Thi thử TST2022 - Ngày 1

Hướng dẫn giải

17-03-2022

Bài 1. FAKECOIN

Bài 2. THELIGHT

Bài 3. BEAUTIPERM

Bài 1. FAKECOIN

Bài 2. THELIGHT

Bài 3. BEAUTIPERM

Bài 1. Tìm đồng xu giả — FAKECOIN

Có n đồng xu được đánh số thứ tự từ 1 đến n. Trong số đó có một đồng xu là giả, đồng xu giả có thể nặng hơn hoặc nhẹ hơn các đồng xu khác. Những đồng xu thật có cùng một trọng lượng.
Yêu cầu: xác định đồng xu giả bằng số lần cân giới hạn.

Bài 1. Tìm đồng xu giả — FAKECOIN

Trước hết ta giải bài toán phụ FAKECOIN1 sau:

FAKECOIN1: Tập A chứa 40 xu có nghi vấn, Tập B chứa 40 xu thật, cân ít lần nhất để tìm ra xu giả

- Chọn $3^3 = 27$ xu A cân với 27 xu B,
- nếu khác nhau bài đưa về 3^3 xu cân 3 lần biết trước nặng nhẹ,
- nếu không đưa về cân 3 lần với 13 xu chưa biết nặng nhẹ.
- Tiếp tục với cân $3^2 = 9$ bên A với 9 bên B, còn lại 4 xu nghi vấn . . . .

Bài 1. Tìm đồng xu giả — FAKECOIN

- $f(i)$: số đồng xu tối đa chỉ cần cân bởi i lần cân và có trước 3^{i-1} đồng xu thật
  $f(2) = 4$, cần 2 lần cân
  $f(3) = 3^2 + f(2) = 13$, cần 3 lần cân, dùng 3 đồng xu thật mỗi bên để test
  $f(4) = 3^3 + f(3) = 40$, cần 4 lần cân
  $f(i) = 3^{i-1} + f(i-1)$, ==> bài fakecoin1!
- Giải công thức truy hồi suy ra $f(i) = (3^i - 1)/2$ cần i lần cân cho bài FAKECOIN1. Suy ra bài FAKECOIN cần $(i+1)$ lần cân cho $3 * f(i) = (3^{i+1} - 1)/2$ xu.

Bài 1. Tìm đồng xu giả — FAKECOIN

Thuật toán chung:

- Cân 1 lần A:$f(i)$ B:$f(i)$ sẽ đưa về bài fakecoin1 với $f(i)$ đồng xu.
- fakecoin1: cân A:3^{i-1} và B:3^{i-1} sẽ đưa về 1 trong 2 trường hợp:
  - Bằng nhau sẽ đưa về bài $f(i-1)$
  - Lệch nhau: đưa về bài 3^{i-1} khi biết trước nặng nhẹ –> cần $(i-1)$ lần cân (mỗi lần cân chia 3).

Thuật toán dựa trên việc cân cho 12 xu:

- Bước 1: chọn cân A: 1 2 3 4 với B: 5 6 7 8, nếu A<B (trường hợp khó nhất, các cái khác tương tự hoặc dễ hơn.
- Bước 2: chọn cân A: 1 9 10 11 với B: 2 3 4 5,
  - nếu A<B : suy ra xu 1 là giả nhẹ hoặc xu 5 là giả nặng.
  - nếu A>B: suy ra 2 3 4 có xu nhẹ

Bài 1. FAKECOIN

Bài 2. THELIGHT

Bài 3. BEAUTIPERM

Ánh sáng — THELIGHT

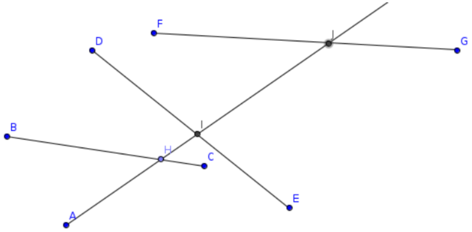
Tính diện tích phần đa giác được chiếu sáng bởi một nguồn sáng nằm bên trong.

Subtask 1: đa giác lồi

- Một đa giác được coi là lồi khi và chỉ khi lấy hai điểm nằm hoàn toàn trong đa giác, đoạn thẳng nối hai điểm đó cũng nằm hoàn toàn trong đa giác.
- Như vậy việc điểm sáng được đặt ở đâu không quan trọng, đáp án luôn là diện tích đa giác được cho.

Subtask 2: đa giác lõm

- Từ điểm sáng được cho, vẽ một tia sáng rồi quét tia sáng đó quanh điểm sáng ban đầu.
- Trong quá trình quét, duy trì một tập S chứa các cạnh của đa giác cắt tia sáng đang được xét, các đoạn trong tập được sắp xếp theo khoảng cách giữa đoạn thẳng và điểm sáng tính theo tia sáng được xét.
- Ví dụ, ở hình dưới đây (điểm sáng A, tia sáng AH, các đoạn BC, DE, FG là các cạnh cắt tia sáng), các đoạn trong tập được sắp xếp theo thứ tự BC, DE, FG vì AH < AI < AJ.

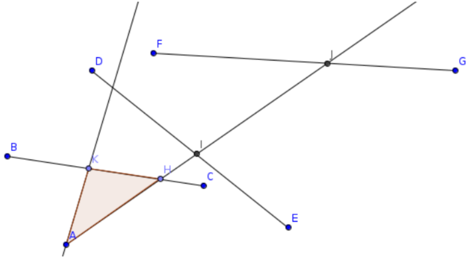


Subtask 2: đa giác lõm

- Do đa giác không tự cắt, thứ tự giữa hai đoạn bất kì trong tập S sẽ không bao giờ thay đổi. Nói cách khác, nếu đoạn a nằm trước đoạn b trong tập S ở một thời điểm nào đó thì đoạn b sẽ không bao giờ có thể ở trước đoạn a trong quá trình quét.
- Từ tính chất này, ta chỉ cần dùng cấu trúc dữ liệu set với một hàm so sánh tự viết là có thể duy trì được tập S.
- Ta thấy trong các khoảng thời gian trong quá trình quét mà đoạn đầu tiên của tập S không thay đổi thì ta có thể dễ dàng tính được phần diện tích được chiếu sáng trong khoảng đó.

Subtask 2: đa giác lõm

- Ví dụ như khi ta quét từ tia AH đến tia AK ở hình dưới đây thì phần diện tích chiếu sáng được cộng vào đáp án là diện tích tam giác AKH. Ta có thể dễ dàng tính được diện tích đa giác này.



- Vậy thì phần tử đoạn đầu tiên trong tập S thay đổi khi nào? Đó là khi mà đường quét của ta chạm vào một đỉnh nào đó của đa giác. Khi đó, sẽ có một số đoạn phải bị loại bỏ khỏi tập S, và một số đoạn có thể được thêm vào tập S và rất có thể ta có thể có một đoạn đầu tiên mới.

Subtask 2: đa giác lõm

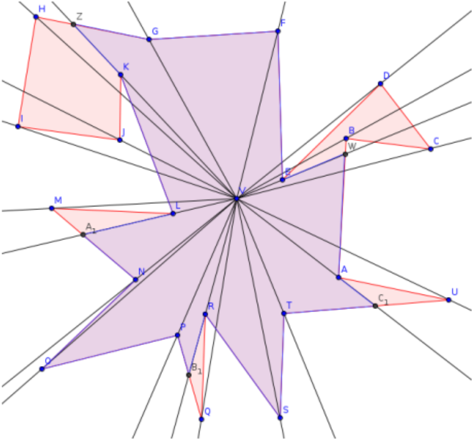
Ta có thuật toán như sau:

- Sắp xếp các đỉnh của đa giác theo thứ tự ngược chiều kim đồng hồ.
- Xét mỗi đỉnh của đa giác theo thứ tự đã sắp xếp
  - Cập nhật tập S như đã mô tả ở trên;
  - Tính phần diện tích chiếu sáng thêm và cộng vào kết quả.

Kĩ thuật duy trì tập S có tính chất như trên được gọi là Z-buffering

Subtask 2: đa giác lõm

Dưới đây là một hình mô phỏng lại thuật toán trên. Khoảng giữa hai tia sáng liên tiếp trong hình là một khoảng mà đoạn đầu tiên trong tập S không thay đổi. Phần diện tích màu xanh tím là phần diện tích được chiếu sáng.



Bài 1. FAKECOIN

Bài 2. THELIGHT

Bài 3. BEAUTIPERM

Hoán vị đẹp — BEAUTIPERM

- Đếm số lượng hoán vị đẹp thỏa mãn tất cả ràng buộc.
- Một hoán vị đẹp p là hoán vị tồn tại một cặp số (i, j) thỏa mãn $i < j$ và $p_i > j$ và $p_j > i$, ta gọi (i, j) là cặp số đẹp.
- Ta đưa về bài toán bù đó là tính số lượng hoán vị không tồn tại cặp đẹp nào. Khi đó với mỗi cặp (i, j), $i < j$ ta có hoặc $p[i] \leq j$ hoặc $p[j] \leq i$.

Hoán vị đẹp — BEAUTIPERM

Nhận xét 1: $p[i] \leq i + 2, \forall i$

Chứng minh:

- Giả sử tồn tại i sao cho $p[i] \geq i + 3$.
- Ta xét mọi j từ 1 đến $i + 2$ (trừ $j = i$ ra), như vậy có $i + 1$ giá trị khác nhau của j.
- Mà $p[i] > j$ nên $p[j] \leq i$, do đó $p[j]$ chỉ có thể chứa i giá trj khác nhau, mâu thuẫn!
- Vậy $p[i] \leq i + 2, \forall i$.

Hoán vị đẹp — BEAUTIPERM

Nhận xét 2: Xét một hoán vị p thỏa mãn $p[i] \leq i + 2$. Khi đó p có một cặp số đẹp (i, j) khi và chỉ khi i và j là 2 số nguyên liên tiếp.

Chứng minh:

- Xét cặp số đẹp (i, j). Khi đó $p[i] > j$, $p[j] > i$, $p[i] \leq i + 2$, $p[j] \leq j + 2$.
- Từ đó ta có $i < p[j] \leq j + 2$.
- Suy ra $i < j + 2$, hay $i \leq j + 1$.
- Tương tự ta cũng có $j \leq i + 1$.
- Từ $i \leq j + 1$ và $j \leq i + 1$ ta có i và j là 2 số nguyên liên tiếp.

Subtask 1: $n \le 8$

Thử 8! hoán vị và kiếm tra từng hoán vị có thỏa mãn đề bài hay không

Subtask 2: $n \leq 10^6$

- Gọi $dp[i][0]$ là số cách điền các phần tử $1 \dots i$ của hoán vị thỏa mãn $p[i] \neq i + 2$.
- Gọi $dp[i][1]$ là số cách điền các phần tử $1 \dots i$ của hoán vị thỏa mãn $p[i] = i + 2$.
- Từ đây ta sẽ chuyển được trạng thái dựa vào 2 nhận xét phía trên. Ở mỗi vị trí i, ta sẽ tính xem có bao nhiêu cách điền thỏa mãn để chuyển được các trạng thái.

Subtask 3: $n \leq 10^9$

- Subtask 2 có thể được tính nhanh bằng cách nhân ma trận, tuy nhiên ma trận có mỗi vị trí có thể khác nhau.
- Nhận xét là có một số vị trí mà bắt đầu từ đó thì ma trận chuyển trạng thái thay đổi giá trị. Các vị trí đó nằm ở các điểm $(x-2, x-1, x, x+1, x+2), (v-2, v-1, v, v+1, v+2), n-1, n, \ldots$
- Ta liệt kê các đoạn này ra và tính ma trận cho từng đoạn đó. Sau đó kết hợp nhân và lũy thừa nhanh ma trận để chuyển trạng thái nhanh.
- Ngoài ra ta cần phải tính $n!$ (n lên đến $1e9$), có thể tính bằng bảng hằng, lưu lại hằng số các giá trị giai thừa của $0, 1e7, 2e7, \ldots$. Như vậy có thể tính $n!$ trong $1e7$ phép tính.
- Độ phức tạp cuối cùng $O(m \log n)$ (ngoài ra tốn 10^7 phép tính để tính $n!$ bằng mảng hằng).