**FAST PATTERN MATCHING IN STRINGS**

**BÁO CÁO VỀ THUẬT TOÁN, CODE TRONG CHAPTER 9**

GIỚI THIỆU:

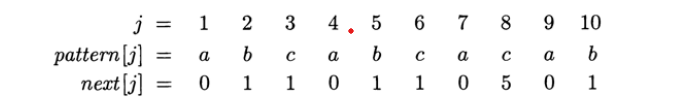
Trong lập trình, chắc hẳn ít nhiều chúng ta cũng nghe về vấn đề xét sự so khớp chuỗi, cụ thể là chúng ta xem thử chuỗi này có đang nằm trong chuỗi kia hay không. Và khi gặp những vấn đề như thế này, những bạn sinh viên CNTT ban đầu đều suy nghĩ ra một phương pháp đó chính là chúng ta sẽ trượt chuỗi cần xét trên chuỗi kia, và mỗi lần như thế chúng ta sẽ so sánh từng kí tự của hai chuỗi với nhau, nếu như chuỗi mẫu hoàn toàn nằm trong chuỗi kia thì chúng ta return true và kết thúc chương trình, còn nếu tại một vị trí mà kí tự của 2 chuỗi khác nhau thì chúng ta sẽ ngừng vòng lặp và tiến hành tịnh tiến chuỗi mẫu lên thêm 1 lần nữa cho đến khi nó khớp hoặc đến cuối vị trí của chuỗi kia. Tuy nhiên cách làm này thực sự không hiệu quả với những chuỗi có kích thước lớn, vì độ phức tạp của cách làm này là O(n \* m) trong đó n là độ dài chuỗi mẫu và m là độ dài chuỗi còn lại. Vì vậy trong chapter này chúng ta sẽ cùng nghiên cứu phương pháp để giảm thiểu độ phức tạp của thuật toán so khớp chuỗi này cụ thể độ phức tạp của thuật toán trong Chapter này sẽ là O(n + m).

THUẬT TOÁN:

Trước tiên, ta gọi+ pattern là chuỗi cần đi so khớp , gọi j là index của các kí tự chuỗi pattern

+ text là chuỗi mà để chuỗi pattern đi so khớp, gọi k là index của các kí tự trong chuỗi text

Trước khi chúng ta tiến hành so khớp chuỗi, chúng ta sẽ đi tạo ra mảng next trước, mảng next là mảng mà tại vị trí j của chuỗi next[j] chính là vị trí cần so sánh tiếp theo nếu như tại vị trí j này next[j] != text[k].

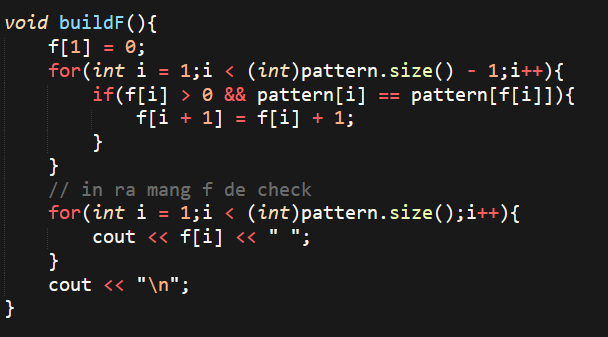
Bản chất của mảng next[j] này đó chính là tại một vị trí j mà pattern[j] != text[k] thì chúng ta không cần phải dịch pattern lên thêm 1 đơn vị để so sánh lại mà chúng ta có thể dịch pattern lên j – next[j] đơn vị, nhờ vậy mà độ phức tạp của thuật toán được giảm đi đáng kể. Đây chính là một ví dụ mẫu về mảng next[j].

Và cách để tính toán mảng next này chỉ tốn O(m) bước với m là độ dài của chuỗi pattern và trước khi đi tính mảng next chúng ta sẽ cùng nhau tạo ra mảng f, mảng f ở đây chính là tiền đề để tạo ra mảng next.

Bây giờ, mục đích chính của mảng next đó chính là lưu lại vị trí i lớn nhất mà chuỗi từ pattern[1] …pattern[i – 1] đã khớp với text để mà khi đó chúng ta sẽ so sánh text[k] với pattern[next[j]] thôi và hiệu quả là chúng ta đã bỏ qua i – 1 vị trí. Để dễ dàng hơn trong việc tạo ra mảng next, chúng ta sẽ đi tạo ra mảng f, mảng f ở đây chính là tại một vị trí j bất kì f[j] = i trong đó i là lớn nhất và nhỏ hơn j sao cho



Và chúng ta thấy điều kiện này hoàn toàn đúng với i bằng 1, vì vậy chúng ta sẽ có f[j] >= 1 với mọi j > 1. Theo quy ước, chúng ta sẽ để f[1] = 0. Và giờ chúng ta sẽ đi tạo ra mảng f

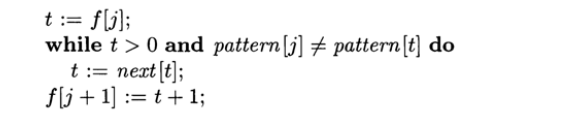


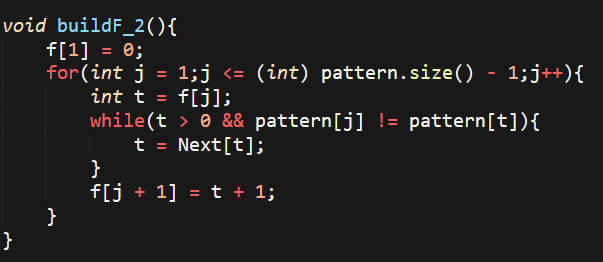
Đây chính là một vấn đề đã từng xuất hiện ở trước đây cụ thể đó chính là ta tính j lớn nhất nhỏ hơn k sao cho

pattern[1] … pattern[j – 1] = text[k – j + 1] … text[k – 1]

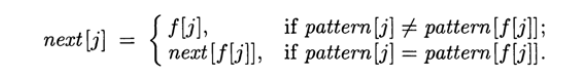
vì vậy chúng ta sẽ sử dụng nó cho vấn đề hiện tại. Và khi chúng ta thực hiện thuật toán trên, chúng ta đã có thể tính toán được mảng f rồi.

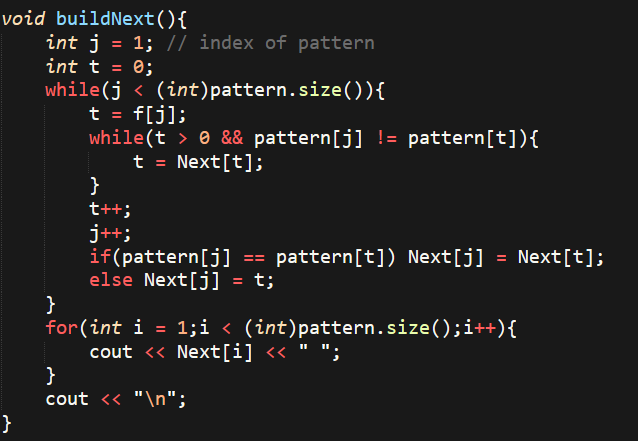
Và em bổ sung thêm một chương trình để tính mảng f, chương trình này sẽ tính f[j + 1] và giả sử rằng f[j] và next[1] …next[j – 1] đã được tính. Thuật toán:





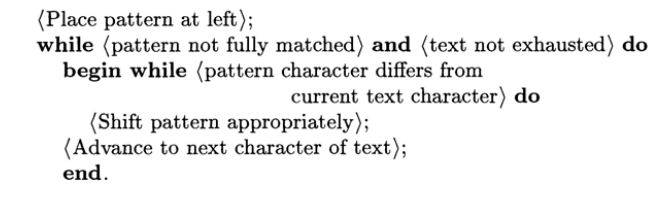
Và sau khi chúng ta đã tính toán được mảng f, chúng ta sẽ cùng nhau đi tạo mảng next, chúng ta có công thức sau:



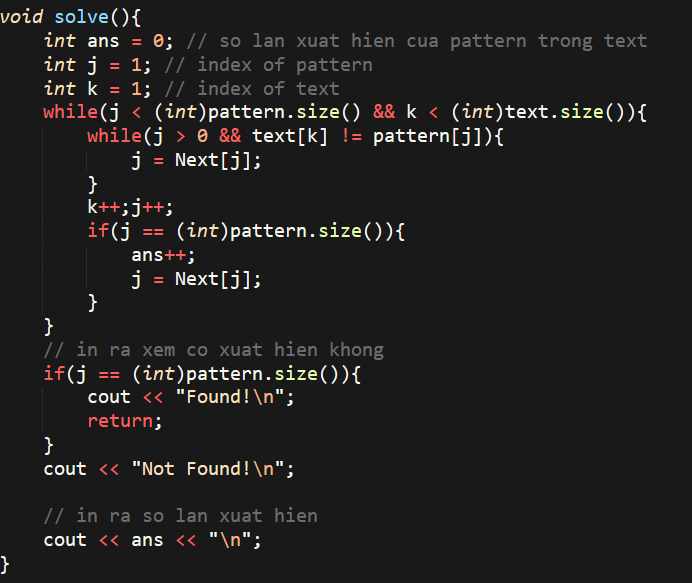
Thuật toán xây dựng mảng next:

Độ phức tạp thuật toán: O(m) với m là độ dài của chuỗi pattern

Và sau khi chúng ta đã có mảng next, việc duyệt so khớp sẽ diễn ra một cách dễ dàng:



Đây là thuật toán trong bài báo, và thuật toán này chỉ để xét xem chuỗi pattern có xuất hiện trong chuỗi text không hay thôi, và em sẽ mở rộng ra để xét thêm chuỗi pattern xuất hiện trong chuỗi text bao nhiêu lần. Thuật toán:



Độ phức tạp của thuật toán: O(m + n) với m là độ dài của chuỗi pattern và n là độ dài của chuỗi text.

Cải tiến độ hiệu quả:

Chúng em vừa mới trình bày thuật toán so khớp chuỗi ở trên, nhưng hình thức này dường như không hiệu quả lắm. Và bây giờ chúng ta sẽ đi tìm hiểu một thuật toán mới được triển khai tốt hơn, và đây chính là thuật toán:

