2021 - Đồ án cuối kỳ IT003 xxTN

Báo cáo tham gia contest Codeforces

Tên: Phan Trọng Nhân

Lớp: KHTN2021

MSSV: 21522407

Link Codeforces: another_alwcod_nn

Link contests: Educational Codeforces Round 128 (Rated for Div. 2) và Codeforces Round 794 (Div. 2)

Minh chứng giải:

Codeforces Round #794 (Div. 2)

Final standings

You may double click into cells (or ctrl+click) to view the submissions history or hack the solution

Standings

#	Who	=	*	A 500	B 1000	C 1500	D 2000	E 2500	F 2500
1 (32)	another_alwcod_nn	4170		486 00:07	964 00:09	1288 00:27	1432 01:11		

Educational Codeforces Round 128 (Rated for Div. 2)

Final standings, open hacking phase a terminé

Double click (or ctrl+click) each entry to view its submission history

Standings

#	Who	=	Penalty	*	A	B	C	D	E	F
1 (449)	another_alwcod_nn	4	165		+ 00:03	+ 00:13	+ 00:36		+ 01:53	

I. Codeforces Round 794 (Div. 2)

- Các bài giải được: A, B, C, D.

- Cảm nhận: Các câu sau của đề (từ bài D) khó hơn bình thường rất nhiều, từ cảm nhận tới thực tế (chỉ có 1, 2 người AC E, F), và lấy được 3750 điểm khó hơn nhiều. Các câu cũng không cần bằng lắm về độ khó và là ad-hoc là nhiều.

1. problem A

- Tóm tắt đề: Cho dãy số nguyên n phần tử $a_1, a_2, ..., a_n$, có phần tử nào đó của dãy mà trung bình của $n-1$ phần tử còn lại bằng phần tử này không?

- Nhận xét: Với giới hạn của đề, có thể giải quyết bài toán bằng thuật toán trâu, nhưng để tiết kiệm thời gian code và để học hỏi thêm, phần phía dưới sẽ tìm cách đưa ra giải thuật tối ưu.

- Bằng cảm giác có nhận định: Nếu trung bình của $n-1$ phần tử bất kì trong mảng bằng phần tử còn lại, thì trung bình của n phần tử trong mảng cũng bằng phần tử còn lại này.

- Có thể dễ dàng chứng minh điều này bằng toán học: Nếu phần tử ta không chọn là a_i, và điều kiện trong đề được đáp ứng, tức là $(a_1 + a_2 + ... + a_{i-1} + a_{i+1} + ... + a_n)/(n-1) = a_1$, sử dụng biến đổi toán học cơ bản, ta có

$(a_1 + a_2 + ... + a_{i-1} + a_{i+1} + ... + a_n)/(n-1) = a_i$

$<=> a_1 + a_2 + ... + a_{i-1} + a_{i+1} + ... + a_n = a_i * (n-1)$

$<=> a_1 + a_2 + ... + a_n = a_i * n$ (cộng hai vế cho a_i)

$<=> (a_1 + a_2 + ... + a_n)/n = a_i$

- Vế trái là trung bình các số trong dãy và có thể tính được từ trước, từ đó ta có giải thuật: ban đầu tính trung bình các số trong dãy, sau đó duyệt từng phần tử của dãy, nếu có ít nhất một phần tử bằng trung bình dãy thì in YES, còn lại in NO. Giải thuật này có độ phức tạp $O(N)$, và vì việc đọc các phần tử đã tốn $O(N)$, giải thuật này là tối ưu.

2. problem B

- Tóm tắt đề: Cho dãy số nguyên n phần tử $a_1, a_2, ..., a_n$, chia dãy thành các đoạn con sao cho số dãy có số nghịch thế là số lẻ là nhiều nhất.

- Ý tưởng: Khi nghĩ tới việc phải chia ra nhiều đoạn nhất, việc tôi nghĩ đầu tiên là các đoạn phải có số phần tử càng ít càng tốt, và nghĩ ngay tới một đoạn gồm hai phần tử kề nhau $[a_i, a_{i+1}]$ mà $a_i > a_{i+1}$, nếu tìm được nhiều nhất các đoạn gồm hai phần tử này thì các đoạn còn lại sẽ tăng

dần (vì nếu không tăng dần thì đã có thêm một đoạn hai phần tử mới) nên không cần quan tâm nữa (chia như thế nào cũng không có nghịch thế).

- Bây giờ chúng ta tìm nhiều nhất các đoạn hai trên. Ý tưởng giải thuật tham lam như sau: duyệt từng phần tử từ đầu tới cuối, khi gặp một phần tử i mà $a_i > a_{i+1}$ thì lấy luôn đoạn $[a_i, a_{i+1}]$ và không xét phần tử $i+1$ nữa. Giải thuật này đúng vì nếu không lấy luôn đoạn này thì sẽ có ít cặp để lựa chọn hơn, và từ phần tử $i+3$ (nếu không lấy đoạn $[a_i, a_{i+1}]$ thì phải lấy đoạn $[a_{i+1}, a_{i+2}]$, nếu không thì phí phần tử a_{i+1} và sẽ không tối ưu hơn cách lấy $[a_i, a_{i+1}]$), đoạn như cũ nên cũng không thể tối ưu hơn.

- Ngoài ra còn có một cách làm khác sử dụng quy hoạch động (nghĩ ra sau contest): $dp_i = max(dp_{i-1}, dp_{i-2} + (a_i < a_{i-1}))$ (bỏ qua a_i hoặc lấy $[a_{i-1}, a_i]$ nếu $a_i < a_{i-1}$)

3. problem C

- Tóm tắt đề: Sắp xếp lại mảng (mảng vòng tròn, tức là kế sau phần tử cuối là phần tử đầu) sao cho mỗi phần tử hoặc lớn hoặc nhỏ hơn hai phần tử liền kề.

- Ý tưởng: Nhìn đề bài, tối nghĩ tới cách xếp phần tử nhỏ, lớn, nhỏ, lớn,... phần tử cuối phải lớn vì kề sau nó là phần tử nhỏ (chứng minh theo toán học : $a_1 < a_2 > a_3 < a_4 > ... < a_n > a_1$, nếu $a_i > a_{i+1}$ thì i chẵn, nên n phải chẵn)

- Từ kinh nghiệm có một cách sắp xếp: sắp xếp tăng dần mảng, gọi mảng này là b, chia hai nửa, nửa đầu là các phần tử nhỏ, tức là $a_1 = b_1$, $a_3 = b_2$, $a_5 = b_3$, ... nửa sau là các phần tử lớn, tức là $a_2 = b_{n/2+1}$, $a_4 = b_{n/2+2}$, $a_6 = b_{n/2+3}$, ... Nếu cách này không đúng thì không có cách nào đúng cả, bởi khi đó sẽ có hai phần tử bằng nhau đứng cạnh nhau, hai phần tử này trong mảng tăng dần sẽ cách nhau $n/2$, tức là mảng có tối thiểu $n/2+1$ phần tử bằng nhau, theo Dirichlet, nếu các phần tử này cách nhau, thì sẽ có tối đa $n/2$ phần tử này, nên mảng trên luôn tồn tại hai phần tử bằng nhau cạnh nhau, và sẽ không có cách sắp xếp thỏa mãn.

4. problem D

- Tóm tắt đề: Có 4 loại từ: A, B, AB, BA. Cho số lượng từng loại từ (bốn số a, b, c, d) và một xâu (s), có thể ghép các loại từ kia với số lượng cho trước tạo thành xâu đó hay không?

- Đầu tiên, hiển nhiên nhất, ta phải kiểm tra số lượng chữ cái, tức là $a+c+d = s_A$ và $b+c+d = s_B$ (s_A là số chữ cái A trong xâu s, tương tự với s_B).

- Quan sát đầu tiên là các xâu dạng ABA hay BAB sẽ ảnh hưởng tới cách chọn (trong xâu ABA hay BAB thì chỉ được chọn AB hoặc BA).

- Ý tưởng tới sau đó là quy hoạch động (từ nhận xét chỉ cần quan tâm AB và BA) sao cho số lượng AB là nhiều nhất và vẫn vừa đủ cho BA, nhưng sau nhiều phút không tìm được cách tối ưu thì tìm cách khác.

- Sau đó có cách gom nhóm từ quan sát phía trên: các xâu chữ cái xen kẽ sẽ là một nhóm ($ABABAB...$ và $BABABA...$) và tôi tập trung tìm đặc điểm của các nhóm này.

- Nhận xét: trong các xâu độ dài chẵn, số AB lấy được và số BA lấy được sẽ khác nhau, cụ thể trong xấu $ABABAB...$ thì nếu lấy ít nhất một BA thì tổng số từ lấy được sẽ bị giảm đi 1, nếu chỉ lấy AB thì không bị giảm (tương tự với xâu $BABABA...$). Trong các xâu độ dài lẻ, số từ AB và BA lấy được không bị giảm. Từ đó, tôi chia thành 4 nhóm theo hai đặc điểm: độ dài chẵn hay lẻ,

chữ cái bắt đầu của xâu.

- Tiếp theo là tìm cách lấy từ. Tôi nghĩ tới thuật toán tham lam tiếp theo. Với xâu độ dài chẵn, với xâu $ABABAB...$ thì ưu tiên lấy từ AB vì sẽ được số từ nhiều hơn, với xâu $BABABA...$ thì sẽ lấy từ BA. Xâu độ dài lẻ thì lấy theo thứ tự nào không quan trọng lắm nên tôi phân cho AB trước rồi BA. Tiếp theo, nếu chưa đủ, phân $ABABAB...$ cho BA, và $BABABA...$ cho AB (số từ lấy được sẽ giảm đi 1).

- Mọi thứ đều rất ổn cho tới khi chạy test mẫu và nhận ra kết quả sai, và tôi nhận ra đó là do thứ tự lấy nhóm (trong test đề có $ABAB$ và $ABABAB$, nếu lấy 2 từ AB từ nhóm 2, thì chỉ còn lại 1 từ BA, nhưng nếu lấy 2 từ AB từ nhóm 1, thì còn lại 2 từ BA). Vấn đề ở đây, là khi lấy xâu độ dài chẵn phân cho từ khác chữ cái đầu (ví dụ $ABABAB...$ phân cho BA), thì với mỗi xâu số lượng từ sẽ giảm đi 1, nên cần phải có số xâu ít nhất có thể, từ đó có giải pháp là khi thực hiện những theo tác trước cần ưu tiên lấy các xâu ngắn hơn, nên sắp xếp xâu theo độ dài tăng dần trong mỗi nhóm.

II. Educational Codeforces Round 128 (Rated for Div. 2)

- Các bài giải được: A, B, C, E.

- Cảm nhận: Đề không quá khó, nhưng tôi mất nhiều thời gian cho bài D, sau đó nhìn bảng rank và chuyển sang E kịp lúc.

1. problem A

- Tóm tắt đề: Tìm số phần tử ít nhất của dãy thỏa điều kiện: có từ l_1 tới r_1 phần tử nhỏ nhất và từ l_2 tới r_2 phần tử lớn nhất.

- Ý tưởng: khá hiển nhiên ta sẽ xét 2 trường hợp: mọi phần tử bằng nhau (phần tử nhỏ nhất cũng là phần tử lớn nhất), và khác nhau (phần tử nhỏ nhất khác phần tử lớn nhất, ta chỉ quan tâm nhỏ nhất và lớn nhất nên dãy lúc này chỉ có 2 giá trị khác nhau).

+ Mọi phần tử bằng nhau: khi đó đoạn $[l_1, r_1]$ giao đoạn $[l_2, r_2]$ và ta lấy số nhỏ nhất thuộc cả hai đoạn này. Giải thuật: do phần tử lớn nhất hay nhỏ nhất không ảnh hưởng tới bài toán, ta giả sử $l_1 <= l_2$ (trường hợp ngược lại có thể hoán đổi l_1 và l_2, r_1 và r_2). khi đó $l_1 <= l_2 <= r_1 <= r_2$ hoặc $l_1 <= l_2 <= r_2 <= r_1$, trong cả hai trường hợp đều thấy $l_2 <= r1$ và đây là điều kiện để kiểm tra có giao, khi đó l_2 là số nhỏ nhất là giao của hai đoạn.

+ Không phải mọi phần tử đều bằng nhau: Đơn giản chỉ cần lấy l_1 phần tử nhỏ nhất, l_2 phần tử lớn nhất, đáp án là $l_1 + l_2$.

- Trường hợp 1 cho ra kết quả tối ưu hơn nên ta ưu tiên xét trước, nếu không giao mới in ra trường hợp 2.

2. problem B

- Tóm tắt đề: Cho vùng hình chữ nhật gồm nxm ô vuông. một số ô trên đó có robot. Được di chuyển mọi robot cùng lúc theo các hướng lên, xuống, trái, phải. Không được di chuyển robot ra khỏi vùng hình chữ nhật. Có thể di chuyển một robot bất kì về góc trái trên được không?

- Ý tưởng: Chỉ cần xét những robot có tọa độ trái nhất và trên nhất, vì những robot đó có khả năng đi ra ngoài. Kiểm tra từng robot, xét nếu robot này tới ô trái dưới thì những robot phía trên có ra khỏi vùng hình chữ nhật không.

- Giải thuật:

+ Lưu lại tung độ và hoành độ nhỏ nhất của các robot, lần lượt gọi là $nmin$ và $mmin$

+ Xét từng robot, có tọa độ là (i, j), để về ô góc trái trên (tọa độ (1, 1)), thì số ô phải đi là $(i-1, j-1)$, nên ta kiểm tra $nmin-(i-1)>0$ và $mmin-(j-1)>0$, nếu không thì in NO.

+ Nếu tất cả robot đều hợp lệ, in YES.

- Ý tưởng sau thi: Sau thi xong tôi có nghĩ ra một ý tưởng gọn hơn: Một robot đi được về ô góc trái trên nếu robot đó vừa là robot ở ô trái nhất vừa là robot ở ô trên nhất, hay cả hai tọa độ của robot phải nhỏ nhất trong số các robot.

3. problem C

- Tóm tắt đề: Cho xâu nhị phân s, được xóa từ đầu xâu hoặc cuối xâu một vài kí tự, tìm giá trị nhỏ nhất của max(số kí tự 0 còn lại, số kí tự 1 đã xóa).

- Thay vì xóa đầu, cuối xâu, tôi suy nghĩ theo hướng khác, là chọn một đoạn của xâu.

- Đặt số kí tự 0, 1 trong xâu là s_0, s_1. Với đoạn được chọn là $[l, r]$, số kí tự 0, 1 là p_0, p_1, cần tối ưu $max(p_0, s_1-p_1)$.

- Tạm gọi điểm mà $max(p_0, s_1-p_1)$ tối thiểu là cực tiểu, cần tìm điểm cực tiểu này. Xét nếu l giữ nguyên, r tăng, p_0 tăng, p_1 tăng nên s_1-p_1 giảm, nên $max(p_0, s_1-p_1)$ sẽ giảm dần sau đó tăng dần. Xét nếu r giữ nguyên, l tăng, p_0 giảm, p_1 giảm nên s_1-p_1 tăng, $max(p_0, s_1-p_1)$ tăng, mà cụ thể là điểm cực tiểu bị dịch sang phải (tăng r sẽ lại tìm được điểm cực tiểu).

- Từ ý tưởng và nhận xét trên, nhận thấy có thể áp dụng kĩ thuật hai con trỏ (two pointers). Có giải thuật như sau:

+ Để tính p_0, p_1, cần mảng cộng dồn: $f[i][k]$ (k là 0 hoặc 1) là số kí tự k trong đoạn $[1, i]$, số kí tự k trong đoạn $[l, r]$ là $f[r][k]-f[l-1][k]$.

+ Kĩ thuật hai con trỏ: duyệt l từ trái sáng phải xâu, tăng dần r trong khi $max(p_0, s_1-p_1)$ còn giảm. Lấy kết quả và so sánh với kết quả tối ưu để cập nhật.

4. problem E

- Tóm tắt đề: Cho bảng a $2xn$ ô bao gồm ô trống('.') và con chip('*'). Được di chuyển mỗi lượt một con chip theo hướng lên, xuống, trái, phải không ra ngoài bảng. Khi di chuyển một con chip tới ô chứa một con chip khác thì cả hai hợp nhất. Tìm số lượt di chuyển ít nhất để tất cả con chip hợp lại thành một.

- Cái đầu tiên tôi nghĩ tới là hợp lại tại chip nào thì sẽ tối ưu, sau một hồi suy nghĩ và thử nghiệm, tôi rút ra rằng hợp lại tại chip nào không quan trọng, sẽ ra cùng kết quả.

- Từ đó, tôi nghĩ tới việc dồn chip từ trái sang phải để áp dụng kĩ thuật quy hoạch động, cơ bản thuật toán như sau:

+ $f[i][j]$ là số lượt di chuyển ít nhất để hợp chip về ô (i, j), $f[i][j]$ có thể đi tới được từ hai ô $f[i-1][j]$ (ô trước cùng hàng) và $f[i-1][!j]$ (ô trước khác hàng). Đi từ $f[i-1][j]$ thì công thức là $f[i-1][j]+1+(a[i][!j]==' *')$ (dồn ô $(i-1, j)$ tới (i, j) và ô cùng cột tới (i, j) nếu đó là chip). Đi từ $f[i-1][!j]$ thì cần $f[i-1][!j]+2$ vì dồn 2 ô, trong lúc đi đã phải dồn qua ô cùng cột rồi. Lấy cách đi tối ưu hơn trong hai cách đi này.

+ Quy hoạch động bắt đầu từ cột đầu tiên chứa chip.

+ Cuối cùng, in ô tối ưu hơn trong hai ô trong cột cuối cùng chứa chip.