Câu 4: Tìm kiếm

Subtask 1:

- Ta sẽ thử từng số trong mảng m * n xem liệu có phải nó là số lớn thứ k hay không. Với mỗi số ta sẽ đếm số các số nhỏ hơn chúng trong mảng. Nếu số lượng đó nhỏ hơn k, thì chúng **có thể** là số lớn thứ k thôi. Kết quả phải là số lớn nhất trong tất cả các số trên.
- Đpt: $O(m^2 \times n^2)$

Subtask 2:

- Ta sẽ ghi hết tất cả các giá trị trong bảng ra và sort tăng dần. Việc của bạn là truy vấn số lớn thứ k thôi.
- Đpt: $O(m \times n \times \log(m \times n))$

Subtask 3:

- Ta sẽ chặt nhị phân kết quả để tìm được số lớn thứ k. Với mỗi giá trị khi chặt, giả sử là x, ta sẽ đếm các số nhỏ hơn x. Hay nói các khác, đếm số cặp (i, j) sao cho:
  - $1 \leq i \leq n, 1 \leq j \leq m$
  - $i \times j < x$
- Nhìn thì công thức trên có vẻ "khoai", nhưng không đâu. Ta có thể duyệt một vòng i hoặc j rồi tính thành phần còn lại bằng phép chia mà.
- Cuối cùng, kết quả sẽ là giá trị x lớn nhất thỏa mãn có số lượng các số bé hơn x nhỏ hơn k.
- Đpt: $O(n \times \log(n \times m))$

Câu 5: Rừng sâu

Subtask 1:

- Ta đơn giản là chỉ duyệt trâu BFS/DFS với mỗi truy vấn để đếm kết quả
- Đpt: $O(n^2 \times m)$

Subtask 2:

- Nhận thấy việc tính đi tính lại rất mất thời gian, nên ta có thể sử dụng mảng hai chiều đường đi có trọng số lớn nhất trên tất cả các đường. Sau đó có thể dùng tổng tiền tố hoặc chặt nhị phân để đếm kết quả với mỗi truy vấn.
- Đpt: $O(n^2 + m)$;

Subtask 3:

- Có thể dùng DSU để đếm kết quả với từng truy vấn.
- Trước hết, ta sẽ sắp xếp các trọng số của các cạnh và trọng số của các truy vấn vào cùng một mảng (nhớ lưu thêm cả vị trí để in ra kết quả). Sau đó, ta sẽ merge tất cả các cạnh có trọng số nhỏ hơn truy vấn hiện tại, rồi tính.
- Trong khi merge hai thành phần liên thông, chẳng hạn số lượng là x, y, ta sẽ trừ kết quả tổng cho $\frac{x(x-1)}{2} + \frac{y(y-1)}{2}$ và cộng với $\frac{(x+y)(x+y-1)}{2}$. Chi tiết xem trong code để hiểu rõ hơn

- Đpt: $O(m \times log\, n)$

Câu 6: Dãy đặc biệt

Subtask 1:

- Ta chỉ cần sinh nhị phân tất cả các trường hợp rồi duyệt để tính kết quả
- Đpt: $O(2^n \times n)$

Subtask 2:

- Ta sẽ dùng công thức quy hoạch động để tính kết quả.
- Gọi f[i][j] là số cách sắp xếp i ký tự đầu tiên thỏa mãn đề bài, và ở cuối xâu đang có j ký tự 1 liên tiếp.
- Ta có cách tính như sau:
    - $f[i+1][0] = f[i][0] + f[i][1] + \cdots + f[i][k]$
    - $f[i+1][j+1] = f[i][j]$
- Đpt: $O(n \times k)$

Subtask 3:

- Từ công thức trên, ta sẽ chuyển thành nhân ma trận để tính toán các giá trị
- Đpt: $O(k^3 \times \log(n))$;