Dhoom 4

Link submit: https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/practice-problems/algorithm/dhoom-4/description/

Solution:

C++	https://ideone.com/79W9eg
Java	https://ideone.com/ZV3HAX
Python	https://ideone.com/yYTD4U

Tóm tắt đề:

Bạn có một chiếc chìa khóa, trên đó có in một số nguyên và một giá trị khóa cần tìm. Bạn cũng có N chìa khóa khác, mỗi chìa có một giá trị. Bạn cần thực hiện nhân số trên chìa khóa bạn đang có, lần lượt với các số trong N số theo một thứ tự nào đó để có kết quả bằng với giá trị khóa cần tìm. Một số trong N số có thể được nhân nhiều lần với số của bạn. Cứ mỗi lần nhân, bạn phải lấy kết quả mod cho 100000.

Ví dụ: số trên chìa khóa của bạn là X và bạn lấy chìa khóa có giá trị Y thì chìa khóa mới bạn có được là (X*Y)%100000.

Mỗi lần nhân hai số với nhau chỉ tốn một giây. Hãy xác định thời gian tối thiểu để đạt được giá trị khóa cần tìm.

Input:

Dòng đầu tiên chứa hai số nguyên là số in trên chìa khóa của bạn và giá trị cần tìm.

Dòng tiếp theo chứa số N là số lượng chìa khóa bạn có.

Dòng cuối cùng gồm N số cách nhau bởi dấu khoảng cách cho biết con số trên từng chìa khóa.

Output:

Thời gian tối thiểu bạn cần để bạn đạt được giá trị cần tìm. Nếu không thể đạt được in -1.

Ví dụ:

3 30 3 2 5 7	2

Giải thích ví dụ:

Số đầu tiên của bạn xuất phát là 3, bạn sẽ thực hiện nhân hai lần như sau:

Lần 1: 3 x 2 = 6

Lần 2: 6 x 5 = 30

Hướng dẫn giải:

Nếu bạn xem những giá trị mà bạn có thể có được sau một số quá trình nhân nhất định là một đỉnh của đồ thị, thì bạn có thể sử dụng được giải thuật BFS cho bài toán này, cụ thể như sau:

Bạn xem như đỉnh xuất phát là giá trị ban đầu bạn có và đỉnh đích là giá trị bạn cần tìm. Hai đỉnh x và y có cạnh nối với nhau (cạnh một chiều từ x đến y) nếu tồn tại một chỉ số i ($1 \le i \le N$) sao cho : (x * a[i]) % 100000 = y. Khi đã có ý tưởng duyệt đồ thị như thế, bạn sẽ tiến hành cài đặt giải thuật BFS như sau:

Chuẩn bị:

dist[i]: số bước ít nhất đi từ đỉnh xuất phát đến đỉnh i. Nếu không thể tới được i thì dist[i] = -1.

queue <int> q: một hàng đợi để lưu các đỉnh sẽ được xét.

Áp dụng:

Bạn đưa đỉnh xuất phát đầu tiên (start) vào queue, đỉnh xuất phát ở đây mặc nhiên bạn hiểu rằng đó là giá trị khởi tạo ban đầu của khóa. Gán dist[start] = 0.

Mỗi lần bạn lấy ra khỏi queue một đỉnh u, bạn duyệt qua toàn bộ phần tử trong mảng chứa giá trị của chìa khóa và tiến hành cập nhật giá trị v = (u * a[i]) % 100000, nếu dist[v] = -1 thì bạn tiến hành gán dist[v] = dist[u] + 1 và đưa v vào queue.

Kết quả cần tìm là dist[target] với target là cái giá trị cần đạt tới. Ở đây vì nếu không tồn tại thì dist = -1 nên kết quả là dist[target] đúng với mọi trường hợp.

Độ phức tạp:

Time Complexity: vì các đỉnh trên đồ thị, khi đã thăm rồi thì không được thăm lại nữa, do đó ta lấy tối đa chỉ có 100.000 đỉnh ra khỏi queue. Mặt khác, với từng đỉnh ta lấy ra khỏi hàng đợi, ta cần duyệt thêm N khóa nữa để có thể sinh ra các khóa mới. Như vậy, độ phức tạp thuật toán theo lý thuyết là **O(100.000 * N)**.

Space Complexity: **O(100.000)**.