Contents

[- **AtCoder Beginner Contest 192 :** 2](#_Toc75807540)

[- **AtCoder Beginner Contest 193 :** 2](#_Toc75807541)

[- **Educational Codeforces Round 105:** 2](#_Toc75807542)

[- **AtCoder Beginner Contest 194** 2](#_Toc75807543)

[- **AtCoder Beginner Contest 195** 3](#_Toc75807544)

[- **AtCoder Beginner Contest 196** 3](#_Toc75807545)

[- **AtCoder Beginner Contest 197** 4](#_Toc75807546)

[- **AtCoder Beginner Contest 197** 5](#_Toc75807547)

[- **AtCoder Beginner Contest 199** 5](#_Toc75807548)

[- **ZONe Energy Programming Contest** 6](#_Toc75807549)

[- **AtCoder Beginner Contest 200 (09 tháng 05 năm 2021)** 7](#_Toc75807550)

[- **AtCoder Beginner Contest 201 :** 8](#_Toc75807551)

[- **AtCoder Beginner Contest 202:** 8](#_Toc75807552)

[- Atcoder dp contest 9](#_Toc75807553)

[- AtCoder Beginner Contest 204: 9](#_Toc75807554)

[- AtCoder Beginner Contest 205: 9](#_Toc75807555)

[- AtCoder Beginner Contest 206: 9](#_Toc75807556)

* **AtCoder Beginner Contest 192 :**

Bài C: Sử dụng mảng để sort các chữ số.

***Bài D***: Base n

***Bài E***: Cài Dijkstra với độ phức tạp O(n^2) chưa cải tiến bằng cách dùng Piority\_queue để đưa độ phức tạp về O(n+m)

* **AtCoder Beginner Contest 193 :**

***Bài C***: Sử dụng vòng lặp for để tính a^b

* [**Educational Codeforces Round 105**](http://codeforces.com/contest/1494)**:**

Mô tả trạng thái bằng các bit và sử dụng bitwise AND ( & )

* **AtCoder Beginner Contest 194**

***Bài C***: Cho mảng tính tổng với độ phức tạp O(n)

Bài này khá hay, làm mình nhớ đến câu nói của thầy Tamaki : ”Tất cả các phương trình đều vẽ ra một cái gì đó, và đôi khi em sẽ nhìn thấy kết quả trước khi tính toán ”. Bài này là một ví dụ như vậy, phương trình đề bài cho vẽ ra một bảng NxN như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Ta có nhận xét sau : với thì tổng đã cho là nửa dưới đường chéo chính của bảng trên. Tiếp đó ta có nhận xét, nếu tính tổng cả bảng trên thì ta phải tính với cả trường hợp , khi đó bảng có dạng, chú ý rằng tổng đường chéo chính = 0.

Nhận xét tiếp theo là 2 nửa của bảng có tổng bằng nhau, bởi vậy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | …. | … |  |
|  |  | .. | … |  |
|  |  |  | … |  |
|  |  |  |  |  |
|  |  |  |  |  |

Sau nhận xét này thì ta có thể tách tổng mới thành từng tổng từng phần. Và giải quyết trong độ phức tạp O(n)

***Bài D***: Cho một đồ thị n đỉnh, ta đứng ở đỉnh 1. Mỗi lần chọn ngẫu nhiên 1 đỉnh trong n đỉnh với xác suất 1/n, nếu không phải đỉnh đang đứng, ta dựng đường đi từ đỉnh 1 đến đỉnh đó. Tính giá trị kỳ vọng đến khi đồ thị đã cho liên thông hoàn toàn.

Các kiến thức có liên quan trong bài :

Phân bố hình học ( Geometric distribution ) : Xác suất để có sự thành công đầu tiên của phép thử. Ví dụ: một phép thử x lần, ta muốn xác suất để lần đầu tiên thành công xảy ra. Mỗi lần thử ta có 2 khả năng, thành công hoặc thất bại, tổng quát, nếu xác suất thành công là p, xác suất thất bại là 1-p, sau n lần thử, ta có hàm phân phối như sau:

Tính chất :

Chứng minh : ta có nhận xét rằng, đây là một chuỗi hình học (geometric series)

Giá trị kỳ vọng : Theo định nghĩa, giá trị kỳ vọng cho một biến ngẫu nhiên x được xác định bởi : với x là giá trị của biến ngẫu nhiên, p là xác suất của biến đó. Ví dụ về giá trị kỳ vọng của một con đề ?

Quay lại phân bố hình học, ta thấy giá trị kỳ vọng có thể được định nghĩa như sau:

Chuỗi hình học ( Geometric series ) : Chuỗi hình học ( hay chuỗi cấp số nhân là một chuỗi có tỉ lệ giữa các số hạng kế tiếp nhau là một hằng số ) chuỗi này thường có tổng là một hằng số. Ví dụ :

(1) r(2)-(1) ⬄

Khi n tiến đến vô cùng

Ví dụ: Một người chơi ném vòng trúng thưởng, anh ta có 5 lượt ném, 4 lượt đầu trượt, tính xs để lượt cuối trúng, biết xác suất thành công để ném trúng là 30%.

* **AtCoder Beginner Contest 195**

***Bài B***: Programing notes:

1. Chia 2 số int thì ra 1 số int
2. Hàm ceil và floor nhận giá trị kiểu double.

* **AtCoder Beginner Contest 196**

***Bài C***: **stoll** : convert string to long long.

**stol** : convert string to long int.

**stoi** : convert string to int.

**stoull** : convert string to unsigned long long.

**stroll** : convert string to long long integer.

**to\_string**:

2021/03/25

* Bitwise << : left shift, bit string longer -> bigger-> x2
* Bitwise >>: right shift, bit string sorter -> smaller-> /2
* **AtCoder Beginner Contest 197**

***Bài B***: với các bài di chuyển trong lưới ô vuông, ta có kỹ thuật dùng mảng đánh dấu

g/s với i là hàng, j là cột

|  |  |  |
| --- | --- | --- |
| i-1, j-1 | i-1, j | i-1, j+1 |
| i, j-1 |  | i , j+1 |
| i+1, j-1 | i+1, j | i+1, j+1 |

trái, trên, phải, dưới.

di = {0, 1, 0, -1}

dj = {-1, 0,1, 0}

đi chéo

di = {-1,-1 ,1,1 }

dj = {-1, 1,1,-1}

***Bài C*** : Cho một chuỗi N<=20 số , ta chia N số thành từng khoảng, rồi lấy OR trên các khoảng, sau đó lấy XOR của tất cả các khoảng này, tìm giá trị MIN đạt được.

Kiến thức: bitwise : AND, OR, XOR, leftshift, rightshift.

Trick: cách lấy bit 1 từ biểu diễn nhị phân của 1 số bằng cách righshift rồi AND với 1

Bài này dùng phương pháp vét cạn, ta biểu diễn trạng thái các khoảng dưới dạng một dãy nhị phân 01. Do N<=20, ta có tối đa 2^(N-1) = 2^19 = 1024\*1024/2 . Tiếp đó ta cần tính tổng trong từng khoảng, ta dùng phép dịch bit để tính toán tiếp.

***Bài D***. Trong mặt phẳng tọa độ, cho một đa giác đều N đỉnh, cho tọa độ 2 đỉnh P0, PN/2, tính P1 ( hoặc bất kỳ điểm nào )

* Kiến thức:

lượng giác 1rad = 180/PI.

Các hàm lượng giác trong C++

sin double sin (double x) : trả về giá trị sin của góc x radians.

cos double sin (double x) : trả về giá trị cos của góc x radians.

Các hàm lượng giác ngược trong C++

arc sin:

double asin (double x): nhận vào một giá trị kiểu double trong khoảng [-1;1] trả về giá trị góc theo radian

arc cos:

double acos (double x): nhận vào một giá trị kiểu double trong khoảng [-1;1] trả về giá trị góc theo radian

arc tan:

double atan (double x): nhận vào giá trị kiểu double và trả về giá trị góc theo radian, tuy nhiên không có khoảng cụ thể cho số nhận vào ban đầu.

double atan2 ( double y, double x ) : nhận vào giá trị tọa độ của 1 điểm P (x,y), sau đó trả về giá trị góc P0y

* Cách làm: đây là bài toán sử dụng phép quay đã học ở cấp 3. Phép quay trong mặt phẳng tọa độ có thể sử dụng công thức lượng giác để tính, tuy nhiên ta còn có thể sử dụng mặt phẳng phức.
* **AtCoder Beginner Contest 197**

***Bài D*** là một bài không khó lắm nhưng cần sử dụng nhiều kỹ thuật lập trình. Qua bài toán này ta thêm được các kiến thức sau.

* for ( auto it = … ; != … ; it++ ) , for ( auto it : [containter ] )
* Ép kiểu
* Các kiến thức liên quan đến container

Sắp xếp một set hoặc sort một mảng theo thứ tự, ta cần nạp vào một greater <> template.

set là một container, vì vậy ta nạp greater<> dưới dạng một typedef

sort là một hàm, vì vậy ta nạp greater<> dưới dạng một hàm, đó là greater<>()

map là một cấu trúc dữ liệu mạnh, tiện lợi, nhưng truy xuất tốn thời gian ( logN )

set: dùng để đếm các phần tử đã xuất hiện, giữ chúng theo một thứ tự

hàm iota : khởi tạo các giá trị liên tiếp trong một khoảng nào đó.

* Hàm sinh hoán vị trong C: next\_permutation.

***Bài E***: Sử dụng dfs, lưu ý khi code dfs

* Khi bắt đầu vào thăm một đỉnh, ta đánh dấu đỉnh đó.
* Khi thăm xong đỉnh đó và chuyển sang đỉnh tiếp theo cùng cha, ta cần lưu ý các điều kiện liên quan, vì lúc vào đỉnh đó ta đã đánh dấu rồi.
* **AtCoder Beginner Contest 199**

***Bài C***: Cho 1 xâu s độ dài 2n (n = 2.10^5) và q truy vấn (q = 3.10^5), mỗi truy vấn thực hiện 1 trong 2 thao tác:

T1: swap 2 ký tự s[a] và s[b];

T2: swap s[0 – n] và s[n – 2n]

Lúc đầu mình đã nghĩ là sử dụng segment tree để thực hiện truy vấn, nhưng cách giải quyết vấn đề thực sự đơn giản hơn nhiều. Ta có nhận xét là nếu thực hiện truy vấn T2 2 lần thì ta không thay đổi xâu. Vậy nên ta đánh dấu những lần xuất hiện truy vấn T2 để thực hiện phép xoay s.substr một lần duy nhất. đối với truy vấn T1, tùy theo trạng thái của xâu mà ta sẽ swap các biến khác nhau.

***Bài D***: Cho một đồ thị N đỉnh ( N<20 ), và m cạnh ( m< N(N-1)/2 ), mỗi đỉnh có thể có 3 màu R,G,B. 2 đỉnh kề không cùng màu. Tính tất cả các cách sinh ra đồ thị. Bài này dựa trên bài toán tô màu đồ thị đã được giới thiệu ở đại học. Trong lúc thi, mình cũng hình dung ra ý tưởng là vét cạn bằng dfs nhưng không code được.

* **ZONe Energy Programming Contest**

***Bài B***: Khá tiếc vì đã hình dung ra ý tưởng nhưng do phân vân nên không code. Mình vẫn rất gượng khi làm mấy bài toán, suy nghĩ khá cồng kềnh. Có thể do lâu không code nên phản xạ ít nhiều bị mất đi.

***Bài D***: phân biệt 2 hàm resever và reverse trong C++

reserve : Ta cấp phát thêm bộ nhớ cho xâu trong trường hợp muốn lưu trữ dữ liệu.

reverse : Ta đảo ngược một xâu.

Cấu trúc dữ liệu deque. Đối với một xâu, ta có thể push\_back, pop\_back. Hãy chú ý nhớ kỹ về điều kiện của đề bài, suy nghĩ kỹ xem sau mỗi bước trạng thái thay đổi như thế nào. Thao tác đang được thực hiện trên dữ liệu ảo hay dữ liệu thực ?

Đây là một bài khá hay, mặc dù đưa ra được ý tưởng nhưng mình không code được. Dù trước đây đã làm việc nhiều với kiểu dữ liệu string trong C++ nhưng mình không biết rằng có thể push\_back hoặc pop\_back một ký tự trong string.

Khi duyệt một xâu thì dùng for (char c : s) vừa ngắn vừa không bug.

Bài này mình suy nghĩ nhiều nhưng vẫn không chạm đến ý tưởng lõi của nó, khi xử lý một xâu thì phải luôn nhớ kỹ rằng đang thao tác trên xâu thực hay xâu tạm

***Bài C***: Cho một mảng 2 chiều a[3000][5], ta cần lấy ra 3 hàng sao cho min của max trên từng cột . Đây là một bài hay, để làm được nó ngoài việc nắm vững kỹ thuật tìm kiếm nhị phân cơ bản còn cần một số kỹ thuật lập trình khác.

Phân tích: Các bài toán có dạng min của max .. hoặc max của min… đều đẩy về dạng tìm kiếm nhị phân được, tại sao lại như vậy ? vì min của max tức là một phần tử duy nhất, một phần tử duy nhất tạo ra một trạng thái mà ta khai thác được max trong min của input và phần tử này tạo ra hiện tượng phân biệt trạng thái rất rõ ràng trong dữ liệu. Bởi tính duy nhất của phần tử này, ta sẽ sử dụng tìm kiếm nhị phân để tìm ra nó.

Kỹ thuật tìm kiếm nhị phân: ý tưởng của tìm kiếm nhị phân rất đơn giản, ta có một mảng gồm n phần tử đã được sắp xếp, ta cần tìm phần tử x có xuất hiện trong mảng hay không bằng cách loại bỏ tất cả các phần tử nhỏ hơn x và tất cả các phần tử lớn hơn x.

Set up l,r : l = 0

While (l<r){

m = (l+r)/2;

if ( a[m] > x ) r = m ; // điểm mid vẫn lớn hơn x, cần thu hẹp phạm vi tìm kiếm lại

if ( a[m] < x ) l = m; // điểm mid vẫn bé hơn x, cần tăng phạm vi tìm kiếm

Như vậy ta nhận thấy là tại điểm cuối của vòng lặp thì có 2 trường hợp, hoặc tìm ra x hoặc là không.

- Nếu không tìm ra x thì có 2 th đó là l chạy về phía r ( x rất lớn so với mảng a ) hoặc r chạy về phía l ( x rất nhỏ so với mảng a ) tức l = r;

- Nếu tìm ra x thì x = a[m], vào lúc này l = r = mid;

Đến đây ta nhận ra khi l = r cũng là lúc kết thúc vòng lặp.

}

Quest: tại sao snuke code là l =0 , l+1 <r ?

------

Bình luận: ZOne contest đánh dấu 2 tháng trở lại với thuật toán của mình. Ưu điểm của mình so với 3 năm trước là mình không sợ bất kỳ bài nào. Mình luôn sẵn sàng đọc và code lại tất cả các bài, miễn là mình hiểu đầy đủ bản chất của nó. Mình lỳ đòn hơn trước. Tuy nhiên mình tự thấy bản thân khá chậm hơn so với ngày xưa, dù đã có những bài tự nghĩ ra trường hợp đặc biệt nhưng vẫn có những bài tự đi vào vết xe đổ của mình. Cần rút kinh nghiệm thêm.

* **AtCoder Beginner Contest 200 (09 tháng 05 năm 2021)**

Bài A: mình không biết công thức tính thế kỷ.

Bài C: cho một dãy dài 2.10^5, tính số cặp sao cho (.

Tính chất modulo khi .

Vì c = 200 nên ta có thể lưu số lượng các đồng dư lại để tính toán d[]++. Ta tính xem từ mỗi đồng dư tạo được bao nhiêu cặp, vì mỗi cặp không quan tâm đến thứ tự, nên việc này tương đương với lấy .

Tản mạn về factorial, permutation, combination. Hồi đi học phổ thông, chúng ta quen thuộc với 3 khái niệm cơ bản liên quan đến cách bố trí k phần tử vào n phần tử. Khi còn đi học tôi được học lần lượt các khái niệm với thứ tự và tên gọi như sau. Factorial: hoán vị, permutation: tổ hợp, combination: chỉnh hợp. Thực sự tôi thấy mấy cái tên này rất khó hình dung. Và tôi quyết định dành ra 1 ngày để tái định nghĩa thứ tự một cách tự nhiên nhất. Vấn đề cách sắp xếp như sau, ví dụ: ta có 4 số 1,2,3,4 và ta muốn tạo ra một từ gồm có 3 chữ cái ghép lại từ 3 từ đó. Một cách rất dễ hình dung, vị trí thứ 1 có 4 cách chọn, thứ 2 có 3 cách chọn, thứ 3 có 2 cách. Và các khả năng của chúng ta được biểu diễn như sau:

123 124 132 134 142 143

213 214 231 234 241 243

312 314 321 324 341 342

412 413 421 423 431 432

Số cách sắp xếp là 4.3.2 = 24. Mở rộng ra với trường hợp chọn k phần tử trong n phần tử ta có: gọi số cách sắp xếp là P, P = n(n-1)(n-2)….(n-( k+1)) và ta ký hiệu giá trị này là

Chữ P của từ Permutation nghĩa là sự hoán đổi, sự thay đổi. Như ta đã thấy, ta cần hoán đổi các con số vào các vị trí khác nhau. Và điều gì sẽ xảy ra nếu chúng ta sắp xếp n phần tử vào n vị trí ? Ta có vị trí thứ 1 có 4 cách chọn, thứ 2 có 3 cách chọn, thứ 3 có 2 cách, vị trí thứ 2 có 2 cách và cuối cùng có 1 cách. Số cách sắp xếp là 5.4.3.2.1 cách chọn. Vì tích này chạy từ 1 đến n, vì vậy ta có thể viết một cách ngắn gọn hơn. Đó là 5! Và gọi nó là factorial.

Chữ factorial mang nghĩa là một điều gì đó mang tính yếu tố, tiếng Anh ta có từ fun fact chỉ những điều vui vẻ thú vị, ít người biết. Ở đây ta có ý nghĩa tương tự

Tiếp tục với khái niệm Combination: Combination có nghĩa là sự kết hợp, động từ của nó là combine. Vậy thì câu hỏi đặt ra ở đây là cái gì kết hợp với cái gì ? Ta xem xét ví dụ vừa rồi. Ta nhận ra là dường như có một sự trùng lặp nào đó ở đây. Ta nhận thấy là với mỗi bộ số sẽ có những hoán vị riêng và thực ra số lượng kết hợp chỉ xoay quanh 4 thứ cơ bản. Và ta có số cách kết hợp chỉ là

123 132 213 231 312 321

124 142 214 241 412 421

134 143 341 314 413 431

234 243 321 342 423 432

Hôm mình làm contest 200 này, buổi sáng nghe thầy Vinh nói về tổ hợp, thiêng thật, buổi tối gặp ngay một bài tổ hợp, cảm thấy năng lượng chảy xuyên qua mình vậy, dù không làm được bài C nhưng mình có thêm một cơ hội để đọc lại về tổ hợp. Nếu như ở hồi đại học, mình nhớ kiến thức về đồng dư thì mình sẽ làm được ngay, mình sẽ tính số cách ghép cặp là n(n-1)/2 với một suy luận là: mỗi số n sẽ có n-1 cách ghép với nó, và vì không tính 2 lần nên kết quả sẽ là n(n-1)/2. Đây là một suy luận hoàn toàn có cơ sở và nó đúng. Thế nhưng bản chất sâu hơn của nó chính là combination.

* **AtCoder Beginner Contest 201 :**

Bài C: mã PIN gồm 4 chữ số, mỗi vị trí bao gồm 1 trong 4 chữ cái từ 0 đến 9 và có thể lặp. (max trạng thái) biểu diễn trạng thái có thể có của mỗi chữ số [0 .. 9] trong PIN bằng xâu ox?

o: chữ số i chắc chắn có trong PIN.

x: chữ số i không có trong PIN.

?: chữ số i có thể có hoặc không có trong PIN.

Tính xem có bao nhiêu khẳ năng của PIN

Bài này khá cay, mình ngồi cả giờ viết công thức tổ hợp cho PIN, trong khi thực tế có thể xử lý bằng vòng lặp. Vì chỉ có 10^4 nên duyệt từng state và thử xem từ tập [o][x][?] có thể sinh ra state đó hay không. Chú ý rằng không cần quan tâm đến tập [?] vì ta sẽ loại bỏ các trạng thái lỗi từ tập [o] và [x].

Điều kiện: tất cả trạng thái trong [o] xuất hiện trong state.

tất cả trạng thái trong [x] không xuất hiện trong state.

Bài này đọc code của một bạn 2k4 mới thấy càng học nhiều thì càng dễ loạn thật.

* **AtCoder Beginner Contest 202:**

Bài C: cho 3 mảng A, B, C mỗi mảng có N = 10^5 phần tử có giá trị trong khoảng [1-N] tính tất cả các cặp ( i , j ) sao cho, bài này chắc chắn precalculate rồi duyệt n để tính tổng, có 2 cách giải quyết.

C1: tính num\_a[ a[i] ] với num\_a[ a[i]] là số lần xuất hiện của phần tử a[i]

tính num\_b[ b[ c[j] ] ] với num\_b[ b[ c[j] ] ] là số lần xuất hiện của phần tử b[ c[j] ]

kết quả được tính bằng

C2: tính num\_a[a[i]] rồi tính tổng num\_b[ b[ c[j] ] ],

Mình thấy là công thức của mình (C1) là hiểu đúng theo nghĩa đen của “tính số cặp i,j” còn công thức c2 là tổng quát hơn, “tính xem có bao nhiêu “ mình không hiểu là tại sao mình hiểu cả 2 cách nhưng mình lại không chứng minh được chúng một cách toán học ?

# Atcoder dp contest

Bài G: dfs O(n^2) với mỗi đỉnh của đồ thị. Vì mình chưa quen đệ quy nên cần chú ý là sau khi thăm nút con thì đánh dấu là chưa thăm ( lúc đi ngược lên ) và độ dài đường đi được tính bằng d[u] = d[v] + 1;

Bài H: quy hoạch động mảng 2 chiều kiểu đường đi, mình dự định làm bài này theo 3 cách:

Đệ quy có nhớ: số cách đi đến (i,j) = số cách đi đến (i-1,j) + số cách đi đến (i,j-1), nếu ô đó có thể đi được.

For xuôi từ [1:1] đến [n:n]

For ngược từ [n:n] về [1:1]

# AtCoder Beginner Contest 204:

Bài C: dfs đến tất cả các đỉnh, sau đó reset lại trạng thái của các đỉnh. Mình khá bực bì không làm được bài này trong thời gian của kỳ thi. Một phần vì vào cuối tuần suy nghĩ không được thông suốt lắm, phần vì vẫn chưa thực sự tự tin khi đứng trước những bài đồ thị.

# AtCoder Beginner Contest 205:

Bài C: Cho 3 số -10^9<= a,b,c <= 10^9

Xác định xem khi nào a^c <,>,= b^c

vô cùng thất vọng vì bài dễ nhưng mình lại tư duy hết sức loằng ngoằng. Khi đứng trước những bài nhiều trường hợp mình luôn rất thiếu tự tin, không hiểu sao. Mình đã nghĩ đến việc chia trường hợp theo tính chẵn lẻ của c và có liên quan đến trị tuyệt đối với trường hợp chẵn, nhưng vẫn tư duy rất loằng ngoằng. Tại sao lại thế nhỉ ? Mình đã nhìn snuke tư duy và thấy là mình cũng có hướng suy nghĩ đúng, nhưng không hiểu sao cách giải quyết vấn đề ở đoạn cuối luôn bị rối. Một phần do lâu rồi mình cũng không làm toán ntn nên tư duy bị mòn đi nhiều.

Mình cần rút ra điều gì sau những bài sai ngớ ngẩn ntn ?

Bình tĩnh xem xét lại các tính chất toán và đọc lại đề.

Bài C bản chất không khó nên nếu làm sai phải thay đổi cách nghĩ ngay.

Giữ năng lượng để còn thi.

# AtCoder Beginner Contest 206:

Bài C : cho N số nguyên, tìm tất cả các cặp i,j với 1<i<j<=N và Ai ≠ Aj. Ràng buộc 2<=N <= 3x10^5, Ai <=10^9.

Đây là một bài toán đếm điển hình. Ý tưởng ban đầu của mình là tạo 1 set chứa tất cả các phần tử khác nhau rồi tính số cặp tạo được, gọi là size set, res = n\*(n-1). Tuy nhiên ý tưởng này sai, bởi vì nếu đếm như vậy thì các phần tử trùng nhau sẽ không được sử dụng, trong khi thực tế là có. Ví dụ 3,1,1,2 nếu cho vào set thì ta được tập là 3,1,2 thì có 2\*3 = 6 cặp

Tuy nhiên thực tế thì có 3,1 | 3,1 | 3,2| 1,2| 1,2 -> 5 cặp

Ngoài ra thì bài này có ràng buộc i<j, vì vậy nên không thể thay đổi thứ tự các phần tử trong mảng được.

Vì i<j nên tại mỗi i ta sẽ chứa kết quả số cặp tại vị trí i -> có thể giải trong O(n) -> tính số cặp tại i

Nếu đến vị trí i thì số cặp tạo ra với phần tử a[i] sẽ chính là giá trị của i.-> bây giờ cần loại bỏ số cặp trùng nhau

Tại mỗi vị trí i, ta đếm số lượng a[i] từ đầu đến vị trí đó. tại mỗi vị trí i, số cặp tạo thành sẽ = i- a[i]

***Tổng quát :***

* Tại mỗi vị trí i, ta có số lượng cặp tạo thành với a[i] = i
* Để tính số lượng cặp khác nhau ta lấy tổng số - số cặp giống nhau
* Số cặp giống nhau = số lượng phần tử a[i] tại vị trí i.

Bài D: Cho một mảng gồm n số nguyên thực hiện thao tác sau :

Thay đổi tất cả các số x thành y đến khi xâu trở thành palindrome, tức

a[i] = a[n-i] with 0<= i <=n-1

1<=N<= 2x10^5

1<=Ai<=2x10^5

Ta nhận thấy luôn có thể biến xâu thành một palind, ví dụ trường hợp tất cả phần tử trong mảng đều khác nhau

1 2 3 4 5 -> 1 1 3 4 5 -> 1 1 1 4 5 ->1 1 1 1 5 ->1 1 1 1 1

1 2 1 2 4

Khi ta thay đổi một đại lượng trong xâu thì có 3 giá trị sau thay đổi, đó là giá trị, số lượng và vị trí

Ta muốn số bước thay đổi nhỏ nhất -> lượng thay đổi sau k lần phải max.

Cần xác định khoảng thay đổi và các thay đổi kéo theo.

Xác định vị trí -> giá trị -> số lượng.

Chạy 2 con trỏ ở 2 đầu xâu, từ các vị trí đó xác định được các giá trị cần thay đổi.

Ta nhận thấy khi thay đổi 2 giá trị a[i] a[j] nào đó thì có a[i] thành a[j] hoặc ngược lại. Điều này khiến cho giữa 2 phần tử a[i] và a[j] xuất hiện quan hệ 2 chiều. và sau khi thay đổi thì số lượng phần tử có giá trị a[i] sẽ thay đổi.

* Câu hỏi đặt ra ở đây là làm thế nào xác định được số bước ít nhất để thay đổi ?

Bằng việc biểu diễn các cặp quan hệ giống nhau thành 1 quan hệ duy nhất giữa 2 phần tử , rồi sau đó từ đây tất cả các phần liên kết với nhau sẽ trở thành 1 nhóm. Ta có thể dựng đồ thị cho các cặp a[i], a[n-i], nếu a[i] khác a[n-i] thì ta sẽ có đường đi từ a[i] -> a[n-i], sau đó dfs để tính đường đi, mỗi đường đi giữa 1 cặp đỉnh là một phép biến đổi.

Ngoài ra theo như hướng dẫn trên mạng thì từ đây sẽ bắt đầu có sử dụng một kiến thức gọi là union find, vì vậy ta sẽ tìm hiểu về unionfind.

Hiểu thuật toán, cài đặt, build thành namespace

Phân tích disjoint-set data structure.

***Cập nhật ngày 27 tháng 6:***

Cũng khá vui vì mình đã tự code được bài này với code dfs đệ quy. Bài này mình nghĩ là đã từng gặp hồi học đại học rồi, nhưng hồi ấy mình chắc sẽ hình dung ra bài này là một cái gì đấy rất kinh khủng. Thế nhưng bây giờ mình đã có thể hiểu được ý tưởng của nó.

Mình code dfs như sau: hàm dfs kiểu void, sử dụng biến đếm bên trong nó. Mỗi lần đến một đỉnh thì tăng giá trị lên 1. Như vậy mình đã đếm tất cả các đỉnh có đường đi -> số bước để đi đến n đỉn là n-1. Cách này ok, mỗi tội quên không đặt số lượng tối đa của ma trận kề là maxn.

Dfs đệ quy cách 2 : ta có thể kết hợp hàm dfs để trả về giá trị số bước để đi đến mỗi đỉnh.

Cách 3: vì ta cần đếm số lượng các đỉnh đã được thăm -> = n –

Cách 4: disjoint set.

Cách 5 : bfs.