HCMUTE

FAST PATTERN MATCHING IN STRINGS

James H.Morris, Jr
Vaughan R.Pratt
Originally published in SIAM Journal on Computing 6
Group 8
Nguyen Thai Van - 21110939
Huynh Nhat Nam - 21110903
Nguyen Trong Phuc - 21110915

MỤC TIÊU CHÍNH

Thuật toán tìm tất cả số lần xuất hiện của một chuỗi bên trong một chuỗi khác

1. Tạo mảng f

Ta gọi f[j] = i sao cho i là vị trí lớn nhất bé hơn j để mà:

$$pattern[1] \dots pattern[i-1] = pattern[j-i+1] \dots pattern[j-1].$$

Theo quy ước, chúng ta sẽ để f[1] = 0, với mọi j > 1 thì f[1] >= 1, ta sẽ đi tính mảng f thông qua thuật toán sau:

$$\text{If } f[j] > 0 \text{ and } pattern[j] = pattern[f[j]] \text{ then } f[j+1] = f[j] + 1;$$

Thuật toán build mảng f C++:

```
void buildF(){
    f[1] = 0;
    for(int i = 1;i < (int)pattern.size() - 1;i++){
        if(f[i] > 0 && pattern[i] == pattern[f[i]]){
            f[i + 1] = f[i] + 1;
        }
    }
    // in ra mang f de check
    for(int i = 1;i < (int)pattern.size();i++){
        cout << f[i] << " ";
    }
    cout << "\n";
}
```

Ngoài cách tính mảng f như trên thì ta còn một cách tính mảng f nữa, chương trình này sẽ tính f[j + 1] với điều kiện f[j] và next[1] ... next[j - 1] đã được tính

$$\begin{aligned} &t := f[j]; \\ &\textbf{while } t > 0 \textbf{ and } pattern[j] \neq pattern[t] \textbf{ do} \\ &\quad t := next[t]; \\ &f[j+1] := t + 1; \end{aligned}$$

2. Tạo mảng next

Ta gọi next[j] = i với i là vị trí cần được so sánh khi mà pattern[j] != text[k], điều này có nghĩa là độ phức tạp của thuật toán sẽ được giảm đi đáng kể vì nó không cần phải xét những phần dư thừa.

Chúng ta có công thức sau:

$$next[j] = \begin{cases} f[j], & \text{if } pattern[j] \neq pattern[f[j]]; \\ next[f[j]], & \text{if } pattern[j] = pattern[f[j]]. \end{cases}$$

Thuật toán để xây dựng nên mảng next:

$$\begin{aligned} &j := 1;\ t := 0;\ next[1] := 0; \\ &\textbf{while } j < m \textbf{ do} \\ &\quad \textbf{begin comment } t = f[j]; \\ &\qquad \textbf{while } t > 0 \textbf{ and } pattern[j] \neq pattern[t] \textbf{ do} \\ &\qquad\quad t := next[t]; \\ &\qquad t := t + 1;\ j := j + 1; \\ &\textbf{if } pattern[j] = pattern[t] \textbf{ then } next[j] := next[t] \\ &\textbf{else } next[j] := t; \\ &\textbf{end}. \end{aligned}$$

3. Duyệt sự so khớp chuỗi

Ta gọi chuỗi pattern là chuỗi cần so khớp, index được đánh bởi j
Ta gọi chuỗi text là chuỗi để xét sự so khớp, index được đánh bởi k

Thuật toán tiến hành so khớp:

$$\begin{aligned} &j := k := 1; \\ &\textbf{while } j \leq m \textbf{ and } k \leq n \textbf{ do} \\ &\quad \textbf{begin while } j > 0 \textbf{ and } text[k] \neq pattern[j] \textbf{ do} \\ &\qquad j := next[j]; \\ &\quad k := k + 1;\ j := j + 1; \\ &\quad \textbf{end}. \end{aligned}$$

4. Cải tiến độ hiệu quả

Vừa rồi chúng ta vừa mới giới thiệu qua thuật toán so khớp chuỗi một cách nhanh chóng, tuy nhiên thuật toán này chưa thực sự hiệu quả. Sau đây là một thuật toán so khớp hiệu quả hơn:

$$\begin{aligned} &\quad a := pattern[1];\ pattern[m+1] := \text{`@'};\ next[m+1] := -1; \\ &\quad k := 1;\ text[n+1] := \text{`}\perp\text{'};\ text[n+2] := a; \\ &\text{get started: } j := 1; \\ &\text{quick test: } \textbf{while } text[k] \neq a \textbf{ do } k := k + 1; \\ &\text{end check: } \textbf{if } k > n \textbf{ then go to } \text{input exhausted}; \\ &\text{char matched: } j := j + 1;\ k := k + 1; \\ &\text{loop: } \textbf{comment } j > 0; \\ &\quad \textbf{if } text[k] = pattern[j] \textbf{ then go to } \text{char matched}; \\ &\quad j := next[j]; \\ &\quad \textbf{if } j = 1 \textbf{ then go to } \text{quick test}; \\ &\quad \textbf{if } j = 0 \textbf{ then} \\ &\qquad \textbf{begin } k := k + 1;\ \textbf{go to } \text{get started}; \\ &\qquad \textbf{end}; \\ &\quad \textbf{if } j > 0 \textbf{ then go to } \text{loop}; \\ &\quad \textbf{comment } text[k-m] \text{ through } text[k-1] \text{ have been matched}; \end{aligned}$$

Tài liệu

[1] ALFRED V. AHO, JOHN E. HOPCROFT AND JEFFREY D. ULLMAN, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974.
[2] ALFRED V. AHO AND MARGARET J. CORASICK, Efficient string matching: An aid to bibliographic search, , Comm. ACM, 18 (1975), pp. 333-340.
[3] M. BEELER, R. W. GOSPER AND R. SCHROEPPEL, HAKMEM, , Memo No. 239, M.I.T. Artificial Intelligence Laboratory, Cambridge, Mass., 1972.
[4] ROBERT S. BOYER AND J. STROTHER MOORE, a fast string searching algorithm, manuscript dated December 29, 1975; Stanford Research Insti- tute, Menlo Park, Calif., and Xerox Palo Alto Research Center, Palo Alto, Calif.