiếp sẽ là 15, và điểm tiếp sẽ là 21 với độ kết nối sẽ là 6

**Câu hỏi 1.** Lần chia tiếp theo ta sẽ chia tại nửa nào?

**Câu hỏi 2.** Độ phức tạp của thuật toán sẽ là?

Do không tới uđ bước xét duyệt đ/á (xem có  $\exists$  tập con ...) nên ta chỉ tới uđ bước tìm kết quả [k/c max].

B1 Sort dãy các điểm

1 2 8 4 9  $\longrightarrow$  1 2 4 8 9

B2 Xét k/c  $0 \rightarrow a[n] - a[1]$   
( $9 - 1 = 8$ )

Liệu có thể có 1 tập con t/nh k/c  $\geq 8$  (do độ liên kết của tập là k/c nhỏ nhất có thể giữa 2 pđ)

$\hookrightarrow$  ① 2 4 8 ⑨

Chỉ lấy đc 2 pđ  $\Rightarrow$  0 đđ  $C = 3$

B3 Xét [binary search] trên k/c mới:  $\frac{8+0}{2} = 4$

Lấy đc tập t/nh?	1	2	4	8	9
$\downarrow$	✓	X	X	✓	X

Không

Xét tiếp khoảng dưới:  $0 \rightarrow 4$   $\frac{4+0}{2} = 2$

Lấy đc tập t/nh?	1	2	4	8	9
$\downarrow$	✓	X	✓	✓	
	✓		✓		✓

Có

Xét khoảng trên:  $2 \rightarrow 4$   $\frac{4+2}{2} = 3$

$\downarrow$

Có

1	2	4	8	9
✓		✓	✓	

(xét trái)

$\rightarrow$  Khoảng cuối cùng là ③

\* Vì đang maximize k/c tìm đc, nên hướng search:

nếu  $\nexists$  tập t/nh: lower k/c

nếu  $\exists$ : up lên ngay!

(xét phải)

### Nếu các phần tử trong dãy lệch nhau nhiều thì cách chia có khác?

VD. Dãy 1,5,10,21,55,100,300,1000

Gợi ý tham lam 2.

- Tính khoảng cách của 2 điểm gần nhau trước
- Chia tổng khoảng cách ra sao cho đều nhất có thể

Với dãy ví dụ thì các khoảng cách sẽ lần lượt là

4,5,11,34,45,200,700

Vậy điểm thứ 3 được chọn (sau khi chọn 2 nút là 1 và 1000) sẽ là 300, điểm thứ 4 được chọn sẽ là 100,...

**Câu hỏi 3.** Thuật toán tham lam này sẽ có độ phức tạp là bao nhiêu?

### Cách chia như vậy có đảm bảo là đúng?

VD với dãy ban đầu 1,5,10,40,50,70,100

- Với  $C = 2$  thì ta chọn 1 và 100
- Với  $C = 3$  ta chọn 1,50,100
- Nhưng  $C=4$  thì liệu 50 còn được chọn?  
Ta sẽ chọn tối ưu là 1,40,70 và 100

Vậy ngoài việc cố định được nút là điểm nhỏ nhất và lớn nhất thì KHÔNG có gì đảm bảo các điểm giữa sẽ được chọn trong các giá trị  $C$  tiếp theo.

### Thuật toán đúng sẽ làm thế nào?

- Sắp xếp dãy trước với thời gian cỡ  $O(n \log n)$
- Min và Max chắc chắn sẽ là 2 điểm thuộc  $C$
- Min và Max sẽ quyết định độ lớn tối đa của kết nối.  
VD. Với dãy 1,5,10,40,50,70,100 và  $C = 5$  thì độ kết nối lớn nhất sẽ phải nhỏ hơn  $(100-1)/(5-2)$  là cỡ 33
- Ta sẽ dò tìm giá trị chính xác của độ kết nối lớn nhất dùng kết hợp thuật toán tìm kiếm nhị phân và backtracking

B1. Sắp xếp dãy

B2. Tìm min, max

B3. Tìm giá trị ban đầu (giới hạn trên) của độ kết nối sẽ là  $(\max - \min)/(C-2)$  với  $C$  là số lượng phần tử của tập con

B4.  $A_1 = \min$ ,  $A_c = \max$ ,  $k = (\max - \min) / (C - 2)$ ,  $k_{\max} = k$  lần lượt tìm  $A_2, A_3, \dots, A_{\max-1}$

Tìm  $A_i$  bằng cách tìm nhị phân phần tử gần  $A_{i-1} + k$  trong dãy, và cập nhật lại  $k_{\max} = \min(k_{\max}, A_i - A_{i-1})$

Nếu dãy tiếp theo KHÔNG còn đủ phần tử để chọn thì quay lại bước trước chọn phần tử ngay trước  $A_i$  đã chọn trong dãy.

Nếu chọn đủ phần tử thì ghi nhận lại giá trị  $k_{\max}$  đã tìm được, sau đó cũng quay lui để thử tìm xem còn cách chọn nào cho  $k_{\max}$  lớn hơn không. Ta cũng tiến hành quay lui như trường hợp không còn phần tử nào để chọn.

Ví dụ với dãy 1, 5, 10, 30, 45, 55, 70, 100 và  $C = 5$

$A_1 = 1$  và  $A_5 = 100$ ,  $k = 33$

Tìm  $A_2$ . Bằng cách tìm nhị phân phần tử gần  $A_1 + k = 34$  nhất trong dãy 5, 10, 30, 45, 55, 70. Ta chọn được  $A_2 = 30$ . Với  $k_{\max}$  sẽ là  $\min(33, 29) = 29$

Tìm  $A_3$ . Bằng cách tìm nhị phân phần tử gần  $A_2 + k = 63$  nhất trong dãy 45, 55, 70. Ta chọn được 70. Với  $k_{\max}$  là  $\min(29, 40) = 29$

Tìm  $A_4$ . Lúc này dãy tiếp theo là rỗng nên phải backtracking lại chọn  $A_3$

Ta sẽ chọn  $A_3$  mới là phần tử gần hơn trong dãy còn lại của **45, 55, 70**. Ta chọn 55, với  $k_{\max} = \min(29, 25) = 25$ .

Tìm  $A_4$ . Trong dãy 70. Ta chọn 70 và  $k_{\max} = \min(25, 15) = 15$

Giờ muốn xét nốt phương án tối ưu hơn?

Quay lại tìm  $A_4 \rightarrow$  Quay lại tìm  $A_3$ . Chọn  $A_3$  mới là 45 thì bằng  $k_{\max}$  lớn nhất ta tìm được ở lần trước  $\rightarrow$  quay lui lại  $A_2 \rightarrow$  chọn  $A_2$  là giá trị trước 30, là 10  $\rightarrow k_{\max}$  nhỏ hơn giá trị lớn nhất ta tìm được  $\rightarrow$  dừng

Vậy trong dãy trên với  $C=5$  ta tìm được  $k_{\max} = 15$

**Câu hỏi 4.** Ta sẽ không quay lui hết mà dừng quay lui tới khi nào?

**Câu hỏi 5.** Độ phức tạp tính toán của thuật toán trên?

\* Lấy điểm vớt: Duyệt toàn bộ (i:  $0 \rightarrow n-1$ ;  $j = i+1 \rightarrow n$ )  
 nếu  $a_i > a_j \rightarrow \text{count}++$

→ Thông minh hơn: Làm sao để o phải duyệt toàn bộ (thứ tốn  $O(\frac{n(n-1)}{2})$ )?

### Bài 3. Inversion – đảo ngược

- Cho một dãy số nguyên  $a_1, a_2, \dots, a_n$ . Một cặp  $(a_i, a_j)$  được gọi là bị đảo ngược nếu  $i < j$  nhưng giá trị  $a_i > a_j$
- Hãy đếm số lượng các cặp bị đảo ngược trong dãy

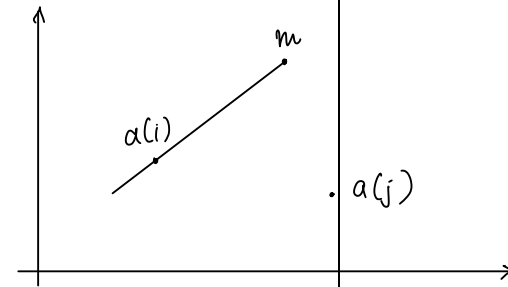
#### Input

- Dòng 1: là giá trị của  $n$  ( $1 \leq n \leq 10^6$ )
- Dòng 2: là các giá trị của  $a_1, a_2, \dots, a_n$ . ( $0 \leq a_i \leq 10^6$ )

#### Output

Số lượng cặp bị đảo  $Q \bmod 10^9 + 7$

stdin	stdout
6	6
3 2 4 5 6 1	



#### Gợi ý:

- Số lượng cặp trong dãy  $n$  phần tử sẽ là  $n(n-1)/2$
- Duyệt và đếm số lượng cặp theo phương pháp tham lam thông thường:  $O(n^2)$
- Thuật toán nhanh hơn?

① Nếu có 1 dãy tăng

Tại  $a(i)$ ,  $a(i) > a(j)$

→ Mọi ptử trong dãy  $> a(j)$   
 ↳ Chỉ cần đếm số ptử

② Những làm sao để sort?

→ Sắp xếp tăng những làm sao để  
 cho id x các ptử đc sort vẫn  $\in$   
 khoảng ban đầu

#### Ý tưởng chia để trị?

- Chia dãy thành 2 nửa bằng nhau:
  - Số lượng đổi chỗ gồm: là tổng số đổi chỗ trong nửa trái  $L$ , và nửa phải  $R$
  - Cộng với số lượng đổi chỗ của 1 phần tử thuộc nửa trái sang 1 phần tử nửa phải -  $M$  ( $\approx$  Merge Sort)
- Cách tìm số lượng đổi chỗ giữa 2 nửa  $M$  sao cho nhanh?
  - Nếu duyệt vét cạn vẫn là  $n/2 \times n/2 = n^2/4 \rightarrow$  thuật toán vẫn là  $O(n^2)$  như vét cạn
  - Không vét cạn?

↳ Code merge sort, cộng thêm 1 vài dòng

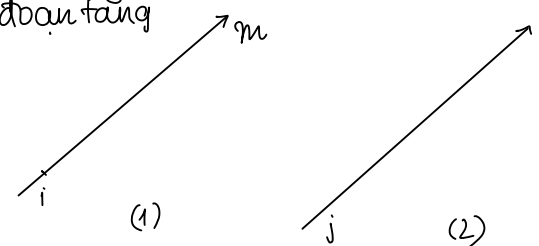
Giả sử dãy trái và phải đều có thứ tự tăng thì số lượng đổi chỗ giữa 1 phần tử nửa trái và 1 phần tử nửa phải sẽ tính là bao nhiêu

VD. Nửa trái  $L = 1, 5, 10, 12$  và nửa phải  $R = 3, 11, 17, 21$

Hãy để ý tới giá trị chốt của mỗi nửa

- Nửa trái: Chốt là phần tử cuối cùng

Xét 2 đoạn tăng



Nếu ở  $a[i] > a[j]$

→ cả đoạn (1) nghịch đảo vs  $a[j]$

⇒ số nghịch đảo:  $(m - i + 1)$



- Nửa phải: Chốt là phần tử đầu tiên

Với dãy trên thì rõ ràng 1 sẽ ko cần đổi chỗ, nhưng 5,10,12 sẽ là 3 lần đổi chỗ có thể với 3 bên nửa phải. 12 sẽ có 1 lần đổi chỗ với 11 bên nửa phải

Cách tính?

- So sánh với phần tử cuối cùng của nửa trái hoặc đầu tiên nửa phải
- Thời gian tìm:  $O(n)$

Vậy trong ý tưởng chia để trị ta kết hợp việc đếm với việc sắp xếp dãy như trong sắp xếp trộn – Merge Sort

### Câu hỏi

Câu 1. Công thức đệ quy tổng quát cho ý tưởng chia để trị sẽ là?

Câu 2. So với thuật toán vét cạn thì thuật toán chia để trị này có nhược điểm gì? Liệu thuật toán vét cạn có tồi hơn chia để trị không?