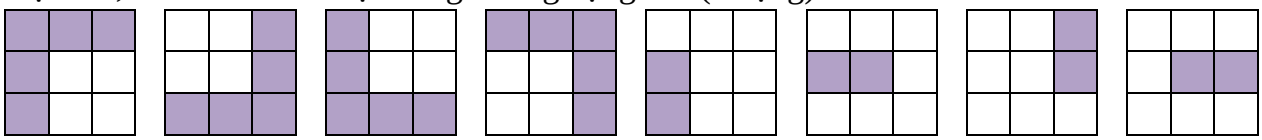
Bài 1. (6 điểm) Chữ L

Nhận xét, mỗi chữ L có một trong những dạng sau (8 dạng)



Mỗi chữ L có một ô $a[i][j]$ làm gốc

Dùng 4 mảng tính trước

$l[i][j]$ là tổng lớn nhất bên trái chứa $a[i][j]$

$r[i][j]$ là tổng lớn nhất bên phải chứa $a[i][j]$

$u[i][j]$ là tổng lớn nhất bên trên chứa $a[i][j]$

$l[i][j]$ là tổng lớn nhất bên dưới chứa $a[i][j]$

Duyệt tất cả các ô trong bảng, mỗi ô $a[i][j]$ ta xét 8 trường hợp trên. Tìm max các chữ L chứa ô $a[i][j]$ làm gốc

Bài 2. (7 điểm) Giá trị lớn nhất và giá trị nhỏ nhất

Subtask 1: Độ phức tạp $O(n^3)$

Ta thử hết với các cặp (l, r) nằm trong đoạn $[1, n]$, với mỗi đoạn ta sẽ tiến hành tìm max và min trong đoạn đó và kiểm tra xem min và max của chúng có bằng y và x tương ứng không

Subtask 2: Độ phức tạp $O(n^2)$

Ta dùng mảng tính trước

Ta gọi max$[i]$ $[j]$ là vị trí số lớn nhất trong đoạn $[i, j]$

$$\max[i][j] = \begin{cases} \max[i][j-1] & a[\max[i][j-1]] > a[j] \\ j & a[\max[i, j-1] \le a[j] \end{cases}$$

Tương tự, ta gọi i min$[i]$ $[j]$ là vị trí số lớn nhất trong đoạn $[i, j]$

Duyệt tất cả các dãy con có vị trí đầu là i và vị trí cuối là j, với mỗi dãy con, ta kiểm tra xem nếu $a[\max[i, j]] = x$ và $a[\min[i, j]] = y$, thì ta đã đếm được một dãy thỏa mãn, ta cộng 1 vào câu trả lời

Subtask 3: Độ phức tạp $O(n)$

Bài toán yêu cầu tìm số lượng các cặp số nguyên (l, r) thỏa mãn điều kiện đã cho, nhưng ta có thể nhìn dưới góc độ khác: "Có bao nhiêu dãy con liên tiếp $a_l, a_{l+1}, \ldots, a_r$ của dãy số (a_n) có giá trị lớn nhất là X và giá trị nhỏ nhất là Y?". Việc biến đổi bài toán đếm số cặp (l, r) thành bài toán đếm các đoạn thỏa mãn sẽ dễ dàng hơn. Ta xét một ví dụ sau: Giả sử dãy $A = (4,2,5,4,3,4,2), X = 4, Y = 2$ Trong ví dụ trên $a_3 = 5$ là phần tử khiến cho dãy A không có giá trị lớn nhất là $X = 4$. Do đó, bất cứ dãy con liên tiếp nào chứa a_3 sẽ không thỏa mãn điều kiện bài toán. Một phần tử như vậy sẽ khiến cho việc chia trường hợp của ta trở nên phức tạp. Do đó ta có thể nghĩ đến việc loại bỏ a_3 từ dãy đã cho, phân vùng và tách dãy con không chứa phần tử này, sau đó ta đếm số lượng các dãy con thỏa mãn. Như tình huống trên, ta tách dãy đã cho thành hai dãy con a_1 gồm 2 phần tử đầu tiên và a_2 gồm bốn phần tử cuối cùng, ta có thể giải quyết bài toán cho hai trường hợp sau:dãy $A_1 = (4,2), X = 4, Y = 2$, dãy $A_2 = (4,3,4,2), X = 4, Y = 2$, sau đó ta tìm câu trả lời trên mỗi dãy con này. Đây cũng là cách tiếp cận của trường hợp tổng quát:

Nếu a_i không thỏa mãn điều kiện $x \le a_i \le y$, ta có thể loại bỏ a_i để tách ra thành các dãy con, và tìm số lượng cặp thỏa mãn trong mỗi dãy con.

Ta gọi B là một dãy sau khi đã bị tách ra bởi thao tác trên. Ta sẽ giải quyết nó bằng các bài toán con sau:

Bài toán con 1: Có bao nhiêu dãy con liên tiếp $b_l, b_{l+1}, \dots, b_r$ của dãy B có giá trị lớn nhất là X và giá trị nhỏ nhất là Y. **Ở đây, mỗi phần tử của B nằm trong đoạn $[Y, X]$**. Điều kiện được bôi đậm ở trên là được thêm vào so với bài toán ban đầu. Khi đó bài toán "**Tìm phần tử lớn nhất và nhỏ nhất là Y và X**" sẽ tương ứng với bài toán "**Cả hai số X và Y có xuất hiện trong dãy con**".

Bài toán con 2: Có bao nhiêu dãy con liên tiếp $b_l, b_{l+1}, \dots, b_r$ của dãy B bao gồm cả hai số X và Y?

Nếu ta giải quyết bài toán con 2 trong thời gian tuyến tính, khi đó độ phức tạp của thuật toán sẽ là O(N), vì tổng số các dãy con B như vậy không vượt quá N. Có một vài cách để giải quyết bài toán 2.

Cách 1. Dùng kỹ thuật trượt cửa sổ

Ta gọi $[l, r]$ biểu thị dãy con $b_l, b_{l+1}, \dots, b_r$ Trong bài toán con 2, tính chất sau sẽ đúng:

Nếu đoạn $[k, l]$ chứa đựng đoạn $[i, j]$, hay nếu i, j, k, l thỏa mãn điều kiện $1 \le k \le i \le j \le l \le |B|$, thì khi đó, nếu đoạn $[i, j]$ thỏa mãn điều kiện, đoạn $[k, l]$ cũng thỏa mãn điều kiện

Trong trường hợp ,ta đếm số lượng các đoạn thỏa mãn điều kiện bằng giải thuật sau:

- Đầu tiên, cho $L = R = 1$
- Trong khi L không vượt quá N, thực hiện thao tác sau:
  - Trong khi $[L, R]$ không thỏa mãn điều kiện, thêm 1 vào R
  - Khi một đoạn [L,R] thỏa mãn điều kiện, khi đó mọi đoạn $[L, x](R \le x \le B)$ cũng thỏa mãn điều kiện, khi đó ta thêm số lượng vào câu trả lời
  - Thêm 1 tới L

Cách 2. Ta sử dụng nguyên lý bù trừ để đếm số lượng dãy con có cả hai X và Y

Bài 3 (7 điểm)

Sub 1:

_ Với mỗi bộ 3 (l, r, 2^k - 1)

=> A[l] = A[L+1] = … = A[R] = 2^k - 1.

_ Kết quả sẽ là: $2^{\text{số lượng số i} \in [1,N] \text{ và không xuất hiện trong bất kỳ bộ số (l,r,x) nào}}$.

_ Với mỗi bộ 3 số (l, r, 2^k - 1), ta chỉ cần đánh dấu tất cả các số từ l đến r. Các số không được đánh dấu là các số không xuất hiện trong bất kỳ bộ số nào.

_ Để đánh dấu các số từ l đến r, ta có thể dùng trick cộng dồn 1D hoặc bất cứ ctdl nào khả thi.

_ Độ phức tạp: O(N+M).

Sub 2:

_ Vì phép AND các số có K bit là tổng hợp của K phép AND từng bit, ta sẽ tách bài toán ra thành K bài toán con, mỗi bài toán tính cho 1 bit j từ 0 đến K-1. Kết quả sẽ là tích K bài toán con.

_ Giả sử ta đang tính cho bit j. Khi đó, M bộ 3 số (l_i, r_i ,x_i) sẽ thành (l_i, r_r, y_i) với y_i là 0 hoặc 1 phụ thuộc vào bit thứ j của x_i là 0 hay 1.

_ Nhiệm vụ của ta là đếm số cách điền N số B_1,B_2,..,B_N (B_i = 0 hoặc B_i = 1) để với mỗi đoạn (l,r,y):

B[L] AND B[L+1] AND … AND B[R] = Y

hay min(B[L], B[L+1],..., B[R]) = Y

_ Khi đó, ta chỉ cần duyệt trâu 2^N xâu nhị phân tương đương với 2^N trạng thái 0 hoặc 1 của N số. Với mỗi trạng thái, chỉ cần for M bộ 3 (l_i, r_i, y_i) để kiểm tra xem có bộ 3 nào không thỏa mãn hay không.
_ Nếu làm như thế độ phức tạp là: $O(2^N * K * M * N)$, nên ta phải tối ưu như sau:

+ Sort M bộ 3 số theo r.
+ For mask từ 0 đến 2^N - 1 tương ứng với 2^N trạng thái.
+ Với mỗi trạng thái, for i từ 1 đến N, gọi last là vị trí số 1 cuối cùng ≤ i.
+ Xét các bộ 3 số có r = i. Kiểm tra xem (min(B[L->R]) = y). Dễ dàng tính được bằng cách so sánh L và last.

_ Độ phức tạp: $O(2^N * K * (M+N))$.

Sub 3:

_ Với mỗi bộ 3 số (l,r,1), cập nhật các số B[L], B[L+1], …, B[R] = 1.
Khi đó, ta chỉ cần quan tâm đến bộ 3 số (l,r,0), rút gọn lại thành các cặp số (l,r).
_ Gọi ok[i][j] = 1 nếu ∃ (l,r) thỏa mãn: $i \le l \le r \le j$.
= 0 nếu ∄ (l,r) thỏa mãn: $i \le l \le r \le j$.
_ Khởi tạo: ok[l][r] = 1 với ∀ (l,r).
_ Công thức truy hồi: ok[i][j] = max(ok[i][j], ok[i+1][j], ok[i][j-1])
_ Gọi dp[i]: là số lượng cách chọn 0/1 cho i số B[1], B[2], .., B[i] mà B[i] = 0.
_ Khởi tạo: dp[0] = 1.
_ Công thức truy hồi:

+ dp[i] = 0 (Nếu b[i] = 1)
+ dp[i] = $\sum_{j=0}^{i-1} dp[j]$ nếu ok[j+1][i-1] = 0.

_ Kết quả: dp[N+1].
_ Độ phức tạp: $O(K * (N^2 + M))$

Sub 4:

_ Do ok[j][i] ≥ ok[j'][i] với j ≤ j'. Công thức trên có thể viết gọn thành:
dp[i] = dp[j-1] + dp[j-2] +...+dp[i-1] (với j nhỏ nhất mà ok[j][i] = 0).
_ Tổng trên có thể dùng mảng cộng dồn (Prefix sum), còn j chính là số L lớn nhất trong các cặp số (L,R) mà R < i. Do đó j có thể được tính bằng chặt nhị phân hoặc 2 con trỏ.
_ Độ phức tạp: O(K * (N + M)).

---------HẾT---------