ROUND 3G - Các cấu trúc hậu tố

Tại một cuộc thi "Bizons" có vấn đề sau đây: ". Bạn đang đưa ra hai từ khác nhau (chuỗi chữ cái tiếng Anh), s và t. Bạn cần phải chuyển đổi từ s thành chữ t". Các nhiệm vụ trông đơn giản với mọi người vì họ biết rõ các cấu trúc dữ liệu hậu tố. Bizon Senior thích ”hậu tố automaton”. Bằng cách áp dụng nó một lần cho một chuỗi, anh ta có thể loại bỏ từ chuỗi này bất cứ ký tự đơn nào. Bizon Giữa biết rõ về mảng hậu tố . Bằng cách áp dụng nó một lần cho một chuỗi, ông có thể trao đổi bất kỳ hai ký tự của chuỗi này. Với những ai không biết gì về cây hậu tố, nó có thể giúp họ làm nhiều hơn nữa.

Có lẽ, các giải pháp không yêu cầu cả hai cấu trúc dữ liệu. Tìm hiểu xem những ai có thể giải quyết vấn đề và nếu họ có thể, làm thế nào để họ làm điều đó? Họ có thể giải quyết nó, hoặc chỉ với việc sử dụng hậu tố, hậu tố automaton, hoặc chỉ với việc sử dụng các mảng hậu tố hoặc họ cần cả hai cấu trúc? Lưu ý rằng bất kỳ cấu trúc có thể được sử dụng không giới hạn số lần, các cấu trúc có thể được sử dụng trong bất kỳ thứ tự.

Input

Dòng đầu tiên chứa một từ không rỗng s. Dòng thứ hai chứa một từ t không rỗng. Các từ s và t là khác nhau. Mỗi từ chỉ gồm các chữ cái tiếng Anh chữ thường. Mỗi từ có chứa ít nhất 100 chữ.

Output

Với mỗi dòng, in câu trả lời cho vấn đề. In ra "need tree" nếu từ s không thể được chuyển thành chữ t thậm chí với việc sử dụng cả hai mảng hậu tố và hậu tố automaton. In "automaton" nếu bạn chỉ cần automaton hậu tố để giải quyết vấn đề. In "array" nếu bạn chỉ cần mảng hậu tố để giải quyết vấn đề. In "both" (không có dấu ngoặc kép), nếu bạn cần cả hai cấu trúc dữ liệu để giải quyết vấn đề.

Nó đảm bảo rằng nếu bạn có thể giải quyết vấn đề chỉ với việc sử dụng các mảng hậu tố, sau đó nó là không thể giải quyết nó chỉ với việc sử dụng hậu tố automaton. Điều này cũng đúng đối với hậu tố automaton.

Example

Input:

automaton

tomat

Output:
automaton