BÀI GIẢNG VỀ KỸ THUẬT LẬP TRÌNH VỚI C++

04 CẤU TRÚC DỮ LIỆU HÀNG ĐỢI – (QUEUE) VÀ CHIẾN THUẬT TÌM KIẾM THEO CHIỀU RỘNG

A. Vấn đề và Chiến thuật

Vấn đề:

Tìm kiếm theo chiều rộng (Breadth - First - Search) thường dùng cho việc:

- Xuất phát từ một kết quả ban đầu (*trạng thái ban đầu*).
- Có một tập hữu hạn các phép biến đổi $\{F_1, F_2, .., F_k\}$, mỗi phép biến đổi cho phép chuyển từ kết quả này sang kết quả khác (*trạng thái này sang trạng thái khác*).
- Cần biến đổi đến một kết quả đích (*trạng thái đích*) với số lần thực hiện biến đổi là **ít nhất**.

Chiến thuật:

Bài toán được giải bằng cách tìm kiếm lời giải theo chiến thuật:

☞ **B_0**: Nếu trạng thái đầu = trạng thái đích → Không cần biến đổi.

☞ **B1**: Từ trạng thái ban đầu, sử dụng tất cả các phép biến đổi $F_1, F_2, .., F_k$, ta sẽ nhận được các trạng thái ***Th1_1***, ***Th1_2***, .. , ***Th1_k***;

Nếu trong tập các trạng thái ***Th1_1***, ***Th1_2***, .. , ***Th1_k*** có chứa trạng thái đích thì lời giải là sau **1 lần biến đổi**.

Ngược lại, tức là các trạng thái ***Th1_1***, ***Th1_2***, .. , ***Th1_k*** **không** chứa trạng thái đích, thì ta kết luận được rằng, sau 1 bước biến đổi sẽ không nhận được trạng thái đích.

☞**B2:** Từ mỗi trạng thái thuộc tập ***Th1_1, Th1_2, .., Th1_k*** sử dụng tất cả các phép biến đổi $F_1, F_2, .. , F_k$ ta nhận được tập **trạng thái mới *Th2_1, Th2_2_, .. Th2_n***.

Nếu trong tập trạng thái này chứa trạng thái đích, ta kết luận được sau ít nhất 2 lần biến đổi , ngược lại sau 2 lần không thể nhận được trạng thái đích.

Tiếp tục biến đổi cho đến khi:

☞ **Tập trạng thái nhận được chứa trạng thái đích.**

Khi đó ta có lời giải.

Hoặc

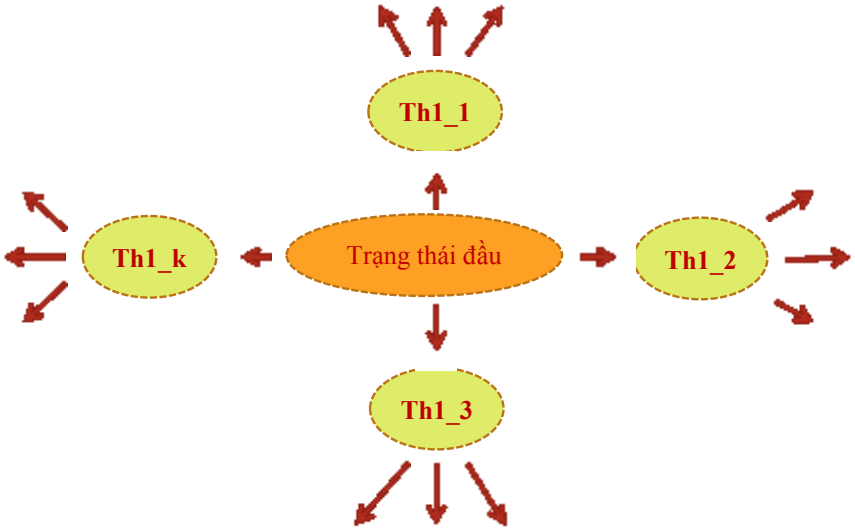
☞ **Không tạo ra được tập trạng thái mới.**

Khi đó ta kết luận, không thể biến đổi về trạng thái đích.

Chú ý:

☞ **Trong quá trình biến đổi, cần phân biệt được trạng thái nào là mới, trạng thái nào là cũ:**

☞ **Ta có thể sử dụng mảng đánh dấu theo chỉ số hoặc cấu trúc dữ liệu map hoặc dùng hash.**



B. Cài đặt

Ta sử dụng cấu trúc dữ liệu hàng đợi queue.

B0: Bỏ trạng thái ban đầu vào hàng đợi

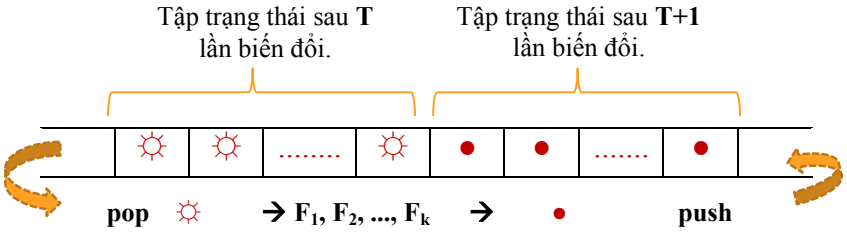
B1: Lặp:

☞ Lấy ra một trạng thái:

Sử dụng các phép biến đổi $F_1, F_2, .. , F_k$ để tạo ra các trạng thái:

- Nếu là trạng thái đích → **Lấy kết quả**.
- Nếu là **trạng thái mới** thì đưa trạng thái này vào hàng đợi.

☞ Quá trình lặp cho đến khi gặp trạng thái đích hoặc hàng đợi rỗng, khi hàng đợi rỗng mà vẫn chưa gặp trạng thái đích thì → Không thể biến đổi.



1☼. Biến đổi số

Cho tập hợp S gồm một số nguyên dương a, tức là: $S = \{a\}$ và một số nguyên dương M. Có hai phép biến đổi tác động lên các phần tử của S.

Chọn x là một số bất kì thuộc tập hợp S:

- Phép biến đổi thứ nhất: $x \rightarrow y = 2x + 1$, đưa y vào S.
- Phép biến đổi thứ hai: $x \rightarrow y = 3x - 1$, đưa y vào S.

Yêu cầu: Hãy thực hiện biến đổi ít lần nhất để nhận được số M.

Dữ liệu cho trong file ReformNumber.Inp gồm ghi số nguyên dương a và M ($a, M \leq 10^7$).

Kết quả ghi ra file ReformNumber.Out là số lần biến đổi ít nhất cần thực hiệc để nhận được số nguyên dương M. Nếu không thể biến đổi được thì ghi -1.

Ví dụ:

ReformNumber.Inp	ReformNumber.Out
2 14	2

2☼. Xóa chữ số

Cho số nguyên dương M. Bạn được biến đổi số bằng cách.

- Xóa 1 chữ số bất kì của M.
- Xóa hai chữ số liên tiếp của M nếu hai chữ số đó bằng nhau.

Sau khi xóa, các chữ số còn lại sẽ dồn lại (vẫn giữ nguyên vị trí) để được số mới.

Với số mới nhận được, ta liếp tục biến đổi theo quy tắc trên.

Yêu cầu: Hãy tìm các biến đổi với số lần thực hiện là ít nhất để được một số nguyên tố.

Dữ liệu cho trong file DelPrime.Inp gồm một số nguyên dương M ($M \leq 10^9$).

Kết quả ghi ra file DelPrime.Out là số lần biến đổi ít nhất để được số nguyên tố. Nếu không thể biến đổi để nhận được số nguyên tố thì ghi -1.

Ví dụ:

DelPrime.Inp	DelPrime.Out
11112	2
14	-1
13	0

3☼. Cộng hai số

Cho tập S gồm hai số nguyên dương a và b , $S = \{a, b\}$, (a và b có thể bằng nhau) và số nguyên dương M.

Ta biến đổi trên tập S như sau:

Lấy hai phần tử bất kỳ u, v thuộc S, tính tổng $x = u + v$; thêm x vào S.

Yêu cầu: Hãy thực hiện ít lần biến đổi nhất để tập S có chứa số M.

Dữ liệu cho trong file NumPlusNum.Inp gồm:

- Dòng đầu ghi số nguyên dương T ($T \leq 3$) là số bộ dữ liệu.
- T dòng tiếp theo, mỗi dòng ghi 3 số nguyên dương a, b, M.

Kết quả ghi trong file NumPlusNum.Out gồm T dòng. Mỗi dòng là số lần biến đổi ít nhất để tập S chứa số nguyên dương M. Nếu không thể biến đổi được thì ghi -1.

Ví dụ:

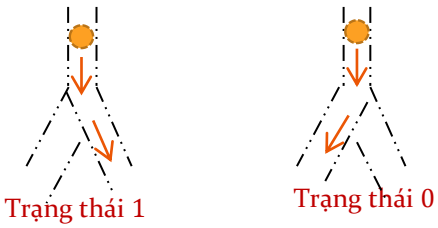
NumPlusNum.Inp	NumPlusNum.Out
1 2 3 7	2
1 2 3 2	0
1 2 3 1	-1

Giới hạn:

- Sub1: $a, b, M \leq 3 \times 10^3$;
- Sub2: $a, b, M \leq 10^4$.

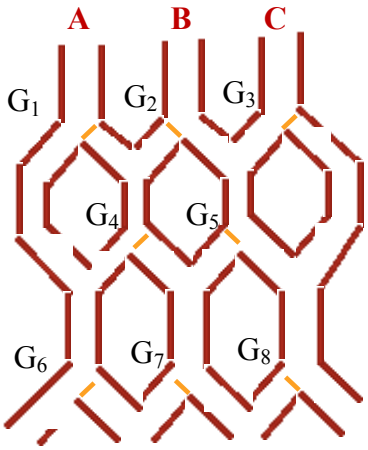
4☼. Otomat

Có một Otomat được ghép từ các chi tiết có một trong hai trạng thái 0 hoặc 1 như hình 1. Otomat có cấu trúc như hình 2 gồm tám chi tiết $G1, G2, \ldots, G8$ với ba lối vào A, B, C. Trạng thái của otomat được thể hiện bởi một xâu nhị phân độ dài 8 là các trạng thái tương ứng của G_1, G_2, … , G8.



Hình 1: Bi sẽ di chuyển theo hướng mũi tên.

Otomat hoạt động như sau: Khi thả một quả cầu vào một lối nào đó, sau khi quả cầu đi qua một chi tiết nào đó, chi tiết đó thay đổi trạng thái từ 0 thành 1 hoặc từ 1 thành 0. Hoạt động của Otomat được thể hiện bởi một xâu kí tự S chỉ gồm các chữ cái hoa A, B, C mà mỗi kí tự trong xâu S thể hiện việc ta thả quả cầu vào lối vào với tên kí tự đó. Ví dụ S = AABC có nghĩa là ta lần lượt thả quả cầu vào các lối A, A, B, C.



Hình 2.

Bài toán đặt ra như sau: Cho hai trạng thái bất kì T1, T2 của Otomat. Hãy tìm một xâu kí tự S ngắn nhất có thể được thể hiện hoạt động của otomat chuyển từ trạng thái T1 sang trạng thái T2.

Dữ liệu cho trong tệp văn bản Otomat.Inp như sau:

- Dòng đầu ghi xâu nhị phân độ dài 8 mô tả trạng thái T1
- Dòng thứ hai ghi xâu nhị phân độ dài 8 mô tả trạng thái T2.

Kết quả ghi ra tệp văn bản Otomat.Out xâu S ngắn nhất mô tả cách chuyển trạng thái T1 sang T2. Nếu không có thì ghi "No solution", nếu không cần biến đổi thì ghi "khong can bien doi". Nếu có nhiều xâu biến đổi thì ghi xâu có thứ tự từ điển nhỏ nhất.

Ví dụ:

Otomat.Inp	Otomat.Out
11111111 10100011	AABCCCC
11100011 10000010	No solution

5☼. Táo thối

Siêu thị Apples supermarket có một gian hàng trưng bày táo rất lớn. Khu hàng được chia làm m dòng và n cột. Ô ở dòng i và cột j được gọi là ô (i, j). Tại mỗi ô (i, j) có thể đặt một quả táo hoặc không có quả táo nào. Do số lượng táo nhiều và bán chưa hết nên có những quả táo bị hỏng, có những quả táo chưa hỏng. Cứ sau một ngày, nếu một quả táo chưa bị hỏng thì sẽ bị hỏng nếu các ô bên cạnh có một quả táo đã bị hỏng (có nhiều nhất 4 ô bên cạnh). Cho biết hiện trạng các quả táo bị hỏng và chưa bị hỏng, hỏi bao nhiêu ngày nữa thì tất quả táo có trong siêu thị bị hỏng.

Dữ liệu cho trong file ROTA.INP như sau:

- Dòng đâu ghi hai số nguyên dương m và n.
- m dòng tiếp theo, mỗi dòng ghi n số thuộc {0, 1, 2} mô tả trạng thái của n quả táo trên dòng đó: số 0 mô tả ô không có táo, 1 mô tả ô chứa quả táo chưa bị hỏng, 2 mô tả ô chứa quả táo bị hỏng.

Kết quả ghi ra file ROTA.OUT là số ngày ít nhất để tất cả các quả táo trong siêu thị đều bị hỏng. Nếu không có thì ghi ra -1.

Ví dụ:

ROTA.INP	ROTA.OUT
3 5 2 1 0 2 1 1 0 1 2 1 1 0 0 2 1	2
3 5 2 1 0 2 1 0 0 1 2 1 1 0 0 2 1	-1

Giới hạn: $n, m \leq 1000$.

6☼. Địa đạo

Trong chiến tranh, người ta xây dựng rất nhiều địa đạo. Địa đạo XYZ là một địa đạo khổng lồ, được cấu thành từ các đường hầm tạo thành một lưới ô vuông gồm 1002 đường hầm dọc, 1002 đường hầm ngang. Các đường hầm dọc được đánh số từ 0 đến 1001 theo hướng từ trái sang phải, các đường hầm ngang được đánh số 0 đến 1001 theo hướng từ trên xuống dưới. Ở tại mỗi điểm giao của đường hầm dọc và đường hầm ngang đều có một cánh cửa. Cánh cửa giao giữa đường hầm ngang i và đường hầm dọc j thì ta nói cặp (i, j) là tọa độ của cánh cửa. Những cánh cửa này giúp địa đạo được an toàn hơn, hiện địa đạo có k cánh cửa đang đóng, các cánh của còn lại đều mở. Ban chỉ huy đang đứng ở cửa có tọa độ (x, y), cửa này đang mở và muốn di chuyển tới cửa có tọa độ (0, 0) theo các đường hầm. Hãy tính xem, Ban chỉ huy cần mở ít nhất bao nhiêu cánh cửa đển có thể đi tới được cánh cửa có tọa độ (0; 0), cửa tọa độ (0; 0) luôn được mở.

Dữ liệu cho trong file DIADAO.INP như sau:

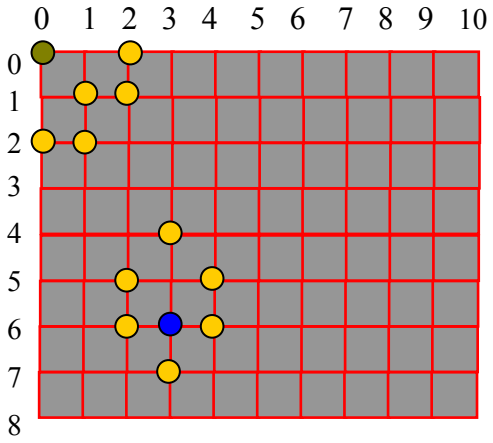
- Dòng 1: Gồm ba số nguyên dương cách nhau là k, x, y là số lượng cánh cửa đang đóng và tọa độ của Ban chỉ huy.
- Dòng 2 đến dòng k+1: Mỗi dòng chứa hai số (x, y) là tọa độ của cánh cửa đang đóng.

Kết quả ghi ra file DIADAO.OUT là số cửa ít nhất cần mở.

Ví dụ:

DIADAO.INP	DIADAO.OUT
11 6 3 6 2 5 2 4 3 2 1 7 3 5 4 6 4 2 0 1 1 1 2 0 2	2

Giới hạn: $1 \le k \le 50.000$



7☼. Trò chơi di chuyển trên bảng

Trên một bảng kí tự gồm *m* x *n* , các hàng được đánh số từ 1 đến m từ trên xuống dưới, các cột đánh số thứ tự từ 1 đến *n* từ trái qua phải. Ô nằm trên hàng *i*, cột *j* được gọi là ô (*i, j*). Tại mỗi ô (*i, j*) có ghi một kí tự thuộc tập {'A', …, 'Z', '*'}. Một người chơi trò chơi di chuyển trên bảng như sau:

- Tại một ô (*i, j*), người chơi có thể di chuyển sang một trong 4 ô kề cạnh (tất nhiên không di chuyển ra ngoài). Chi phí cho một lần di chuyển như vậy được tính: Nếu di chuyển sang ô kề cạnh có cùng kí tự với ô hiện tại thì mất chi phí là 0, người lại mất chi phí là 1.

Yêu cầu: Từ một ô (*x, y*). Tìm cách di chuyển sao cho chi phí là ít nhất để đến được ô chứa kí tự '*'.

Dữ liệu cho trong file MOVE1.INP như sau:

- Dòng đầu ghi 3 số nguyên dương *m, n, Q* tương ứng là số dòng, số cột số câu hỏi cần tính ($1 < m, n \le 2000$; $1 \le Q \le m*n$);
- m dòng tiếp theo, mỗi dòng ghi một xâu kí tự gồm *n* kí tự thuộc {'A', … , 'Z', '*'}.
- Q dòng tiếp theo, mỗi dòng ghi cặp chỉ số *x, y*.

(dữ liệu luôn đảm bảo ít nhất có một kí tự '*')

Kết quả ghi ra file MOVE1.OUT gồm Q dòng, mỗi dòng là chi phí ít nhất cần di chuyển của ô (x, y) tương ứng.

Ví dụ:

MOVE1.INP	MOVE1.OUT
3 4 2 AAAB *BBC AABB 1 1 3 4	1 1