Dhoom 4

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

Link solution:

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

Tóm tắt đề: Bạn có chìa khóa mang một con số và một giá trị khóa cần tìm. Cho bạn danh sách N số khác, hỏi bạn có cách nào để nhân số 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.

Nếu tìm được số kết quả bạn in ra số lượng số mà bạn phải nhân với số của bạn để có được kết quả cần tìm, nếu không tìm được bạn in ra -1.

Input

Dòng đầu tiên chứa 2 số là số chìa khóa của bạn và một giá trị cần tìm.

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

Dòng cuối cùng chứa N số cách nhau bởi dấu khoảng cách.

Output

Một số nguyên duy nhất là số lần ít nhất bạn cần có để kết quả bạn đạt được giá trị cần tìm.

Example

Input

3 30 3 2 5 7	2

Giải thích: Số đầu tiên của bạn xuất phát là 3, bạn sẽ sử dụng 2 lần nhân 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 của bạn có và đỉnh đích là giá trị bạn cần tìm. 2 đỉnh x và y có cạnh nối với nhau (cạnh một chiều từ x đến y) nếu như 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 để từ đỉnh xuất phát đến đỉnh i. Nếu như 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 toàn bộ mọi giá trị a[i] có trong mảng và tiến hành cập nhật một 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 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.

Đánh giá độ phức tạp thuật toán:

- 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).
- Độ phức tạp không gian: **O(100.000)**.