Thực hành TTUD

Buổi 1: Các thuật toán và cấu trúc dữ liệu cơ bản

Bài 1. Kiểm tra và phân tích dữ liệu log cuộc gọi thoại

Một nhà mạng muốn thực hiện truy vấn dữ liệu log lịch sử cuộc gọi trong ngày, dữ liệu log này được format dạng

call <from_number> <to_number> <date> <from_time> <end_time>

Ý nghĩa của các trường là

- Từ khóa call: đây là log cuộc gọi điện thoại
- <from_number> và <to_number>: là SDT gọi và nhận cuộc gọi, là kiểu xâu ký tự độ dài 10 (chỉ gồm các chữ số 0-9)
- <date>: Là ngày thực hiện cuộc gọi theo định dạng YYYY-MM-DD (VD. 2022-10-21)
- <from_time> và <end_time>: Là thời gian bắt đầu, kết thúc cuộc gọi trong ngày (định dạng theo hh:mm:ss, VD. 10:07:23)

Chú ý:

- Số lượng log cuộc gọi này có thể lớn tới 100000 dòng
- Các tham số ngăn cách với nhau bởi 1 dấu cách trống

Các truy vấn dữ liệu log được đưa vào với định dạng bắt đầu bằng dấu ?, trong đó

- **?check_phone_number**: in ra màn hình (dòng mới) giá trị 1 nếu các số điện thoại đều hợp lệ
- **?number_calls_from <phone_number>**: in ra màn hình (dòng mới) số cuộc gọi được xuất phát từ SDT <phone_number>
- **?number_total_calls**: in ra màn hình (dòng mới) tổng số cuộc gọi có trong log
- **?count_time_calls_from <phone_number>**: in ra màn hình (dòng mới) toognr thời gian gọi (tính theo second) xuất phát từ SDT <phone_number>

Chú ý:

- Số lượng truy vấn cũng có thể lên tới 100000 dòng
- Các tham số ngăn cách với nhau bởi 1 dấu cách trống

Ví dụ

stdin	**stdout**

call 0912345678 0132465789 2022-07-12 10:30:23 10:32:00 call 0912345678 0945324545 2022-07-13 11:30:10 11:35:11 call 0132465789 0945324545 2022-07-13 11:30:23 11:32:23 call 0945324545 0912345678 2022-07-13 07:30:23 07:48:30 # ?check_phone_number ?number_calls_from 0912345678 ?number_total_calls ?count_time_calls_from 0912345678 ?count_time_calls_from 0132465789 #	1 2 4 398 120

=====

Gợi ý chung

- Xác định đâu là phần dữ liệu log và đâu là phần truy vấn nhờ ký hiệu #, hoặc nhờ từ khóa call và dấu ? ở trước, hoặc nhờ ký tự c và ? hoặc # ở đầu dòng
- Đọc các phần của log và tách ra các trường? Các tham số này đều tuân theo format chuẩn tuy nhiên nếu đọc vào thì sẽ bị lỗi do có ký tự - và :
  cin>>sdt1>>sdt2>>y1>>m1>>d1>>h1>>m1>>s1>>h2>>m2>>s2
    - Vậy có thể dùng dạng scanf("%s%s%d-%2d-%2d%2d:%2d:%2d%2d:%2d:%2d",…)

```
char sdt1[15], sdt2[15];
int y1,mm1,d1,h1,m1,s1,h2,m2,s2;
scanf("%s %s %d-%d-%d %d:%d:%d %d:%d:%d",sdt1,sdt2,&y1,&mm1,&d1,&h1,&m1,&s1,&h2,&m2,&s2);
```

    - tuy nhiên dễ nhất là đọc thành xâu ý tự rồi xử lý tiếp (nhưng bạn sẽ cần thêm việc chuyển xâu thành giá trị số)

- Kiểm tra SDT đúng định dạng hay sai định dạng? Độ dài 10 là đủ?
- Tính thời gian thoại: Chỉ là thời gian trong cùng 1 ngày

====

Chuẩn bị trước tại nhà

Đọc vào dữ liệu log cuộc gọi thế nào?

Kiểm tra SDT đúng hay sai thế nào?
Tính thời gian của 1 cuộc gọi thế nào? VD từ h1:m1:s1 tới h2:m2:s2

Bài 2. Tìm đường ngắn nhất thoát khỏi mê cung

Một mê cung hình chữ nhật được biểu diễn bởi 0-1 ma trận NxM trong đó

- A[i,j] = 1 thể hiện ô (i,j) là tường gạch
- và A[i,j] = 0 thể hiện ô (i,j) là ô trống, có thể di chuyển vào.

Từ 1 ô trống, ta có thể di chuyển sang 1 trong 4 ô lân cận (lên trên, xuống dưới, sang trái, sang phải) nếu ô đó là ô trống. Để thoát khỏi mê cung chỉ cần đi ra được tới biên là xong. Xuất phát từ 1 ô trống trong mê cung, hãy tìm đường ngắn nhất thoát ra khỏi mê cung.

- Input
  - Dòng 1: ghi 4 số nguyên dương n, m, r, c trong đó n và m tương ứng là số hàng và cột của ma trận A (1 <= n,m <= 999) và r, c tương ứng là chỉ số hàng, cột của ô xuất phát.
  - Dòng i+1 (i=1,...,n): ghi dòng thứ i của ma trận A
- Output
  - Ghi số bước cần di chuyển ngắn nhất để thoát ra khỏi mê cung, hoặc ghi giá trị -1 nếu không tìm thấy đường đi nào thoát ra khỏi mê cung.

stdin	**stdout**
8 12 5 6 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1	7

1 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 1 1 1 0 1 0 1	

Gợi ý:

- Tìm đường đi trong mê cung: Thuật toán backtracking, nhưng thuật toán này không đảm bảo việc tìm được đường đi ngắn nhất
- Nếu đã có hình dạng mê cung, dùng thuật toán lan – tìm theo chiều rộng – BFS, lan từ vị trí hiện tại ra xung quanh theo 4 hướng cho tới khi gặp biên nào đầu tiên là được

Chuẩn bị trước tại nhà

Tại sao thuật toán lan – BFS lại đảm bảo tìm được đường ngắn nhất?
Làm thế nào tại vị trí M[i][j] bạn biết được mình đã thoát khỏi mê cung hay chưa
Dùng cấu trúc gì để hỗ trợ quá trình duyệt theo chiều rộng?

Bài 3. Giá trị nhỏ nhất trong khoảng

Cho 1 đoạn gồm n số nguyên với giá trị $a_0,. . ., a_{n-1}$, ta định nghĩa rmq(i, j) là giá trị nhỏ nhất trong đoạn từ a_i tới a_j (giá trị số nhỏ nhất trong các số $a_i, a_{i+1}, . . ., a_j$).
Ví dụ dãy 10 phần tử 1,5,3,7,8,43,23,5,12,7 thì

- rmq(0,9) = 1
- rmq(1,9)=3

- rmq(3,5)=7

Với đầu vào là m đoạn $(i_1, j_1),. . ., (i_m, j_m)$, giá trị tổng của các rmq định nghĩa trên m cặp được tính như sau $Q = rmq(i_1, j_1) + . . . + rmq(i_m, j_m)$

Input

- Dòng 1: là số nguyên n (1 <= n <= 106)
- Dòng 2: chứa giá trị các phần tử trong đoạn ban đầu a0, . . . , an-1 (1 <= ai <= 106)
- Dòng 3: là giá trị m (1 <= m <= 106)
- Các dòng tiếp theo từ k+3 (k = 1, . . ., m): là các cặp giá trị i_k, j_k ($0 <= i_k < j_k < n$)

Output: in ra giá trị Q

stdin	**stdout**
16 2 4 6 1 6 8 7 3 3 5 8 9 1 2 6 4 4 1 5 0 9 1 15 6 10	6

Gợi ý:

- Tìm giá trị nhỏ nhất trong đoan – rmq bằng cách dùng vòng lặp tìm giá trị min? Vậy với mỗi 1 đoạn con ta đều sẽ phải lặp lại
- Có cách gì tính được rmq nhanh hơn? VD. Nếu biết trước $rmq(a_i, a_j)$ thì $rmq(a_{i+1},a_j)$ hoặc $rmq(a_i,a_{j+1})$ sẽ được tính như thế nào?

Chuẩn bị tại nhà

Nếu tìm rmq của các đoạn con dùng cách tìm min bằng vòng lặp thì tính Q sẽ có chi phí thời gian trong trường hợp tồi nhất là bao nhiêu theo O-lớn?
$rmq(a_{i+1},a_{j+1})$ được tính từ $rmq(a_i, a_j)$ theo công thức nào?

Bài 4. Tìm hình chữ nhật tạo nên bởi các bit 1 có diện tích lớn nhất

Một hình chữ nhật kích thước n x m được cấu tạo từ các ô con điền giá trị là 0 hoặc là 1. Hãy tìm hình chữ nhật được tạo nên bằng các bit 1 có diện tích lớn nhất.

Đầu vào

- Dòng 1: chứa số nguyên dương n và m (1 <= n, m <= 1000)
- Dòng i+1 (i = 1,…, n): chứa hàng thứ i của ma trận A

Kết quả

- In ra diện tích của hình chữ nhật lớn nhất tìm được

stdin	stdout
4 4 0 1 1 1 1 1 1 0 1 1 0 0 1 1 1 0	6

Gợi ý:

- Biểu diễn hình chữ nhật ta sẽ cần 1 tọa độ là: điểm trên cùng bên trái và dưới cùng bên phải
- Hình chữ nhật thỏa mãn nếu nó chỉ chứa các bit 1

Câu hỏi chuẩn bị tại nhà

1 vị trí chứa bit 1 đã xét qua rồi thì liệu có cần xét lại nó trong lần xem xét HCN tiếp theo không?
Cận dưới thời gian thực hiện của thuật toán này trong trường hợp tồi nhất theo O-lớn sẽ là?
Khi đang tìm được HCN nhỉ, việc mở rộng để xét xem tiếp thế nào VD. 0 1 1 1 1 1 1 0 1 1 0 0

1 1 1 0

Bạn đang tìm được HCN có diện tích là 2, vậy bạn mở rộng HCN trên như thế nào?