Hướng dẫn giải Code Challenge #5

Big-O & VNG - 02/10/2022

BÀI A: Colorful Matrix

Tóm tắt đề bài:

Xem tại đây

Hướng dẫn giải

Thuật toán

Đây là một bài ứng dụng thuật toán quen thuộc ***LIS (Longest increasing subsequence)*** ở mức cao.

Nhận xét chung cho cả bài toán rằng tại một hàng rào i $(1 \leq i \leq n)$ đang xét thì ta chỉ cần xét các hàng rào j $(1 \leq j \leq i-1)$ trước đó. Gọi f_i là dãy con tăng dài nhất tại i. Từ đây ta thấy nếu như hai hàng rào i và j có ít nhất chung 1 cột tô màu đỏ thì $f_i = max(f_i, f_j + 1)$.

Vậy số hàng rào cần bỏ đi của bài toán là $n - max(f_i)$ với $(1 \leq i \leq n)$.

Subtask 1

Với cả n, m đều nhỏ, ta tiến hành ***Brute Force***. Duyệt các hàng rào từ 1 đến n, với mỗi hàng rào, ta kiểm tra các hàng rào đã duyệt trước đó, nếu thấy hàng rào nào có cột chung tô màu đỏ với hàng rào hiện tại thì cập nhật kết quả.

Độ phức tạp : $O(n^2logm)$

Subtask 2

Ở subtask này, chúng ta cần sử dụng một ***Data Structure*** để giải quyết cho phần việc phải duyệt hết tất cả các hàng rào j trước đó. Từ công thức $f_i = max(f_i, f_j + 1)$ ta thấy chỉ cần tìm giá trị lớn nhất của các j trước đó nên ta sẽ dùng ***Segment Tree*** để giải quyết.

Xây dựng cây *Segment Tree* cho 2 giá trị len và $index$ và một biến $last$ với ý nghĩa :

- $segment_i.len$ là độ dài dãy con tăng dài nhất.
- $segment_i.index$ là chỉ số cuối của dãy con tăng dài nhất.
- $last$ lưu chỉ số kết thúc sớm nhất của dãy con tăng dài nhất.

Ta vẫn xét các hàng i $(1 \leq i \leq n)$ theo thứ tự. Đầu tiên, ta tìm cặp giá trị lớn nhất nằm trong các khoảng $[l_i, r_i]$, đặt là u. Hai giá trị len và $index$ trong u thể hiện :

- len là độ dài của dãy dài nhất trước đó có ít nhất 1 cột được tô màu đỏ trong các khoảng $[l_i, r_i]$.
- $index$ là chỉ số cuối cùng của dãy hàng rào ứng với len.

Sau đó, ta cập nhật tất cả các khoảng $[l_i, r_i]$ theo ông thức :

- $segment_i = max(segment_i, (u.len + 1, i))$
- $last = i$ nếu tìm được một dãy có độ dài tốt hơn tại i

Chú ý : Các thao tác cập nhật và lấy giá trị dùng ***Lazy Propagation***.

Độ phức tạp : $O(n + mlogm)$

Mã nguồn tham khảo

- C++: https://ideone.com/K60jdH

BÀI B: Kiki - Trợ lý AI của Zalo

Tóm tắt đề bài:

Xem tại đây

Hướng dẫn giải

- Một trong mấy tính chất của hàm phi Euler là $$\phi(m\times n) \ge \phi(m) \times \phi(n)\text{ }(^*)$$ Cụ thể là $$\phi(m\times n) = \phi(m)\times\phi(n)\times\dfrac{d}{\phi(d)}$$ Với $d = gcd(m, n)$ và m, n là hai số nguyên dương bất kì. Do $\dfrac{d}{\phi(d)} \geq 1$ nên ta có điều $(^*)$ trên.
- Như thế chọn đoạn có đáp án lớn nhất là cả mảng A, vấn đề chỉ còn là tính giá trị $$\phi(\displaystyle \prod_{i = 1}^{N} A[i]) \mod(10^9 + 7)$$
- Ngoài ra, chúng ta có công thức tính hàm $\phi(n)$ với $n = p_1^{e_1} \times p_2^{e_2} \times ... \times p_k^{e_k}$ như sau:$$\phi(n) = p_1^{e_1 - 1}\times(p_1 - 1)\times p_2^{e_2 - 1}\times(p_2 - 1) \times ... \times p_k^{e_k - 1}\times(p_k - 1)$$
- Vậy ta sẽ phân tích thừa số nguyên tố từng phần tử $A[i]$ để biết các số nguyên tố và tính được hàm $\phi(n)$ theo công thức trên.
- Độ phức tạp cuối cùng $O(N \times log(\max(A[i])))$

Mã nguồn tham khảo

- C++: https://ideone.com/hFP8p9

BÀI C: VNG18 - Into the Verse

Tóm tắt đề bài:

Xem tại đây

Hướng dẫn giải

Với một giá trị N ta cần đếm số lượng thanh kẹo tạo ra được. Gọi x, y lần lượt là số viên kẹo theo chiều rộng và số viên kẹo theo chiều dài của thanh kẹo. Vì đề bài yêu cầu các thanh kẹo là khác nhau nên để tránh trường hợp đếm trùng ta sẽ đặt điều kiện là $y \geq x \geq 1$.

Vì số lượng kẹo không quá N nên: $xy \leq N \Rightarrow x^2 \leq N \Rightarrow x \leq \lfloor \sqrt{N} \rfloor$.

Với $x = x_0 \leq \lfloor \sqrt{N} \rfloor$ thì những giá trị y thỏa mãn $xy \leq N$ là $y \in \left\{x_0;\ x_0 + 1;\ x_0 + 2;\ ...; \left\lfloor \frac{N}{x_0} \right\rfloor \right\}$. Suy ra có $\left\lfloor \frac{N}{x_0} \right\rfloor - x_0 + 1$ giá trị y thỏa mãn.

Do đó với một giá trị N thì số thanh kẹo tạo được là: $S(N) = \sum_{i=1}^{\lfloor \sqrt{N} \rfloor} \left(\left\lfloor \frac{N}{i} \right\rfloor - i + 1 \right)$

Độ phức tạp: $O(T * \sqrt{N})$

Mã nguồn tham khảo

- C++: https://ideone.com/SeS6E2

BÀI D: Vua Trò Chơi

Tóm tắt đề bài:

Xem tại đây

Hướng dẫn giải

Trước hết ta sẽ xác định nếu quái vật còn lại i máu, thì người chơi tại lượt đó có thể thắng hay không bằng cách quy hoạch động như sau:

- $win[i]$ là mảng quy hoạch động nhận 2 giá trị $0, 1$
- $win[i] = 1$ khi người chơi đó sẽ **thắng** nếu quái vật còn i máu tại lượt của mình
- $win[i] = 0$ khi người chơi đó sẽ **thua** nếu quái vật còn i máu tại lượt của mình.
- Khởi tạo $win[0] = 0$, nghĩa là người chơi đó thua nếu quái vật chết khi đến lượt mình.

Cách tính $win[i]$:

- Để chiến thắng, người chơi cần khiến đối thủ đi vào trạng thái thua.
- Vậy $win[i] = 1$ khi tồn tại một lá bài sức mạnh v sao cho $win[i - v] = 0$, nếu không tìm được v thì $win[i] = 0$.

Vậy trường hợp $win[h] = 0$, tức là người chơi đầu tiên chắc chắn thua thì xuất ra -1.

Trường hợp $win[h] = 1$, ta sẽ tính toán lượng sát thương lớn nhất có thể gây ra mà vẫn đảm bảo người đi lượt đầu thắng bằng cách quy hoạch động lần nữa để tìm đường đi dài nhất từ đỉnh h về đỉnh 0 (mỗi đỉnh là lượng máu còn lại của quái vật). Để đảm bảo người chơi đầu tiên chiến thắng, đồ thị có thêm điều kiện nếu tại đỉnh u mà $win[u] = 1$, thì chỉ được thăm những đỉnh v có $win[v] = 0$.

Độ phức tạp: $O(H * n)$

Mã nguồn tham khảo

- C++: https://ideone.com/FNx0jY