Thi thử TST2022 - Ngày 3

Hướng dẫn giải

10-04-2022

Trạm xăng — FUEL

- Tiến có một chiếc xe máy kỳ diệu, có thể chứa được lượng xăng tùy thích và cứ 1 lít xăng đi được 1 km. Trước khi rời khỏi nhà, Tiến đã đổ vào chiếc xe của mình F lít xăng.
- Tiến muốn đi thăm một người bạn ở cách Tiến D km. Trên đường đi có N trạm xăng để Tiến đổ thêm xăng; trạm xăng thứ i cách nhà Tiến X_i km. Ở trạm xăng thứ i, Tiến có thể nạp thêm A_i lít xăng, nhưng chỉ khi lượng xăng hiện tại trên xe không quá B_i.
- **Yêu cầu:** Hãy tìm lượng xăng ban đầu F nhỏ nhất sao cho Tiến có thể có đủ xăng để đến thăm bạn của mình.

Subtask 1 (50%): $N \leq 10^4, D \leq 10^9$

- Sắp xếp các trạm xăng theo thứ tự tăng dần của B_i.
- Nhận xét rằng với mọi F nằm trong khoảng $[B_{i-d1} + 1, B_i]$, số trạm xăng có thể dùng là như nhau. Vì vậy, để tìm F nhỏ nhất thuộc đoạn trên sao cho có thể đi hết được đoạn đường, ta có thể đặt $F = B_i$, tìm F_{rem} là lượng xăng nhỏ nhất còn lại trong bình trong suốt quãng đường đi.
- Đáp án là $F - F_{rem}$.
- Độ phức tạp: $O(n^2)$.

Subtask 2 (50%): ($N \leq 3 \times 10^5, D \leq 10^9$

- Sắp xếp các trạm xăng theo thứ tự tăng dần của B_i.
- Đầu tiên, cho $F = B_1$, khi đó ta có thể sử dụng tất cả các trạm xăng. Với mỗi trạm xăng, ta tính xem nếu sử dụng tất cả các trạm xăng trước đó thì khi đến trạm xăng đó, trong bình còn bao nhiêu lít xăng (trước khi đổ thêm xăng).
- Với trạm xăng thứ i tính từ trái qua phải, gọi giá trị này là R_i.
- Nếu tất cả R_i đều không âm thì đáp án là $F - F_{rem}$ như trong subtask 1;
- Nếu không, đặt $F = B_2$. Khi đó, ta tăng các giá trị R_i lên $B_2 - B_1$ bằng lazy propagation. Tuy nhiên, lại có một trạm xăng không sử dụng được, gọi chỉ số trạm xăng này là p; khi đó, ta phải trừ bớt A_p khỏi $R_{p+1}, R_{p+2}, ..., R_n$.
- Nếu $F = B_2$ thỏa mãn, ta dễ dàng tính được đáp án. Nếu không, ta cho $F = B_3$ và lặp lại quá trình trên.
- Trong trường hợp tất cả các $F = B_i$ đều không thỏa mãn, đáp án là độ dài quãng đường đi.
- Độ phức tạp: $O(n \log n)$.

Bài 1. FUEL

Bài 2. SSUBSTR

Bài 3. POLYGAME

SSUBSTR

- Cho N xâu không rỗng gồm các chữ cái latin viết thường $s_1, s_2, \ldots, s_n$.
- Gọi S_i là tập các xâu con liên tiếp của s_i. Ví dụ xâu $"xyz"$ có tập các xâu con liên tiếp là $\{"x", "y", "z", "xy", "yz", "xyz"\}$, xâu $"aa"$ có tập các xâu con liên tiếp là $\{"a", "aa"\}$.
- **Yêu cầu:** Gọi t_i là số phần tử thuộc S_i nhưng không thuộc bất cứ tập S_j nào mà $j \neq i$. Hãy tính mảng t.

Subtask 1: Tổng độ dài các xâu s_i không vượt quá 2000

Sử dụng hash kết hợp với set. Độ phức tạp là $O(n^2 \log n)$.

Subtask 2: $N = 1, \sum |s_i| \leq 10^6$

Áp dụng suffix array cơ bản.

- Đếm số lượng xâu con liên tiếp phân biệt của s.
- Gọi $sa[i]$ là suffix array bé thứ i.
- $Lcp[i]$ là longest common prefix của $sa[i]$ và $sa[i+1]$.
- Đáp số là $sum(len[sa[i]], i$ from 0 to $len-1) - sum(lcp[i], i$ from 0 to $n-2)$.

Subtask 3: $N \leq 10^5, \sum |s_i| \leq 10^6$

- Tạo 1 xâu mới bằng cách ghép các xâu lại với nhau, phân cách bởi một kí tự bé hơn tất các các chữ trong bảng chữ cái, gọi là '?'. Khi đó xâu mới có dạng $S = s1?s2?s3?...sn?$.
- Ta sẽ tìm suffix array của S. Xét suffix array $sa[i]$ của s. Ta xác định được $sa[i]$ có kí tự bắt đầu thuộc xâu nào.
- Gọi các string thuộc Si mà không thuộc Sj nào với mọi j khác i là các signature string của si. Từ đó có thể đếm số "signature substring" thuộc xâu đó mà là tiền tố của $sa[i]$.
- Gọi xâu gốc của một suffix là xâu đề bài cho chứa phần tử đầu tiên của suffix đó.
- Với mỗi suffix S trong suffix array, ta sẽ tìm được suffix L gần nhất bên trái S mà S và L có xâu gốc khác nhau.
- Khi đó ta sẽ loại bỏ được các các prefix của S mà cũng là prefix của L khỏi tập các signature string.
- Ngoài ra xét suffix R ngay bên phải S trong suffix array, ta loại bỏ tất cả prefix của S mà cũng là prefix của R.
- Từ đó đếm được số "signature string" mà S đóng góp cho xâu gốc của S.

Bài 1. FUEL

Bài 2. SSUBSTR

Bài 3. POLYGAME

POLYGAME

Có 2 người chơi 1 trò chơi trên đa giác n đỉnh như sau: Hai người luân phiên thực hiện lượt chơi. Tại mỗi lượt, người chơi nối thêm 1 đường chéo cho đa giác, sao cho nó không cắt bất cứ đường nào đã nối trước đó (vẫn được phép chung đầu mút, nhưng không được trùng với bất kỳ cạnh hay đường chéo đã nối nào của đa giác) và không tạo thành bất cứ tam giác nào (cụ thể hơn, không xuất hiện 3 đỉnh nào của đa giác đôi một có cạnh nối). Ai không thực hiện được lượt đi nữa thì thua cuộc. Yêu cầu viết chương trình chơi trực tiếp với máy chấm.

POLYGAME

- Subtask 1: n chẵn, người chơi trước, kẻ đường chéo ở giữa, máy chơi thế nào thì người chơi giống hệt thế nửa còn lại.
- Subtask 2: n nhỏ, duyệt hết mọi khả năng chơi tìm phương án chắc chắn thắng.
- Subtask 3: n vừa, áp dụng lý thuyết trò chơi tổng Nim, tổng xor các hàm bằng 0 thì luôn thắng. Không cần lưu đường chéo vào CTDL, một lượt for lại để xác định vị trí đường chéo, DPT $O(n^2)$.
- Subtask 4: n lớn, đầu tiên phát hiện quy luật lặp lại chu kỳ 34 của hàm Grundy, tiếp theo lưu các đường chéo lên CTDL 2 chiều, mỗi đường chéo đặc chưng bởi hai chỉ số x, y (giống dạng các dấu ngoặc lồng nhau), một đa giác là nằm trong khoảng 2 dấu ngoặc hợp lệ liên tiếp. Do là cấu trúc động nên một chiều sử dụng BIT, chiều còn lại có thể sử dụng cây Trie tĩnh để lưu và truy vấn.

POLYGAME

Hàm grundy ứng với trò chơi này được tính như sau:
g(n)=MEX({g(x) xor g(n+2-x), 3 < x < n-1})
Ở đây n là số đỉnh của đa giác, $MEX(S)$ là số tự nhiên nhỏ nhất không nằm trong tập S.
Ví dụ: $g(0) = g(1) = ... = g(5) = MEX(NULL) = 0$;
$g(6) = MEX(\{0\}) = 1$

POLYGAME

Người chơi trước thắng khi và chỉ khi $g(n)$ khác 0, nghĩa là trong tập MEX tồn tại giá trị 0, vì vậy chọn chơi sao cho $g(n) = 0$. Chiến thuật chơi như sau:

- Tại mỗi thời điểm, ta có 1 tập các đa giác mà có thể chơi độc lập với nhau do mỗi lần lấy đường chéo sinh ra thêm (vì không được nối các đường chéo cắt nhau). Gọi X là tập các đa giác đang có, ta chỉ quan tâm đến số đỉnh của đa giác, do đó X có thể coi là một multiset<int>
- Đặt G=XOR(g(n), n thuộc X)
- Nếu G=0, ta thua dù chiến thuật tốt thế nào
- Nếu G khác 0, khi đó luôn tồn tại n thuộc X thỏa mãn (G xor g(n)) < g(n). Ta sẽ nối 1 đường chéo cho hình này:
  - Đặt k=G xor g(n)
  - Đặt S={g(x) xor g(n+2-x), 3 < x < n-1}
  - Do MEX(S)=g(n) nên mọi số < g(n) đều thuộc S, do đó k thuộc S, tức tồn tại x: 3 < x < n-1, g(x) xor g(n+2-x)=k
  - Ta nối đỉnh 1 của đa giác này với đỉnh x

Tính đúng đắn chứng minh từ lý thuyết trò chơi: Tổng của các trò chơi

Để tính được hàm g(n), ta mất độ phức tạp $O(n^2)$.
Ta thấy số đường nối thêm là không quá n/2, tại vì: Nếu có k đường được nối thì sẽ có k+1 đa giác (rời nhau về diện tích), 2 đa giác bất kỳ đều chung nhau không quá 2 điểm, tức sẽ rời nhau ít nhất 2 điểm, vậy sẽ có nhiều hơn (k+1)*2 điểm. Ngoài ra, thực nghiệm thấy rằng g(n)<10, nên việc chọn đường để nối theo chiến thuật có thể làm nhanh trong O(n), do đó độ phức tạp của toàn bài là $O(n^2)$
Tài liệu tham khảo:
https://manhhomienbienthuy.github.io/2017/May/02/algorithm-sprague-grundy-theorem.html