Phần mở rộng * là PAS hay CPP tùy theo ngôn ngữ và môi trường lập trình

Cấu hình dịch:

G++ 4.9.2: -std=c++11 -O2 -s -static -Wl,--stack,66060288 -lm -x c++

FPC 3.0.4: -O2 -XS -Sg -Cs66060288

Đề có 3 trang.

Hãy lập chương trình giải các bài toán sau đây

Bài 1: Biến đổi nhị phân

Trong giờ ra chơi, Ánh viết lên tờ giấy của mình một dãy nhị phân *A* có độ dài là *n*, các phần tử lần lượt là $a_1, a_2, \ldots, a_n$. Sau đó Ánh chọn ra một dãy con liên tiếp $a_i, a_{i+1}, \ldots, a_j$ $(1 \le i \le j \le n)$ và thực hiện phép *xor* với 1 cho mỗi phần tử. Sau khi thực hiện, nếu a_x có giá trị ban đầu bằng 0 thì sẽ biến thành 1 và ngược lại. Ánh tự hỏi, với giá trị nào của *i* và *j* thì sẽ được một dãy nhị phân với nhiều số 1 nhất.

Bạn hãy viết chương trình trả lời câu hỏi của Ánh.

Dữ liệu vào: Đọc từ tệp văn bản ***BITSEQ.INP*** có cấu trúc:

- Dòng đầu chứa số nguyên *n*.
- Dòng thứ hai chứa *n* số thể hiện dãy nhị phân, mỗi số cách nhau ít nhất một khoảng trắng.

Dữ liệu ra: Ghi ra tệp văn bản ***BITSEQ.OUT*** một số nguyên duy nhất là số chữ số 1 nhiều nhất của dãy đã cho sau khi biến đổi.

Giới hạn:

- 20% số test đầu ứng với 20% số điểm có $1 \le n \le 500$.
- 30% số test tiếp theo với 30% số điểm có $500 < n \le 7000$
- 50% số test sau với 50% số điểm còn lại có $7000 < n \le 10^6$.

Ví dụ:

BITSEQ.INP	***BITSEQ.OUT***
9 1 0 0 1 0 0 1 1 0	7

Gợi ý thuật toán:

Thuật toán 1: Độ phức tạp $O(n^3)$ đạt 20% điểm số

Duyệt toàn bộ bằng 3 vòng lặp For lồng nhau để xét mọi khả năng xảy ra và so sánh để tìm kết quả.

Vòng lặp 1: Duyệt chỉ số i

Vòng lặp 2: Duyệt chỉ số j

Vòng lặp 3: Đếm số lượng chữ số 0 và 1 của đoạn [i,j] rồi so sánh để cập nhật kết quả.

Thuật toán 2: Độ phức tạp $O(n^2)$ đạt 50% điểm số.

Biến đổi mảng a thành mảng b: $b_1, b_2, \ldots, b_{n-1}, b_n$ theo quy tắc: $b_i = 1$ nếu $a_i = 0$, $b_i = -1$ nếu $a_i = 1$.

Chuyển bài toán ban đầu về dạng tìm đoạn con liên tiếp có tổng lớn nhất trên dãy b.

Gọi S[i] là tổng của dãy b từ phần tử b_1 đến phần tử b_i.

Sử dụng 2 vòng lặp để duyệt các giá trị i,j. Tổng $b_i + \ldots + b_j = S[j] - S[i-1]$.

Thuật toán 3: Độ phức tạp O(n) đạt 100% số điểm.

Chuyển đổi mảng a về mảng b như ở thuật toán 2.

Gọi k tổng của đoạn con liên tiếp có tổng lớn nhất xét từ phần tử đầu đến *i*.

Gọi res là tổng đoạn con có giá trị lớn nhất.

Gọi number1 là số lượng số -1 trong dãy $b_1, b_2, \ldots, b_{n-1}, b_n$

Duyệt *i* từ *1* đến *n*:

Ở mỗi bước duyệt ta cần cập nhật: $\boldsymbol{k = max(0, k + b_i)}$

Sau khi cập nhật *k*, ta tiếp tục cập nhật: ***res = max(res, k)***
Nếu *res* = 0 thì kết quả bài toán là *number1*-1.
Nếu *res* $\neq 0$ thì kết quả bài toán là *res* + *number1*.

BÀI 2: *Dãy nhị phân đẹp*
Một dãy nhị phân được coi là *"đẹp"* nếu không có quá *K* số 0 đứng cạnh nhau và không có quá *L* số 1 đứng cạnh nhau.
Cho trước hai số nguyên dương N, M.
Yêu cầu: Đếm số dãy nhị phân *"đẹp"* gồm có N số 0 và M số 1.
Dữ liệu vào: Đọc từ tệp văn bản **BINARY.INP** chứa bốn số lần lượt là N, M, K, L trên một dòng, mỗi số cách nhau ít nhất một khoảng trắng.
Dữ liệu ra: Ghi ra tệp văn bản **BINARY.OUT** một số duy nhất là kết quả tìm được sau khi chia lấy dư cho $10^9 + 7$.
Ví dụ:

BINARY.INP	BINARY.OUT
2 1 1 10	1

Giới hạn:
- 40% số test đầu ứng với 40% số điểm có *N, M* < 11 và *K, L* < 20.
- 60% số test tiếp theo ứng với 60% số điểm còn lại có *N, M* $\leq$ 100 và *K, L* $\leq$ 50

Gợi ý thuật toán:
1. Với subtask đầu tiên ta có thể sử dụng thuật toán quay lui để tính kết quả bài toán.
2. Với subtask thứ 2 ta có thể sử dụng quy hoạch động với mô hình top-down như sau:

Ý nghĩa: f[n][m][cntk][cntl] là số dãy nhị phân gồm có n chữ số 0 và m chữ số và *cntk* và *cntl* để lưu lại *cntk* số 0 liên tiếp hoặc *cntl* số 1 liên tiếp.
Ta có công thức:
f[a][b][*cntk*][*cntl*] = f[a-1][b][cntk+1][0] + f[a][b-1][0][cmtl+1].
Quá trình tính cần một mảng visit để đánh dấu xem đã tính f[a][b][cntk][cntl] hay chưa.
Kết quả: f[n][m][0][0].

Bài 3: DIVSEQ
Ở đất nước Alpha, mỗi người dân khi sinh ra được Nhà Vua cấp cho một dãy số nguyên dương B. Vì muốn đất nước an khang thịnh vượng vị Vua quyết định chỉ chọn các dãy số B là các dãy số "*may mắn*". Một dãy số B được Nhà Vua coi là dãy số "*may mắn*" nếu nó thỏa mãn B_i chia hết cho B_{i-1}.
Hiện tại Nhà Vua đang có một dãy số nguyên dương *A* có *n* phần tử $A_1, A_2, \ldots, A_n$ ($0 < A_i \leq 10^6$), các phần tử đôi một khác nhau. Nhà vua muốn biết, dãy A có bao nhiêu *dãy con* là dãy số "*may mắn*".
Dãy B được coi là dãy con của dãy số A nếu có thể thu được dãy B bằng cách xóa đi một số phần tử của dãy A. Hai dãy con được coi là khác nhau nếu có ít nhất một phần tử chỉ thuộc một trong hai dãy.
Dữ liệu vào: Đọc từ tệp văn bản **DIVSEQ.INP** có cấu trúc
- Dòng đầu tiên chứa số nguyên dương *n*.
- Dòng tiếp theo chứa *n* số nguyên là các phần tử của dãy A.

Dữ liệu ra: Ghi ra tệp văn bản **DIVSEQ.OUT** một số nguyên duy nhất là số dãy con may mắn của dãy A. Trước khi ghi ra, kết quả phải được mod cho $10^9 + 7$.
Ví dụ:

DIVSEQ.INP	DIVSEQ.OUT	*Giải thích ví dụ*
3 1 2 3	5	có 5 dãy con "*may mắn*" là: {1, 2}, {1, 3}, {1}, {2}, {3}

Giới hạn:
- 20% số test tương ứng với 20% số điểm có n $\leq$ 20.
- 40% số test tiếp theo tương ứng với 40% số điểm có 20 < n $\leq$3000.

- 40% số test cuối cùng tương ứng với 40% số điểm có $3000 < n \leq 100000$.

Gợi ý thuật toán:

Subtask 1: Độ phức tạp $O(2^n)$

- Sinh dãy nhị phân C với $C_i = 1$ khi và chỉ khi phần tử thứ i của dãy a nằm trong dãy con chia hết tương ứng với dãy C. Số lượng dãy C tìm được chính là đáp án của bài toán.

Subtask 2: Độ phức tạp $O(n^2)$

- Gọi $f(i)$ là số lượng dãy may mắn kết thúc tại phần tử thứ i của dãy a.
- Công thức để tính $f(i)$ như sau:

  $f(i) = \sum_{a_j \mid a_i} f(j)$. ($\boldsymbol{a_j \mid a_i}$ nghĩa là a_j chia hết cho a_i)
- Kết quả là $\sum_{i=1}^{n} f(i)$.

Subtask 3: Độ phức tạp O(n.logn)

- Gọi $f(i)$là số lượng dãy may mắn kết thúc tại phần tử thứ i của dãy a.
- Gọi $g(x)$ là số lượng dãy may mắn kết thúc bằng phần tử có giá trị là x và phần tử cuối có số thứ tự không vượt quá vị trí đang xét.
- Khi duyệt đến vị trí thứ i, ta thấy $f(i) = g(i)$. Dùng thêm một vòng lặp để cập nhật tất cả các $g(x)$ sao cho $x\ chia\ hết\ cho\ a_i$.
- Kết quả là $\sum_{i=1}^{n} f(i)$.

--------- Hết ---------