Đáp án PREVOI 2012. Hải Phòng.

SEARCH

Phương pháp được đề xuất dựa trên kỹ thuật đánh dấu:

Duyệt các phần tử mảng B, đánh dấu $bmark[v] = True$ nếu $v \in B$ $(O(n))$

Sort lại mảng A bằng đếm phân phối, loại các phần tử trùng nhau $(O(m))$

Trong mảng C, loại bỏ các phần tử $\in B$ dựa trên mảng đánh dấu $bmark$. $(O(p))$

Khởi tạo mảng $cmark[1 \dots 2.10^5] =$ False, xét các đoạn gồm các phần tử liên tiếp trong C chưa bị loại bỏ, với mỗi đoạn độ dài l:

- Duyệt các phần tử trong đoạn, mỗi giá trị v duyệt qua sẽ đánh dấu $cmark[v] =$ True cho biết giá trị v thuộc đoạn $(O(l))$
- Duyệt các phần tử của A, chú ý rằng các phần tử của A đã hoàn toàn phân biệt, nếu mọi phần tử $a[i]$ đều thuộc đoạn ($cmark[a[i]] =$ True) thì ghi nhận để cập nhật độ dài đoạn dài nhất. $O(\min(m, l)) = O(l)$
- Xét các phần tử trong đoạn, mỗi giá trị v duyệt qua sẽ đánh dấu lại $cmark[v] =$ False để đặt lại mảng $cmark$ thành False hết (không dùng FillChar hay memset) $(O(l))$

Chi phí xử lý đoạn độ dài l là $O(l)$ nên chi phí xử lý tất cả các đoạn là $O(p)$.

Độ phức tạp $O(m + n + p)$

Loại giải thuật: ad-hoc

CALC

$$\frac{a}{m} - \frac{b}{n} = \frac{a-b}{m-n}$$

Quy đồng mẫu số:

$$(an - bm)(m - n) = (a - b)mn$$
$$\underline{amn} - an^2 - bm^2 + bmn = \underline{amn} - bmn$$
$$a = \frac{2bmn - bm^2}{n^2} = \frac{bm(2n - m)}{n^2}$$

Từ $a \geq 0$ nên $m \leq 2n$, các giá trị $m \in [1,2n]$ sao cho $bm(2n - m)$ chia hết cho n^2 sẽ cho ta một giá trị duy nhất a. Thuật toán đơn giản là duyệt các giá trị $m \in [1,2n]$ và đếm các giá trị m sao cho $bm(2n - m)$ chia hết cho n^2.

Loại giải thuật: arithmetic

NUCLEAR

Nếu A là tập những hộ gia đình nằm trong phạm vi nguy hiểm của nhà máy thứ nhất và B là tập những hộ gia đình nằm trong phạm vi nguy hiểm của nhà máy thứ hai thì từ công thức đếm bù trừ:

$$|A \cup B| = |A| + |B| - |A \cap B|$$

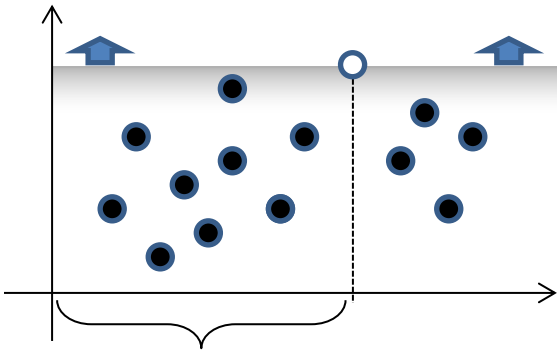
Ta thấy rằng với mỗi truy vấn, ta chỉ cần biết số hộ gia đình nằm trong phạm vi nguy hiểm của cả hai nhà máy ($|A \cap B|$) là xong, vì việc tính $|A|$ và $|B|$ có thể thực hiện dễ dàng trong thời gian $O(1)$ sau phép tiền xử lý bằng phép đếm tích lũy.

Với mỗi hộ gia đình cho ứng với một cặp (p, q), ở đây p và q lần lượt là khoảng cách từ hộ gia đình tới nhà máy thứ nhất và nhà máy thứ hai. Nếu các khoảng cách không nguyên, ta làm tròn lên (lấy trần – ceiling) để được số nguyên cho dễ xử lý.

Coi mỗi cặp (p, q) là tọa độ một điểm đen trên mặt phẳng và mỗi truy vấn (R_1, R_2) là tọa độ một điểm trắng trên mặt phẳng. Có tất cả $n + q$ điểm.

Ban đầu số điểm đen tại mọi hoành độ được khởi tạo bằng 0.

Quét các điểm từ dưới lên (tung độ tăng dần), các điểm cùng tung độ thì xét theo hoành độ tăng dần, nếu các điểm cùng tung độ và hoành độ thì điểm đen xếp trước điểm trắng.



Khi gặp một điểm đen (p, q) ứng với một hộ gia đình ta tăng số điểm đen tại hoành độ p lên 1

Khi gặp một điểm trắng (R_1, R_2) ứng với một truy vấn, ta đếm số điểm đen có hoành độ $\leq R_1$, đây chính là số hộ gia đình nằm trong phạm vi nguy hiểm của cả hai nhà máy với bán kính nguy hiểm là R_1 và R_2.

Việc tăng số điểm đen tại một hoành độ và truy vấn số điểm đen tại các hoành độ $\leq R_1$ có thể thực hiện hiệu quả bằng một cấu trúc dữ liệu truy vấn phạm vi, chẳng hạn Fenwick trees (BIT).

Độ phức tạp $O\big((n + q) \log maxR\big)$ với $maxR$ là giá trị cực đại của R_1 (2.10^5). Nếu tính cả chi phí sắp xếp để thực hiện quét các điểm theo thứ tự thì cần cộng thêm $O\big((n + q) \log(n + q)\big)$ nữa.

Loại giải thuật: line sweeping, range query, advanced data structures.

HIRE

Quy hoạch động theo hàm mục tiêu $f[x]$ là chi phí tối thiểu để đi xe từ ngày x tới ngày n mà ngày x ta có thuê xe. Đáp số là $f[1]$

Dễ thấy rằng $f[x] = p[x]$ nếu $t_x = n$ tức là khi xe thuê ngày x có thể dùng tới hết ngày n.

Nếu $t_x \neq n$, xe x không thể dùng tới ngày n thì sau khi thuê xe tại ngày x ta phải thuê thêm một xe khác tại ngày x' nào đó mà $x' \in [x+1, t_x+1]$, tức đó có công thức truy hồi:

$$f[x] = \min_{x' \in [x+1, t_x+1]} \{f[x']\} + p[x]$$

Thuật toán đơn giản tính min bằng cách duyệt tất cả các phần tử thuộc tập có độ phức tạp $O(n^2)$

Mỗi truy vấn min trong một phạm vi có thể sử dụng cây segment trees để trả lời trong thời gian $O(\log n)$, khi đó độ phức tạp của thuật toán giảm xuống cỡ $O(n \log n)$.

Dùng thêm một nhận xét: Nếu $x < x'$ và $f[x] \leq f[x']$ thì sau khi tính $f[x]$, $f[x']$ sẽ không bao giờ tham gia quá trình tính toán nữa (vì nếu x' thuộc phạm vi truy vấn thì x cũng sẽ thuộc phạm vi truy vấn), dùng một ngăn xếp loại bỏ những điểm x' như vậy và dùng cấu trúc Disjoint-set forest để nhập x' với x, ta có thể giảm độ phức tạp xuống $O(n\alpha(n))$.

Loại giải thuật: Dynamic programming.

TRIP

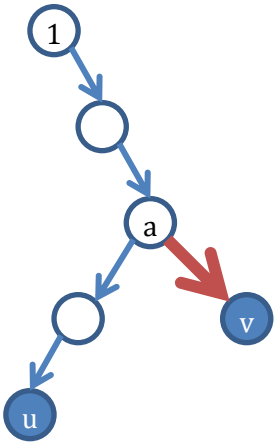
Dựng mô hình đồ thị dạng cây có gốc ở đỉnh 1, Dùng DFS từ 1 gán cho mỗi đỉnh một mã số là số thứ tự của đỉnh đó theo quá trình thăm DFS. Gọi mã số của đỉnh u là $number[u]$ và độ sâu của u là $depth[u]$.

Gọi $v_1, v_2, \dots, v_k$ là một dãy các đỉnh xếp theo thứ tự tăng dần của mã số. Giả sử một người muốn đi thăm các địa điểm thuộc tập $S = \{v_1, v_2, \dots, v_k\}$ thì những nhánh cây không chứa đỉnh nào $\in S$ có thể tỉa bỏ để được một cây con T. Dễ thấy rằng từ đỉnh 1, để thăm mọi điểm $\in S$ thì hành trình buộc phải qua mọi đỉnh của cây con T này. Vì hành trình là một chu trình nên mỗi cạnh (u, v) trên T sẽ phải đi qua ít nhất 2 lần, một lần $u \to v$ và một lần $v \to u$. Suy ra mọi chu trình qua tất cả các đỉnh phải có độ dài ≥ 2 lần số cạnh của T.

Thay mỗi cạnh trên cây con bằng 2 cung có hướng ngược chiều, ta được một đồ thị Euler (Chu trình Euler trong trường hợp này có thể tìm bằng DFS). Chu trình Euler có độ dài đúng bằng 2 lần số cạnh của T nên đây chính là chu trình ngắn nhất cần tìm.

Vấn đề còn lại là xác định số cạnh của cây T. Coi như các cạnh ban đầu đều chưa tô màu. Duyệt các đỉnh $\in S$ theo thứ tự tăng dần của $number[.]$. Khi xét một đỉnh u, ta đi từ 1 xuống u và đi qua cạnh nào ta tô màu luôn cạnh đó nếu nó chưa được tô, khi duyệt qua mọi đỉnh của S thì các cạnh đã tô màu cho ta cây T và số lần tô màu chính là số cạnh của T.

Nhắc lại rằng ta duyệt các đỉnh của S theo thứ tự tăng dần của $number[.]$. Giả sử ta vừa đi từ 1 xuống u và tô màu xong các cạnh trên đường đi, sau đó đi từ 1 tới v để tô màu tiếp. Gọi a là tiền bối chung thấp nhất của cả u và v ($LCA(u, v)$).



Khi đi từ 1 xuống v, đoạn đường từ 1 tới a không có cạnh nào phải tô vì những cạnh đó đã tô khi ta đi từ 1 xuống u, đoạn đường từ a tới v qua cạnh nào ta cũng phải tô cạnh đó, suy ra số cạnh phải tô khi đi từ 1 tới v đúng bằng $depth[v] - depth[a]$. Từ đó suy ra:

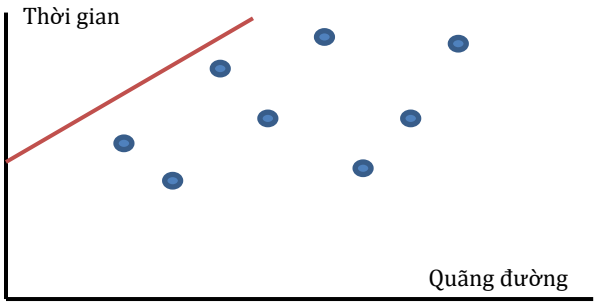
Cây T nhỏ nhất chứa đỉnh 1 và các đỉnh $v_1, v_2, \dots, v_k$ có số cạnh bằng:

$$\sum_{i=1}^{k} depth[v_i] - \sum_{i=2}^{k} depth[LCA(v_i, v_{i-1})]$$

Giá trị này có thể tính trong thời gian $O(k)$ hay $O(k \log n)$ tùy theo thuật toán tìm LCA được sử dụng,

RIDER

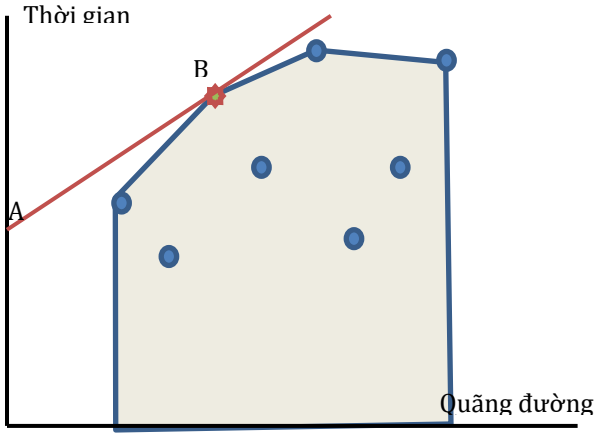
Xét mặt phẳng tọa độ với Ox là trục quãng đường và Oy là trục thời gian. Mỗi lễ hội i có thể coi như một điểm (i, t_i)



Nếu đi từ đầu đường tại thời điểm s bằng chuyển động đều thì có thể mô tả đồ thị chuyển động như một đoạn thẳng đi qua điểm $(0, s)$ (màu đỏ trong hình vẽ), một điểm (x, y) trên đoạn thẳng cho biết khi đi được x km thì đồng hồ chỉ thời điểm y. Muốn không bị vướng lễ hội thì mọi điểm (i, t_i) ứng với lễ hội đều không được nằm phía trên đường thẳng này. (Chú ý đường thẳng đồ thị chuyển động có hệ số góc luôn dương, hệ số góc càng nhỏ thì ứng với tốc độ càng lớn)

Để tìm hệ số góc nhỏ nhất thỏa mãn, trước hết ta tìm bao lồi của tập điểm ứng với các lễ hội bổ sung thêm hai điểm $(1,0)$ và $(L, 0)$.

Từ điểm $A(0, s)$, nhìn dọc bao lồi theo chiều kim đồng hồ từ điểm $(1,0)$ tới điểm $(L, 0)$ đầu tiên hướng nhìn của ta là từ phải qua trái sau đó lại từ trái qua phải, đoạn thẳng nối từ A tới điểm chuyển hướng B đó chính là đồ thị chuyển động có tốc độ nhanh nhất. Điểm P có thể tìm bằng thuật toán tìm kiếm nhị phân với độ phức tạp O$(\log m')$ (với m' là số đỉnh trên bao lồi, $m' < m$).



Vì ta phải giải quyết bài toán cho n ngày, bao lồi có thể coi như một khâu tiền xử lý, đáp số cho mỗi ngày có thể trả lời trong thời gian O$(\log m)$ và toàn bộ đáp số cho n ngày có thể trả lời trong thời gian O$(n \log m)$.

Loại giải thuật: Geometry, Convex Hull, Binary searching.