SOLUTION 18/12_C

Bài 1. Đếm bộ số nguyên dương

- **Subtask 1**

– Ta có thể dùng đệ quy quay lui để thử toàn bộ từng cách điền bộ số, sau đó kiểm tra xem bộ số đó có thỏa mãn hai tính chất đề cho hay không.

– Để tối ưu thời gian chạy thì sau khi điền số x ở vị trí i, ta chỉ thử các số $x, 2x, 3x, \ldots$ ở vị trí $i + 1$. Khi đó, chúng ta cũng không cần kiểm tra xem bộ số được xây dựng có hợp lệ hay không nữa.

- **Subtask 2**

– Ta đặt $f[i][x]$ là số bộ số $A_1, A_2, \ldots, A_i$ thỏa mãn đề, và $A_i = x$. Công thức quy hoạch động sẽ là:

$$f[i][x] = \sum_{d/i} f[i-1][d]$$

với $d|i$ có nghĩa là d là ước của i.

– Để giải được Subtask 2, chúng ta tìm các ước d của i trong $O(\sqrt{i})$

- **Subtask 3**

– Để giải được Subtask 3, thay vì giải $f[i]$ từ $f[i-1]$, thì chúng ta làm ngược lại ta tính $f[i+1]$ từ các $f[i]$.

– Sau khi ta tính được đáp án của $f[i][x]$, ta thấy nó có ảnh hưởng tới công thức quy hoạch động của các f$[i+1][kx]$, với $k \in N$. Vì vậy ta lần lượt cộng đáp án của $f[i+1][x], f[i+1][2x], \ldots$ cho $f[i][x]$

Bài 2. Kỳ thi Duyên Hải bổ ích

- **Subtask 1**

– Thử tất cả các cách đi, kiểm tra xem cách đi đó có thoả mãn không.

– Độ phức tạp: $O(\sum_{i=1}^{n}(i! \cdot i))$

- **Subtask 2**

Một đường đi thoả mãn qua các đỉnh $t_1, t_2, \ldots, t_k$ sẽ luôn có tính chất: $x_{t1} > x_{t2} > \ldots > x_{tk}$ và $y_{t1} < y_{t2} < \ldots < y_{tk}$. Do đó, khi ta sắp xếp lại các điểm theo chiều giảm dần của hoành độ thì dãy chỉ số của một đường đi thoả mãn chắc chắn là một dãy con của $1, 2, \ldots, n$. Tới đây thì không khó để nhận ra rằng một đường đi thoả mãn sẽ tương ứng với một dãy con tăng của $y_1, y_2, \ldots, y_n$

Vì vậy, số đường đi thoả mãn đi qua đúng l điểm sẽ chính là số dãy con tăng có đúng l phần tử của dãy $y_1, y_2, \ldots, y_n$

Ta sẽ giải quyết bài toán trên bằng phương pháp quy hoạch động.

Nhận thấy là ta không quan trọng các giá trị cụ thể của y mà chỉ cần quan tâm đến tính tương đối giữa các giá trị y, do vậy ta có thể rời rạc hoá chúng thành các toạ độ y sao cho 1 ≤ y ≤ n.

Đặt dp(i, j) là số dãy con có độ dài i và có phần tử cuối cùng bằng j.

Công thức quy hoạch động: dp(i, j) = $\sum_{k=1}^{j-1} dp(i-1,k)$

Khi đó tổng số dãy con có độ dài l là $\sum_{i=1}^{n} dp(l,i)$

Lưu ý xét cẩn thận trường hợp có nhiều điểm tham quan có chung hoành độ.

Độ phức tạp: $O(n^3)$.

- **Subtask 3**

Nhận thấy rằng ta có thể cải tiến quy hoạch động từ Subtask 2 bằng cách sử dụng cấu trúc dữ liệu như Fenwick Tree, Segment Tree để tính tổng một đoạn liên tiếp trong $O(log(n))$ thay vì $O(n)$.

Độ phức tạp: $O(n^2 log(n))$

Bài 3. Ảnh đẹp

Ta có nhận xét sau:

"Cho một hộp có m viên bi với nhiều màu khác nhau thỏa mãn:

- Tồn tại một màu x sao cho trong hộp có nhiều hơn một nửa số bi mang màu này

Khi đó, ta thực hiện liên tục thao tác sau cho đến khi trong hộp chỉ còn các viên bi cùng màu:

- Lấy hai viên bi khác màu và bỏ ra khỏi hộp

Với mọi cách thực hiện thao tác, màu cuối cùng còn lại sẽ là x"

Nếu cần biết số còn lại cuối cùng nếu thực hiện thao tác trên một dãy con của một dãy số cho trước, ta có thể sử dụng Cây phân đoạn (Segment Tree) như sau:

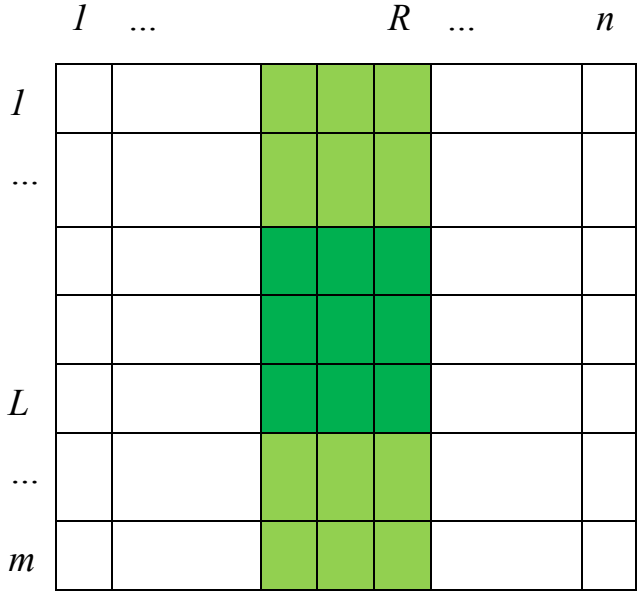
- Mỗi node quản lí đoạn $[L, R]$ lưu 2 giá trị:
  - x: Giá trị xuất hiện cuối cùng nếu ta thực hiện thao tác trên đoạn $[L, R]$
  - v: Số lượng x xuất hiện cuối cùng
- Khi đó, việc truy vấn thực hiện trên đoạn bất kì của dãy có thể làm bằng Cây phân đoạn này

- Thao tác hợp hai nút con lên nút cha như sau:

```
node Merge(node LeftChild, node RightChild):
    if LeftChild->v > RightChild->v:
        return [LeftChild->x, LeftChild->v-RightChild->v]
    else if RightChild->v < RightChild->v:
        return [RightChild->x, RightChild->v-LeftChild->v]
    else:
        return [0, 0]
```

Theo đó, với mỗi hình chứ nhật con kích thước $H \times W$, ta sẽ thực hiện một số lần thao tác trên nhằm lấy ra màu còn lại cuối cùng. Xem xét một cách thực hiện các thao tác như sau:

- Duyệt một biến R: $W \rightarrow n$, duy trì bảng con kích thước $m \times W$ (Sliding Window)
- Khi đó, ta coi mỗi hàng là một "hộp bi" như trong bài toán ở trên; việc tính màu còn lại và số lượng cho mỗi hàng có thể thực hiện trong khi duy trì bảng con, sử dụng m cây phân đoạn 1D



- Dựng một cây phân đoạn 1D quản lí m hàng của bảng con $m \times W$ này (Tương tự như m cây phân đoạn quản lí các hàng)
- Khi này duyệt L: $H \rightarrow m$, việc tính toán màu cuối cùng còn lại và số lượng có thể sử dụng cây phân đoạn mới dựng ra ở trên

Gọi màu còn lại nhận được qua Segment Tree là x, ta cần kiểm tra có thực x xuất hiện hơn một nửa bảng không. Bài toán còn lại là đếm số lần xuất hiện của giá trị x, ta có thể sử dụng nén số và cập nhật dần trong khi thực hiện Sliding Windows

Độ phức tạp: $O(m \times n \times \log 2\ m)$