TỔNG HỢP BÀI DAY 3

Bài 12: Đồ thị ý tưởng - IDEAGRAPH (3.5s, 10G)

Bây giờ, khởi đầu Day 3, ta hãy cùng nói một chút về người ra đề bài này. Bạn có phải đang hỏi: Làm sao người ra đề này có thể nghĩ ra 15 bài độc lập mà không trùng lặp ý tưởng như vậy?

**- Thì các bạn tự đi mà giải thích chứ nhờ ai!!!**

Đùa thôi, Trôn Việt Nam! Cho T đồ thị được biểu diễn dưới dạng lưới M x N, mỗi đồ thị gồm một gốc và các đích ý tưởng. Mỗi ô có một chi phí riêng. Với chi phí tối đa có thể cho mỗi đồ thị là K, liệu bạn có thể đến các ô đích ý tưởng, quay trở lại ô gốc ý tưởng, và tạo ra thêm một bài không?

Các loại ô trong đồ thị là:

. là ô trống (không đi tới được)

G0 là ô gốc ý tưởng.

*x* là một ô trong đồ thị mà bạn phải tốn chi phí x nếu muốn đến ô đó.

| hay – là cạnh nối (có những cạnh nối kiểu |- là chéo-xuống-phải hoặc chéo-lên-phải hay -| là chéo-xuống-trái hoặc chéo-lên-trái)

P*Q* là ô nhảy. Khi đến ô này, bạn có thể (không bắt buộc) nhảy đến một ô có dạng P*Q* khác với chi phí là Q.

F*x* là ô đích ý tưởng với chi phí x.

Quy tắc di chuyển: Q lượt di chuyển đầu tiên được miễn thuế. Sau đó, A lượt di chuyển tiếp theo chịu thuế C%, tiếp tục A + B lượt di chuyển tiếp theo chịu thuế C + D%, A+2B lượt di chuyển tiếp theo chịu thuế C+2D%, cứ thế cộng dồn.

Input: Dòng đầu: số T, thể hiện số testcase

Mỗi testcase có cách thức như sau:

Dòng đầu: Tám biến M, N, K, Q, A, B, C, D

M dòng sau, mỗi dòng N kí tự như trên.

Output: Với mỗi testcase, in ra 1 dòng là các hướng di chuyển để tới các ô đích ý tưởng rồi quay lại ô ban đầu.

T dòng sau đó, mỗi dòng in ra NO – nếu không thực hiện được, hoặc YES *x* với x là số chi phí còn thừa. Dòng cuối, in ra số bài thực hiện được.

Ví dụ:

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 1  3 4 12 5 1 2 3 4  G0 – 4 -|  | . |- 7  F0 . . . | v ^  YES 12  1 |

Subtasks:

Subtask 1 (5%) : M, N < 100, T < 30, K < 500, Q, A, B, C, D < 100, không có ô nhảy, chỉ có 1 ô đích ý tưởng.

Subtask 2 (15%) : 100 < M, N < 1000000, 30 < T < 500, 500 < K < 100000, Q, A, B, C, D < , không có ô nhảy.

Subtask 3 (30%) ; 1000000 < M, N < , 500 < T < , 100000 < K < , Q, A, B, C, D <

Subtask 4 (50%) : < M, N < , < T < , < K < , giới hạn của Q, A, B, C, D giữ nguyên.

“Ta sẽ… backup vào ***tim của những người đọc đề*** thay vì USB”

Bài 13: Trận chiến sinh tử - FIGHTORDIE

Sau khi An và Anh Coder May Mắn đạt được mức rank “**Thách đấu người ra đề TBCP**”, người ra đề của TBCP đã thật sự gạ kèo đấu 1 vs 2 với họ, nhưng lần này, họ sẽ phải giải những bài toán khó nhất do chính tay Người Ra Đề biên soạn. Người Ra Đề còn đích thân tạo ra chế độ 1 vs 2 chỉ để đấu với An và Anh Coder May Mắn.

Quy tắc đấu 1 vs 2 kiểu “chỉ dành cho An và Anh Coder May Mắn” như sau: Người Ra Đề có 12 bài trong ngân hàng bài bí mật của TBCP. Sẽ có tổng cộng 3 vòng đấu, mỗi vòng là một sự đặc biệt với một bài được hệ thống AI của TBCP phối như sau:

Vòng 1: Bao giờ em mới hết nhảy người yêu hay em chỉ mãi mãi là một “*trap girl*” - SNAKENINJA (4.0s, 8G)

Phần hai của trò chơi TURTLE NINJA mang tên SNAKE NINJA gồm T màn chơi, mỗi màn chơi là một lưới M x N, mục tiêu là chú rắn (không dài ra) ở tọa độ (x, y) (0-indexed) sẽ phải ăn hết số táo sau đó quay lại ô ban đầu. Lưới gồm các loại ô như sau:

. là ô trống (có thể đi tới)

W là ô tường (xem lại bài 1)

P*q* là ô cổng dịch chuyển (xem lại bài 12)

<hướng di chuyển>*x* là ô di chuyển tự động (xem lại bài 8)

A*x* là các ô chứa táo với chi phí x (xem lại bài 12)

X là ô có chi phí x (xem lại bài 12)

S là một ô gai. Khi đến hay chỉ vô tình đi qua ô này, Ninja Rắn sẽ “ỏm củ tỏi” ngay lập tức. ***Game Over***.

Ninja Rắn sẽ có thể di chuyển như Ninja Rùa + 4 ô cạnh Ninja Rắn. (xem lại bài 8)

Quy tắc di chuyển: Q lượt di chuyển đầu tiên được miễn thuế. Sau đó, A lượt di chuyển tiếp theo chịu thuế C%, tiếp tục A + B lượt di chuyển tiếp theo chịu thuế C + D%, A+2B lượt di chuyển tiếp theo chịu thuế C+2D%, cứ thế cộng dồn.

Hãy tính giúp An và Anh Coder May Mắn:

1. Độ dài đường đi (tính theo ô) và số lần rẽ (tối thiểu)

2. Đường đi (xem lại bài 1)

3. Chi phí tối thiểu.

Input: Dòng đầu: số T, thể hiện số testcase

Mỗi testcase có cách thức như sau:

Dòng đầu: Chín biến M, N, X, Y, Q, A, B, C, D

M dòng sau, mỗi dòng N kí tự như trên.

Output: Với mỗi testcase, in ra như sau:

Ví dụ:

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 1  2 2 0 1 12 1 1 1 1  . .  A0 2 | 3 1  v < ^ >  2 |

Subtasks:

Subtask 1 (3%) : M, N < 100, T < 30, Q, A, B, C, D < 100, không có ô cổng dịch chuyển, không có ô gai, không có ô tường, số táo < 5

Subtask 2 (15%) : 100 < M, N < 1000000, 30 < T < 500, Q, A, B, C, D < , không có ô cổng dịch chuyển

Subtask 3 (30%) ; 1000000 < M, N < , 500 < T < , Q, A, B, C, D <

Subtask 4 (52%) : < M, N < , < T < , giới hạn của Q, A, B, C, D giữ nguyên.

Vòng 2: Hôm đó em yêu anh đến mức nào mà anh phải chia xâu nhị phân? – CUTBINARYSTRING (3.5s, 8G)

Cho T xâu nhị phân S với độ dài là L cho trước. Với mỗi xâu S, hãy chia nó thành những đoạn con thỏa mãn các yêu cầu sau đây sao cho tổng độ dài tất cả các đoạn con thỏa mãn yêu cầu là lớn nhất. Nếu có nhiều cách chia, chọn cách chia có chi phí ít nhất.

- Phải chọn ít nhất (max (m, n, y)) đoạn con trở lên.

- Độ dài đoạn con phải là số nguyên tố > P.

- Số lượng đoạn con *ch* trong đoạn con đó cũng phải là số nguyên tố.

- Hai kí tự ở vị trí x của đoạn con M và N phải giống nhau.

- Đoạn con thứ y phải có ít nhất A đoạn con đối xứng có độ dài lớn hơn 1 trong nó.

Chi phí để chia một đoạn con lần đầu là K, lần tiếp theo là K + C, lần sau nữa là K + 2C, cứ thế cộng dồn. Mục tiêu của An là chia sao cho tổng độ dài tất cả các đoạn con là lớn nhất và chi phí nhỏ nhất.

Input: Dòng đầu: số T – thể hiện số testcase

Mỗi testcase có cách thức như sau:

Dòng đầu: Mười biến L, *ch*, K, C, M, N, X, Y, A, P trên cùng một dòng

Dòng sau: xâu S

Output: Với mỗi bộ test, in ra như sau:

Dòng đầu: Tổng độ dài các đoạn con thỏa mãn yêu cầu, số lần chia và chi phí tối thiểu

Dòng sau: Vị trí của các lần chia (1-indexed)

Ví dụ:

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 1  10 1 1 1 1 2 1 2 1 1  1010111010 | 10 1 1  5 |

Subtasks:

Subtask 1 (10%) : T < 50, L < 1000

Subtask 2 (15%) : 50 , T < , 1000 < L <

Subtask 3 (20%) : < T < , < L <

Subtask 4 (55%) : < T < , < L <

Vòng 3: Kích hoạt Laser – ACTIVATELASER (5.0s, 8G)

Bây giờ, Người Ra Đề đã giấu đề bài vào một mê cung rất khó giải mã. An và Anh Coder May Mắn phải dùng một chiếc “máy laser” giả lập để có thể lấy được đề bài từ Người Ra Đề. Mê cung của Người Ra Đề chính là T lưới M x N, mỗi lưới giấu một mảnh đề bài. Các ô trong Mê Cung như sau: (Lưu ý mỗi ô gương sẽ có loại – , - hoặc +, nó đứng trước chức năng đặc biệt của gương đó.)

. là ô trống

| , - hay + là ô gương (| chỉ phản chiếu khi chạm vào theo hướng trái hoặc phải, - chỉ phản chiếu khi nó được tiếp xúc ở mặt trên hoặc dưới, + phản chiếu ở cả 4 mặt trên – dưới – trái – phải)

S là ô hấp thụ, ô này chặn đường (trap) laser.

F là ô cất giấu mảnh đề bài

P*x* là ô dịch chuyển, khi đến ô này có thể dịch chuyển tia đến ô cũng có dạng P*x*, giữ nguyên hướng. (x là ID của cổng, chỉ có thể dịch chuyển đến ô có cùng ID)

$*x* là ô gương yêu cầu, nó sẽ chỉ phản xạ khi số lượt bắn chính xác là x

L*x* là ô gương yêu cầu, nó sẽ chỉ phản xạ khi số lượt bắn x

U*x* là ô gương yêu cầu, nó sẽ chỉ phản xạ khi số lượt bắn x

E*x* là ô gương yêu cầu, nó sẽ chỉ phản xạ khi số lượt bắn là chẵn

O*x* là ô gương yêu cầu, nó sẽ chỉ phản xạ khi số lượt bắn là lẻ

*x* là ô gương với số lần sử dụng là x

=*x* là Màng Nhẹ Nhàng cấp x, tia sáng sau khi đi xuyên qua sẽ có thể phản chiếu bằng ô gương có giới số lần sử dụng mà không bị giảm 1 lần sử dụng nhưng cũng không thể lấy mảnh đề bài. Hiệu ứng có tác dụng trong x lần phản xạ tiếp theo.

@ là ô biến, bạn có thể biến ô này thành một ô chức năng khác với số x bất kì ngoại trừ ô F và ô P*x*. Nếu không biến, ô này là ô trống.

Bạn có thể chọn 1 trong các ô trước hàng dưới cùng để bắn laser với 1 trong 2 góc: < (góc trái 45 độ) hoặc > (góc phải 45 độ)

Hãy giúp An và Anh Coder May Mắn tìm ra đề bài cuối cùng để đánh bại Người Ra Đề nhé!

Input: Dòng đầu: số T – số mê cung.

Mỗi testcase có cách thức như sau:

Dòng đầu: Hai số M, N

M dòng sau, mỗi dòng N kí tự như trên

Output: Với mỗi mê cung, in ra -1 nếu không thể tiếp cận ô chứa mảnh đề thi, nếu có thể tiếp cận, in ra vị trí đặt máy laser (1-indexed), hướng của máy laser và việc biến đổi các ô (nếu có ô biến đổi) theo kiểu: <số thứ tự ô biến đổi (1-indexed, các ô biến đổi được đánh số theo thứ tự từ trên xuống dưới, trái qua phải)><chức năng biến đổi> trên cùng một dòng. Ví dụ 3+U4 chỉ việc gán ô biến đổi thứ 3 thành ô +U4.

Ví dụ:

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 1  3 3  F . .  S . .  . . . | -1 |

Subtasks:

Subtask 1 (10%) : M, N < 50, T < 10 chỉ có loại gương – và |, chỉ có các loại ô trống, gương, đích đến, hấp thụ.

Subtask 2 (15%) : 50 < M, N < 1000, 10 < T < 100, có thêm loại ô gương yêu cầu.

Subtask 3 (15%) : 1000 < M, N < , 100 < T < 10000, có thêm ô gương với giới hạn số lần sử dụng.

Subtask 4 (20%) : < M, N < , 10000 < T < , có thêm ô dịch chuyển.

Subtask 5 (20%) : < M, N < , < T < , có thêm ô Màng Nhẹ Nhàng.

Subtask 6 (20%) : < M, N < , < T < , có thêm tất cả các ô còn lại.

Bài này là một bài đặc biệt, nó sẽ gộp input 3 vòng vào mà bạn vẫn phải in ra theo đúng yêu cầu.

“Khi họ đọc đề bài cuối cùng, họ sững sờ. Không có một chữ nào cả. Lúc này, Anh Coder May Mắn mới nói với An: Thật ra, tớ chính là cậu… của 11 năm sau. Tớ đến để giúp cậu thoát khỏi số phận hẩm hiu, khi bị “*hỏng tim*” do code quá nhiều, và cũng để giúp cậu chiến thắng trận chiến 1 vs 2 với người ra đề, nếu không có tớ giúp cậu, lúc đó, cậu sẽ mất rank: “***Thách Đấu Người Ra Đề TBCP.*** ” và cậu sẽ trầm kẽm mà buồn đau, nhưng thôi, tôi lại phải trở về thế giới coder của tôi, hết giờ giúp đỡ cậu rồi.”