**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN I**



**BÁO CÁO**

**MÔN: CHUYÊN ĐỀ CÔNG NGHỆ PHẦN MỀM**

Giảng viên hướng dẫn: Nguyễn Duy Phương

Sinh viên :

MSV :

*Hà Nội, 2020*

**MỤC LỤC**

[1 Thuật toán Brute-Force 2](#_Toc429239876)

[2 Thuật toán Karp-Rabin 5](#_Toc429239877)

[3 Thuật toán Morris-Pratt 10](#_Toc429239878)

[4 Thuật toán Knuth-Morris-Pratt 15](#_Toc429239879)

[5 Thuật toán Boyer-Moore 20](#_Toc429239880)

[6 Thuật toán Tuned Boyer-Moore 27](#_Toc429239881)

[7 Thuật toán Horspool 31](#_Toc429239882)

[8 Thuật toán Shift Or 34](#_Toc429239883)

[9 Thuật toán Raita 35](#_Toc429239884)

[10 Thuật toán Turbo Boyer-Moore 39](#_Toc429239885)

[11 Thuật toán Quick Search 44](#_Toc429239886)

[12 Thuật toán Zhu-Takaoka 46](#_Toc429239887)

[13 Thuật toán Berry – Ravindran 48](#_Toc429239888)

[14 Thuật toán Colussi 51](#_Toc429239889)

[15 Thuật toán Galil-GianCarlo 58](#_Toc429239890)

[16 Thuật toán Apostolico-Crochemore 62](#_Toc429239891)

[17 Thuật toán Not So Naive 66](#_Toc429239892)

[18 Thuật toán Colussi Reverse 71](#_Toc429239893)

[19 Thuật toán String Matching on Ordered Alphabets 75](#_Toc429239894)

[20 Thuật toán Two Way 78](#_Toc429239895)

# Thuật toán Brute-Force

## Những đặc điểm chính

* Không có giai đoạn tiền xử lý
* Cần thêm không gian liên tục
* Luôn di chuyển của sổ so sanh 1 vị trí sang phải
* So sánh có thể hoàn thành trong thứ tự bất kỳ
* Độ phức tạp thời gian trong giai đoạn tìm kiếm là O(m.n)
* So sánh khoảng 2n ký tự

## Mô tả

Thuật toán Brute Force kiểm tra ở tất cả các vị trí trong văn bản giữa 0 và n-m, bất kể có xuất hiện một pattern hay không. Mỗi bước so khớp dịch một vị trí sang phải.

Thuật toán Brute Force không cần giai đoạn tiền xử lý cũng như các mảng phụ cho quá trình tìm kiếm. Độ phức tạp tính toán của thuật toán này là O(m.n).

***Input :***

• Xâu mẫu X =(x0, x1,..,xm), độ dài m.

• Văn bản nguồn Y=(y1, y2,..,yn) độ dài n.

***Output:***

• Mọi vị trí xuất hiện của X trong Y.

***Formats:***

Brute-Force(X, m, Y, n);

## Mã hóa

void BruteForce(char \*x, int m, char \*y, int n) {

int i, j;

/\* Searching \*/

for (j = 0; j <= n - m; ++j) {

for (i = 0; i < m && x[i] == y[i + j]; ++i);

if (i >= m)

cout<<(j); // Output

}

}

## Kiểm nghiệm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
| 1 | 2 | 3 | 4 |  | | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | 1 |  | | | | | | | | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | 1 |  | | | | | | | | | | | | | | | | | | | | |
|  | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | 1 |  | | | | | | | | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | 1 |  | | | | | | | | | | | | | | | | | | |
|  | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | 1 |  | | | | | | | | | | | | | | | | |
|  | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eighth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | 1 |  | | | | | | | | | | | | | | | |
|  | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ninth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | 1 | 2 |  | | | | | | | | | | | | | |
|  | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | 1 |  | | | | | | | | | | | | | |
|  | | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eleventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | 1 | 2 |  | | | | | | | | | | | |
|  | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Twelfth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | 1 |  | | | | | | | | | | | |
|  | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Thirteenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | 1 | 2 |  | | | | | | | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourteenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | 1 |  | | | | | | | | | |
|  | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifteenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixteenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventeenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 1 |  | | | | | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |
|  | | | | | | | | | | | | | | | |  |  |  |  |  |  |  |  |

# Thuật toán Karp-Rabin

## Những đặc điểm chính

* Sử dụng hàm băm
* Giai đoạn tiền xử lý có độ phức tạp thời gian O(m) và độ phức tạp không gian hằng số
* Giai đoạn tìm kiếm có độ phức tạp thời gian O(m.n)
* Thời gian chạy dự đoán là O(m+n)

## Mô tả

Hàm băm cung cấp phương pháp đơn giản để tránh việc so sánh ký tự trùng nhau trong tình huống thực tế nhất. Thay vì kiểm tra tại mỗi vị trí của văn bản nếu có pattern, cách hiệu quả hơn để kiểm tra là chỉ khi các nội dung nội dung của cửa sổ trông giống như pattern. Để kiểm tra sự giống nhau giữa 2 đoạn này, hàm băm được sử dụng.

Ta có hàm băm :

hash(*w*[0 .. *m*-1])=(*w*[0]\*2*m*-1+ *w*[1]\*2*m*-2+···+ *w*[*m*-1]\*20) mod *q*

rehash(*a*,*b*,*h*)= ((*h*-*a*\*2*m*-1)\*2+*b*) mod *q*

Pha tiền xử lý có độ phức tạp O(m). Tuy nhiên thời gian tìm kiếm lại tỷ lệ với O(m.n) vì có nhiều trường hợp hàm băm bị lừa và không phát huy tác dụng. Tuy nhiên đó là trường hợp đặc biệt, trong thực tế thời gian tính toán thường tỉ lệ thuận với O(m+n). Hơn nữa thuất toán Karp Rabin có thể dễ dàng mở rộng cho các mẫu, văn bản dạng 2 chiều, hữu ích cho các thuật toán xử lý ảnh.

***Input :***

• Xâu mẫu X =(x0, x1,..,xm), độ dài m.

• Văn bản nguồn Y=(y1, y2,..,yn) độ dài n.

***Output:***

• Mọi vị trí xuất hiện của X trong Y.

***Formats:***

Brute-Force(X, m, Y, n);

## Mã hóa

#define REHASH(a, b, h) ((((h) - (a)\*d) << 1) + (b))

void KR(char \*x, int m, char \*y, int n) {

int d, hx, hy, i, j;

/\* Preprocessing \*/

/\* computes d = 2^(m-1) with

the left-shift operator \*/

for (d = i = 1; i < m; ++i)

d = (d<<1);

for (hy = hx = i = 0; i < m; ++i) {

hx = ((hx<<1) + x[i]);

hy = ((hy<<1) + y[i]);

}

/\* Searching \*/

j = 0;

while (j <= n-m) {

if (hx == hy && memcmp(x, y + j, m) == 0)

cout<<(j);

hy = REHASH(y[j], y[j + m], hy);

++j;

}

}

## Kiểm nghiệm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

*hash*(*y*[0 .. 7]) = 17819

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |

*hash*(*y*[1 .. 8]) = 17533

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | |

*hash*(*y*[2 .. 9]) = 17979

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

*hash*(*y*[3 .. 10]) = 19389

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | |

*hash*(*y*[4 .. 11]) = 17339

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

*hash*(*y*[5 .. 12]) = 17597

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | |

*hash*(*y*[6 .. 13]) = 17102

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eighth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | |

*hash*(*y*[7 .. 14]) = 17117

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ninth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | |

*hash*(*y*[8 .. 15]) = 17678

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | |

*hash*(*y*[9 .. 16]) = 17245

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eleventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | |

*hash*(*y*[10 .. 17]) = 17917

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Twelfth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | | |

*hash*(*y*[11 .. 18]) = 17723

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Thirteenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

*hash*(*y*[12 .. 19]) = 18877

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourteenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | |

*hash*(*y*[13 .. 20]) = 19662

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifteenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

*hash*(*y*[14 .. 21]) = 17885

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixteenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  |

*hash*(*y*[15 .. 22]) = 19197

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventeenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

*hash*(*y*[16 .. 23]) = 16961

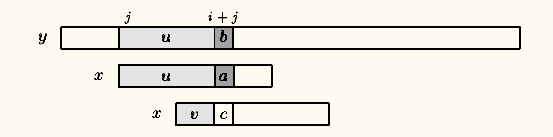
# Thuật toán Morris-Pratt

## Những đặc điểm chính

* Thực hiện việc so sanh từ trái qua phải
* Pha tiền xử lý có độ phức tạp không gian và thời gian là O(m)
* Pha tiền xử lý có độ phức tạp thời gian là O(m+n)
* Thực thi 2n-1 thông tin thu thập được trong quá trình quét văn bản
* Độ trễ m (số lượng tối đa các lần so sánh ký tự đơn)

## Mô tả

Thuật toán MP cải tiến thuật toán Brute Force, thay vì dịch chuyển từng bước một, phí công các ký tự đã so sánh trước đó, ta tìm cách dịch x đi một đoạn xa hơn.



Hình 3.1 : Dịch chuyển trong thuật toán Morris-Pratt

Giả sử tại bước so sánh bất kỳ, ta có một pattern “*u*” trùng nhau giữa x và y, tại x[i] != y[j+i] ( *a* != *b*), thay vì dịch chuyển 1 bước sang phải, ta cố gắng dịch chuyển dài hơn sao cho một tiền tố (prefix) *v* của x trùng với hậu tố (suffix) của u.

Ta có mảng mpNext[] để tính trước độ dài trùng nhau lớn nhất giữa tiền tố và hậu tố trong x, khi so sánh với y tại vị trí thứ i, x sẽ trượt một khoảng = i – mpNext[i].

Việc tính toán mảng mpNext[] có độ phức tạp thời gian và không gian là O(n). Giai đoạn tìm kiếm sau đó có độ phức tạp thời gian là O(m+n).

## Mã hóa

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

#include <iostream>

using namespace std;

#define MAX 12

int mpNext[MAX];

void Init()

{

for(int i = 0; i < MAX; i++)

mpNext[i] = 999;

}

void preMp(char \*x, int m) {

int i, j;

i = 0; //mang mpNext the hien do dai trung nhau lon

j = mpNext[0] = -1; //nhat giua tien to va hau to

while (i < m) {

while (j > -1 && x[i] != x[j])

{

j = mpNext[j]; //chay nguoc xet xem do dai lon nhat cua

//vi tri giong voi x[i]

}

i++;

j++;

mpNext[i] = j;

int a = 2;

}

}

void MP(char \*x, int m, char \*y, int n) {

int i, j;// mpNext[m];

//int mpNext[8];

/\* Preprocessing \*/

Init();

preMp(x, m);

for(int k =0;k<m;k++){

cout<<x[k]<<" "<<mpNext[k]<<endl;

}

/\* Searching \*/

i = j = 0;

while (j < n) {

while (i > -1 && x[i] != y[j])

i = mpNext[i];

i++;

j++;

if (i >= m) {

cout<<j - i;

i = mpNext[i];

}

}

}

void main()

{

char \*x = "GCAGAGAG"; //"ATCACATCATCA ";

int m = strlen(x);

char \*y = "GCATCGCAGAGAGTATACAGTACG"; //"AGTATCATCACATCATCAGA";

int n = strlen(y);

MP(x, m, y, n);

}

## Kiểm nghiệm

***Kiểm nghiệm pha tiền xử lý( thuật toán preMp)***

x[] = GCAGAGAG

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **x[j]** | **x[i]** | **i** | **j** | **mpNext[i]** | **ghi chú** |
|  |  | 0 | -1 |  |  |
|  | G | 0 | -1 | -1 | =>mpNext[1] =0 |
| G | C | 1 | 0  -1 | 0 |  |
| G | A | 2 | 0  -1 | 0 |  |
| G | G | 3 | 0 | 0 |  |
| C  G | A  A | 4 | 1  0  -1 | 1 |  |
| G | G | 5 | 0 | 0 |  |
| C  G | A  A | 6 | 1  0  -1 | 1 |  |
| G | G | 7 | 0 | 0 |  |
|  |  | 8 | 1 | 1 |  |

Ta được bảng mpNext[]

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| x[i] | G | C | A | G | A | G | A | G |  |
| mpNext[i] | -1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |

***Kiểm nghiệm Pha tìm kiếm***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
| 1 | 2 | 3 | 4 |  | | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 3 (*i*-*mpNext*[*i*]=3-0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | 1 |  | | | | | | | | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

Shift by: 1 (*i*-*mpNext*[*i*]=0- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | 1 |  | | | | | | | | | | | | | | | | | | |
|  | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | |

Shift by: 1 (*i*-*mpNext*[*i*]=0- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 7 (*i*-*mpNext*[*i*]=8-1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | 1 |  | | | | | | | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

Shift by: 1 (*i*-*mpNext*[*i*]=1-0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | 1 |  | | | | | | | | | |
|  | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | |

Shift by: 1 (*i*-*mpNext*[*i*]=0- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

Shift by: 1 (*i*-*mpNext*[*i*]=0- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eighth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  |

Shift by: 1 (*i*-*mpNext*[*i*]=0- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ninth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 1 |  | | | | | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

Shift by: 1 (*i*-*mpNext*[*i*]=0- -1)

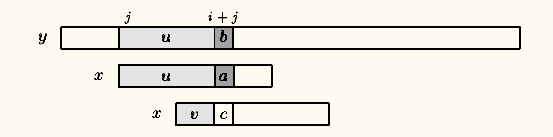
# Thuật toán Knuth-Morris-Pratt

## Những đặc điểm chính

* Thực hiện so sánh từ trái sang phải
* Pha tiền xử lý có độ phức tạp không gian và thời gian là O(m)
* Pha tìm kiếm có độ phức tạp thời gian O(m+n)
* Độ trễ logPhi(*m*) với Phi là tỷ lệ vàng (golden ratio)

## Mô tả

Thuật toán là bản đơn giản và xử lý tương tự như thuật toán Morris-Pratt khi cố gắng dịch chuyển một đoạn dài nhất sao cho một tiền tố (prefix) *v* của x trùng với hậu tố (suffix) của u.



Điểm khác nhau là KMP sẽ thực hiện thêm so sánh *c* và *b*, có nghĩa KMP sẽ thực hiện một pha dòm trước ký tự bên phải đoạn đang so khớp. Do đó mỗi bước KMP sẽ dịch chuyển thêm một bước sang phải so với MP nếu *c != b*

## Mã hóa

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

#include <iostream>

using namespace std;

#define MAX 9

int kmpNext[MAX + 1];

void Init()

{

for(int i = 0; i < MAX + 1; i++)

kmpNext[i] = 999;

}

void preKMP(char \*x, int m)

{

int i, j;

i = 0;

j = kmpNext[0] = -1;

while (i < m)

{

while (j > -1 & x[i] != x[j])

{

j = kmpNext[j];

}

i++;

j++;

if(x[i] == x[j])

kmpNext[i] = kmpNext[j];

else

kmpNext[i] = j;

}

}

void KMP(char \*x, int m, char \*y, int n)

{

int i,j;

Init();

/\*Preprocessing\*/

preKMP(x, m);

/\*Searching\*/

i = j = 0;

while (j < n)

{

while (i > -1 && y[j] != x[i])

{

i = kmpNext[i];

}

i++;

j++;

if(i >= m)

{

cout<<j-i;

i = kmpNext[i];

}

}

}

void main()

{

char \*y = "GCATCGCAGAGAGTATACAGTACG"; //"ABABDABACDABABCABAB";

int n = strlen(y);

char \*x = "GCAGAGAG"; //"ABABCABAB";

int m = strlen(x);

KMP(x, m, y, n);

}

## Kiểm nghiệm

***Kiểm nghiệm pha tiền xử lý ( thuật toán preKMP )***

x[] = GCAGAGAG

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **x[j]** | **x[i]** | **i** | **j** | **mpNext[i]** | **ghi chú** |
|  |  | 0 | -1 |  |  |
| \0 | G | 0 | -1 | -1 | =>mpNext[1] =0 |
| G | C | 1 | 0  -1 | 0 |  |
| G | A | 2 | 0  -1 | 0 |  |
| G | G | 3 | 0 | -1 | kmpNext[i] = kmpNext[j]; |
| C  G | A  A | 4 | 1  0  -1 | 1 |  |
| G | G | 5 | 0 | -1 | kmpNext[i] = kmpNext[j]; |
| C  G | A  A | 6 | 1  0  -1 | 1 |  |
| G | G | 7 | 0 | -1 | kmpNext[i] = kmpNext[j]; |
| C | \0 | 8 | 1 | 1 |  |

Ta được bảng kmpNext[]

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| x[i] | G | C | A | G | A | G | A | G |  |
| kmpNext[i] | -1 | 0 | 0 | -1 | 1 | -1 | 1 | -1 |  |

***Kiểm nghiệm Pha tìm kiếm***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
| 1 | 2 | 3 | 4 |  | | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 4 (*i*-*kmpNext*[*i*]=3- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | 1 |  | | | | | | | | | | | | | | | | | | |
|  | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | |

Shift by: 1 (*i*-*kmpNext*[*i*]=0- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 7 (*i*-*kmpNext*[*i*]=8-1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | 1 |  | | | | | | | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

Shift by: 1 (*i*-*kmpNext*[*i*]=1-0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | 1 |  | | | | | | | | | |
|  | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | |

Shift by: 1 (*i*-*kmpNext*[*i*]=0- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

Shift by: 1 (*i*-*kmpNext*[*i*]=0- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  |

Shift by: 1 (*i*-*kmpNext*[*i*]=0- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eighth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 1 |  | | | | | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

Shift by: 1 (*i*-*kmpNext*[*i*]=0- -1)

# Thuật toán Boyer-Moore

## Những đặc điểm chính

* Thực hiện so sánh từ phải sang trái
* Pha tiền xử lý có độ phức tạp không gian và thời gian ***O***(*m*+)
* Pha tìm kiếm có độ phức tạp thời gian O(m.n)
* Trường hợp xấu nhất thực hiện so sánh 3n ký tự khi tìm kiếm mẫu không chu kỳ
* Hiệu suất tốt nhất O(n/m)

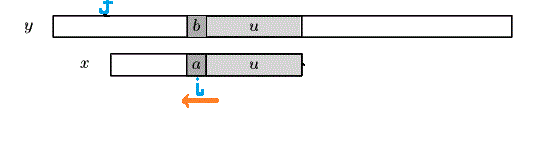
## Mô tả

Thuật toán Boyer-Moore được coi là thuật toán hiệu quả nhất trong vấn đề tìm kiếm chuỗi trong các ứng dụng thường gặp. Các biến thể của nó được dùng trong các bộ soạn thảo cho các lệnh như <<search> và <<subtitle>>.

Thuật toán sẽ quét các ký tự của mẫu(pattern) từ phải sang trái bắt đầu ở phần tử cuối cùng.

Trong trường hợp mis-match (tìm được đoạn khớp với mẫu), nó sẽ dùng 2 hàm được tính toán trước để dịch cửa sổ sang phải. Hai hàm dịch chuyển này là good-suffix shift (dịch chuyển khớp) và bad-character shift (dịch chuyển xuất hiện).

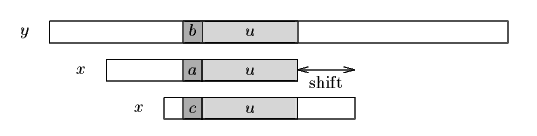
Giả sử trong quá trình so sánh ta gặp mis-match tại x[i] = a và y[j+i] = b khi khớp tại vị trí thứ j



Ta có các trường hợp dịch chuyển như sau:

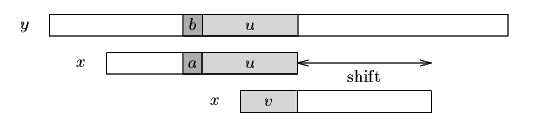
1. Good-suffix shift

* Đoạn khớp *u* lại xuất hiện ở một vị trí khác trong x, dịch chuyển sang phải cho



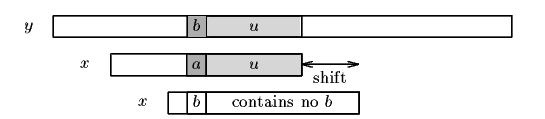
tới khi gặp ký tự khác với x[i], (c != a)

* Đoạn *u* không xuất hiện lại trong x, mà chỉ có phần hậu tố của u khớp với phần tiền tố của x, ta dịch một đạn sao cho phần hâụ tố dài nhất v khớp với phần tiền tố của x.

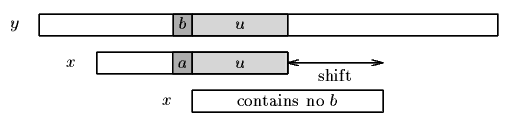


1. Bad-character shift

* Trong trường hợp không thể dịch chuyển theo Good-suffix shift. Ta dịch chuyển theo Bad-character shift như sau
* Nếu ký tự y[j+i] = b xuất hiện trong x, ta dịch chuyển sao cho y[j+i] khớp với ký tự bên phải nhất trong đoạn x[0 … m-2]



* Nếu ký tự y[j+i] = b không xuất hiện trong x, ta dịch chuyển hết x sang ngay sau y[j+1]



* Thuật toán Boyer-Moore sẽ chọn đoạn dịch chuyên dài nhất trong 2 hàm dịch chuyển Good-suffix shift và Bad-character shift. Hai hàm này tạo ra 2 bảng bmGs và bmBc
* Bảng bmGs và bmBc được tính toán trong thời gian ***O***(*m*+) trước khi thực hiện việc tìm kiếm và cần 1 không gian phụ là ***O***(*m*+). Giai đoạn tìm kiếm có độ phức tạp thời gian bậc hai nhưng lại chỉ có 3n phép so sánh khi tìm kiếm 1 chuỗi không có chu kì. Đối với việc tìm kiếm trong 1 khối lượng lớn các chữ cái thuật toán thực hiện với tốc độ nhanh “khủng khiếp”.  Khi tìm kiếm chuỗi am-1b trong bn chuỗi thuật toán chỉ sử dụng O(n/m) phép so sánh, đây được coi là “hình mẫu” cho bất cứ một thuật toán tìm kiếm chuỗi nào mà mẫu đã được xử lý trước.

## Mã hóa

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

#include <iostream>

using namespace std;

#define ASIZE 255

#define XSIZE 9

int bmBc[ASIZE]; //Mảng bmBc[ ] là mảng chứa vị trí xuất hiện cuối cùng của các ký tự trong xâu x

int bmGs[XSIZE]; //Mảng chứa vị trí có hậu tố trùng nhau hoặc có phần chung giữa 2 xâu

int suff[XSIZE]; //Mảng chứa giá trị xâu hậu tố dài nhất

void Init()

{

for(int i = 0; i < XSIZE; i++)

{

bmGs[i] = 999;

suff[i] = 999;

}

}

void preBmBc(char \*x, int m/\*, int bmBc[]\*/) {

int i;

for (i = 0; i < ASIZE; ++i)

bmBc[i] = m;

for (i = 0; i < m - 1; ++i) // mang bmBC chua vi tri xuat hien cuoi cung cua cac ky tu trong xau X, khong duyet Ky tu ben phai nhat (i < m-1)

bmBc[x[i]] = m - i - 1; // duyet xau tu trai qua phai, vi tri ky tu tinh' tu` phai qua trai. ( AGCG => A = 3; G = 2; C = 1)

}

void suffixes(char \*x, int m/\*, int \*suff\*/) { //xac dinh hau to chung dai nhat giua xau x[0 ... x] va x[0 ... m-1]

int f, g, i;

suff[m - 1] = m;

g = m - 1;

for (i = m - 2; i >= 0; --i) { //duyet tu cuoi xau ve dau xau

if (i > g && suff[i + m - 1 - f] < i - g)

suff[i] = suff[i + m - 1 - f];

else {

if (i < g)

g = i;

f = i;

while (g >= 0 && x[g] == x[g + m - 1 - f]) //neu trung nhau thi dua do dai trung vao

--g;

suff[i] = f - g;

}

}

}

void preBmGs(char \*x, int m/\*, int bmGs[]\*/) {

int i, j/\*, suff[XSIZE]\*/;

suffixes(x, m/\*, suff\*/);

for (i = 0; i < m; ++i)

bmGs[i] = m;

j = 0;

for (i = m - 1; i >= 0; --i)

if (suff[i] == i + 1)

for (; j < m - 1 - i; ++j)

if (bmGs[j] == m)

bmGs[j] = m - 1 - i;

for (i = 0; i <= m - 2; ++i)

bmGs[m - 1 - suff[i]] = m - 1 - i;

}

void BM(char \*x, int m, char \*y, int n) {

int i, j/\*, bmGs[XSIZE], bmBc[ASIZE]\*/;

Init();

/\* Preprocessing \*/

preBmGs(x, m/\*, bmGs\*/);

preBmBc(x, m/\*, bmBc\*/);

/\* Searching \*/

j = 0;

while (j <= n - m)

{

for (i = m - 1; i >= 0 && x[i] == y[i + j]; --i);

if (i < 0) {

cout<<j;

j += bmGs[0];

}

else

{

int a = bmGs[i];

int b = bmBc[y[i + j]] - m + 1 + i;

j += max(a, b);

}

}

}

void main()

{

char \*y = "GCATCGCAGAGAGTATACAGTACG"; //"ABABDABACDABABCABAB";

int n = strlen(y);

char \*x = "GCAGAGAG"; //"ABABCABAB";

int m = strlen(x);

BM(x, m, y, n);

}

## Kiểm nghiệm

x[] = GCAGAGAG, m = 8

1. Giai đoạn tiền xử lý tạo bảng bmBc với thuật toán preBmBC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| c | A | C | G | T |
| bmBC[c] | 1 | 6 | 2 | 8 |

1. Giai đoạn tiền xử lý tạo bảng bmGs với thuật toán preBmGs

* Thuật toán suffixes tạo bảng suff[] chứa giá trị xâu hậu tố dài nhất

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *suff[i]* | *f* | *g* | *suff[i + m -1 –f] < (i-g) ?* | *i>g ?* | *i* |
| 8 | 0 | 7 |  |  | 7 |
| 0 | 6 | 6 |  | False | 6 |
| 4 | 5 | 1 |  | False | 5 |
| 0 | 5 | 1 | True | True | 4 |
| 2 | 3 | 1 | False | True | 3 |
| 0 | 1 | 1 |  | False | 2 |
| 1 | 0 | 0 |  | False | 1 |

Ta được mảng suff[] như sau

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| x[i] | G | C | A | G | A | G | A | G |
| suff[i] | 1 | 0 | 0 | 2 | 0 | 4 | 0 | 8 |

* Thuật toán preBmGs tạo bảng bmGs

*Giai đoạn đầu*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *bmGs[i] = m-1-i* | *j* | *j < m-1-i ?* | *bmGs[j]=m ?* | *Suff[i]= i+1 ?* | *i* |
| 9 | 0 | False |  | True | 7 |
| 8 | 0 |  |  | False | 6 |
| 8 | 0 |  |  | False | 5 |
| 8 | 0 |  |  | False | 4 |
| 8 | 0 |  |  | False | 3 |
| 8 | 0 |  |  | False | 2 |
| 8 | 0 |  |  | False | 1 |
| 7 | 0 | True | True | True | 0 |
| 7 | 1 | True | True |  | 0 |
| 7 | 2 | True | True |  | 0 |
| 7 | 3 | True | True |  | 0 |
| 7 | 4 | True | True |  | 0 |
| 7 | 5 | True | True |  | 0 |
| 7 | 6 | True | True |  | 0 |
| 8 | 7 | False | True |  | 0 |

*Giai đoạn sau*

|  |  |  |  |
| --- | --- | --- | --- |
| *bmGs[m-1-suff[i])* | *i<m-2 ?* | *suff[i]* | *i* |
| bmGs[6] = 7 | True | 1 | 0 |
| bmGs[7] = 6 | True | 0 | 1 |
| bmGs[7] = 5 | True | 0 | 2 |
| bmGs[5] = 4 | True | 2 | 3 |
| bmGs[7] = 3 | True | 0 | 4 |
| bmGs[3] = 2 | True | 4 | 5 |
| bmGs[7] = 1 | True | 0 | 6 |
|  | False |  | 7 |

Ta được mảng bmGs[] như sau

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| x[i] | G | C | A | G | A | G | A | G |
| suff[i] | 1 | 0 | 0 | 2 | 0 | 4 | 0 | 8 |
| bmGs[i] | 7 | 7 | 7 | 2 | 7 | 4 | 7 | 1 |

1. Giai đoạn Tìm kiếm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | 1 |  | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

1. Shift by: 1 = max(*bmGs*[7]=*bmBc*[A]-7+7)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | 3 | 2 | 1 |  | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |

1. Shift by: 4 = max (*bmGs*[5]=*bmBc*[C]-7+5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

1. Shift by: 7 (*bmGs*[0])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | 3 | 2 | 1 |  | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

1. Shift by: 4 = max (*bmGs*[5]=*bmBc*[C]-7+5)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | 2 | 1 |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

1. Shift by: 7 (*bmGs*[6])

# Thuật toán Tuned Boyer-Moore

## Những đặc điểm chính

* Phiên bản đơn giản của thuật toán Boyer-Moore
* Dễ dàng cài đặt
* Rất nhanh trong thực tiễn

## Mô tả

Là phiên bản đơn giản của Boyer-Moore sử dụng hàm Bad-character shift để tìm x[m-1] trong y và giữ trượt tới khi tìm được, thực hiện trượt mù 3 lần mỗi dòng. Ký tự cuối cùng có giá trị = 0 trong bmBc, giá trị shift của ký tự cuối được lưu trữ trước khi set = 0

## Mã hóa

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

#include <iostream>

using namespace std;

#define ASIZE 255

#define XSIZE 9

int bmBc[ASIZE]; //Mảng bmBc[ ] là mảng chứa vị trí xuất hiện cuối cùng của các ký tự trong xâu x

int bmGs[XSIZE]; //Mảng chứa vị trí có hậu tố trùng nhau hoặc có phần chung giữa 2 xâu

int suff[XSIZE]; //Mảng chứa giá trị xâu hậu tố dài nhất

void Init()

{

for(int i = 0; i < XSIZE; i++)

{

bmGs[i] = 999;

suff[i] = 999;

}

}

void preBmBc(char \*x, int m/\*, int bmBc[]\*/) {

int i;

for (i = 0; i < ASIZE; ++i)

bmBc[i] = m;

for (i = 0; i < m - 1; ++i) // mang bmBC chua vi tri xuat hien cuoi cung cua cac ky tu trong xau X, khong duyet Ky tu ben phai nhat (i < m-1)

bmBc[x[i]] = m - i - 1; // duyet xau tu trai qua phai, vi tri ky tu tinh' tu` phai qua trai. ( AGCG => A = 3; G = 2; C = 1)

}

void TUNEDBM(char \*x, int m, char \*y, int n) {

int j, k, shift/\*, bmBc[ASIZE]\*/;

/\* Preprocessing \*/

preBmBc(x, m/\*, bmBc\*/);

shift = bmBc[x[m - 1]]; // m = 8; x[7] = G

bmBc[x[m - 1]] = 0; //bmBc[71] = 0; => A = 1; C = 6; G = 0; T = m = 8; ky tu cuoi cung trong x : G = 0

memset(y + n, x[m - 1], m);

/\* Searching \*/

j = 0;

while (j < n) {

k = bmBc[y[j + m -1]];

while (k != 0)

{

j += k;

k = bmBc[y[j + m -1]];

j += k;

k = bmBc[y[j + m -1]];

j += k;

k = bmBc[y[j + m -1]];

}

char \* buff = y + j;

if (memcmp(x, y + j, m - 1) == 0 && j < n)

cout<<j;

j += shift; /\* shift \*/

}

}

void main()

{

char y[] = "GCATCGCAGAGAGTATACAGTACG"; //"ABABDABACDABAACABABE";

int n = strlen(y);

char \*x = "GCAGAGAG"; //"ABAACABAB";

int m = strlen(x);

TUNEDBM(x, m, y, n);

}

## Kiểm nghiệm

1. Xây dựng bảng bad-character shift bmBC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | A | C | G | T |
| bmBC[a] | 1 | 6 | 0 | 8 |

shift = 2

1. Pha tìm kiếm thực hiện 4 bước cơ bản sau :

* Tính k = bmBC[y[m-1]] //y[m-1] ký tự cuối cùng của cửa sổ trong y
* If(k != 0)
  + Dịch theo k
  + Thực hiện 3 lần tính lại k, dịch theo k
* So sanh pattern và cửa sổ, nếu đúng output
* Dịch theo shift

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | 1 |  | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | 2 |  | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | 3 |  | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | 4 |  | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | 5 |  | | | | | | | | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |

Shift by: 2 (*shift*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | 1 |  | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | 2 |  | | | | | | | | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

Shift by: 2 (*shift*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | 1 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 2 (*shift*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | 2 |  | | | | | | | |
|  | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | 3 |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | 4 |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |
|  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | 5 |  | | | | | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

Shift by: 2 (*shift*)

# Thuật toán Horspool

## Những đặc điểm chính

* Phiên bản đơn giản của Boyer-Moore
* Dễ dàng thực hiện
* Pha tiền xử lý có độ phức tạp thời gian O(m+) và đọ phức tạp không gian O()
* Pha tìm kiếm có độ phức tạp thời gian O(mn)
* Giá trị trung bình của việc so sanh 1 ký tự trong khoảng 1/ và 2/( +1)

## Mô tả

Phiên bản đơn giản của Boyer-More, chỉ dịch chuyển bad-character shift dựa theo ký tự cuối cùng bên phải của cửa sổ y[m-1]

## Mã hóa

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

#include <iostream>

using namespace std;

#define ASIZE 255

#define XSIZE 9

int bmBc[ASIZE]; //Mảng bmBc[ ] là mảng chứa vị trí xuất hiện cuối cùng của các ký tự trong xâu x

int bmGs[XSIZE]; //Mảng chứa vị trí có hậu tố trùng nhau hoặc có phần chung giữa 2 xâu

int suff[XSIZE]; //Mảng chứa giá trị xâu hậu tố dài nhất

void preBmBc(char \*x, int m/\*, int bmBc[]\*/) {

int i;

for (i = 0; i < ASIZE; ++i)

bmBc[i] = m;

for (i = 0; i < m - 1; ++i) // mang bmBC chua vi tri xuat hien cuoi cung cua cac ky tu trong xau X, khong duyet Ky tu ben phai nhat (i < m-1)

bmBc[x[i]] = m - i - 1; // duyet xau tu trai qua phai, vi tri ky tu tinh' tu` phai qua trai. ( AGCG => A = 3; G = 2; C = 1)

}

void HORSPOOL(char \*x, int m, char \*y, int n) {

int j/\*, bmBc[ASIZE]\*/;

char c;

/\* Preprocessing \*/

preBmBc(x, m/\*, bmBc\*/); //tien xu ly

/\* Searching \*/

j = 0;

while (j <= n - m) {

c = y[j + m - 1];

char \* buff = y + j;

if (x[m - 1] == c && memcmp(x, y + j, m - 1) == 0)

cout<<j;

j += bmBc[c];

}

}

void main()

{

char y[] = "GCATCGCAGAGAGTATACAGTACG"; //"ABABDABACDABAACABABE";

int n = strlen(y);

char \*x = "GCAGAGAG"; //"ABAACABAB";

int m = strlen(x);

HORSPOOL(x, m, y, n);

}

## Kiểm nghiệm

1. Bảng bad-character shift bmBc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | A | C | G | T |
| bmBc[a] | 1 | 6 | 2 | 8 |

1. Searching

* Dịch theo bmBC[y[m-1]] , y[m-1] = ký tự cuối cùng bên phải nhất của cửa sổ

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | 1 |  | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

* Shift by: 1 (*bmBc*[A])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | 2 |  |  |  |  |  |  | 1 |  | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |

* Shift by: 2 (*bmBc*[G])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | 2 |  |  |  |  |  |  | 1 |  | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

* Shift by: 2 (*bmBc*[G])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

* Shift by: 2 (*bmBc*[G])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | |

* Shift by: 1 (*bmBc*[A])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | |

* Shift by: 8 (*bmBc*[T])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 2 |  |  |  |  |  |  | 1 |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

* Shift by: 2 (*bmBc*[G])

# Thuật toán Shift Or

## Những đặc điểm chính

* Sử dụng kỹ thuật thao tác bit
* Hiệu quả trong trường hợp mâu nhỏ hơn kích thước bộ nhớ của 1 từ trong máy
* Pha tiền xử lý có độ phức tạp thời gian và không gian là O(m+)
* Pha tìm kiếm có độ phức tạp thời gian là O(n)
* Dễ dàng khi so sánh những chuỗi tương tự nhau

## Mô tả

* N/A

## Mã hóa

#define WORD 64

int preSo(char \*x, int m, unsigned int S[]) {

unsigned int j, lim;

int i;

for (i = 0; i < ASIZE; ++i)

S[i] = ~0;

for (lim = i = 0, j = 1; i < m; ++i, j <<= 1) {

S[x[i]] &= ~j;

lim |= j;

}

lim = ~(lim>>1);

return(lim);

}

void SO(char \*x, int m, char \*y, int n) {

unsigned int lim, state;

unsigned int S[ASIZE];

int j;

if (m > WORD)

cout<<("SO: Use pattern size <= word size"); // error

/\* Preprocessing \*/

lim = preSo(x, m, S);

/\* Searching \*/

for (state = ~0, j = 0; j < n; ++j) {

state = (state<<1) | S[y[j]];

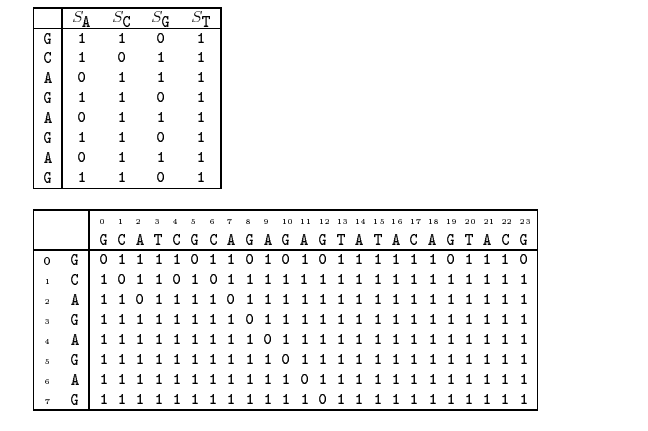
if (state < lim)

cout<<(j - m + 1); // output

}

}

## Kiểm nghiệm



R12[7] = 0, x được tìm tại vị trí j – m + 1 = 12 – 8 + 1 = 5

# Thuật toán Raita

## Những đặc điểm chính

* Trước tiên so sánh các ký tự của các mẫu cuối cùng, sau đó là mẫu đầu tiên và cuối cùng là một trong những so sánh khác nhau ở hiện tại.
* Thực hiện các thay đổi như ở thuật toán Horspool
* Thời gian đưa vào từng bước xử lý trước là O(m + a) và độ phức tạp không gian O(a)
* Độ phức tạp của thời gian tìm kiếm là O(m.n).

## Mô tả

Raita thiết kế một thuật toán mà tại mỗi nỗ lực đầu tiên so sánh các ký tự cuối cùng của mô hình với các ký tự bên phải của cửa sổ, sau đó nếu chúng phù hợp với các so sánh các ký tự đầu tiên của mô hình với các ký tự văn bản ngoài cùng bên trái của cửa sổ, rồi nếu chúng phù hợp với so sánh ký tự văn bản giữa các mô hình với giữa các ký tự của cửa sổ. Và cuối cùng của chúng phù hợp với các ký tự của mô hình khác từ thứ hai đến cuối, nhưng một, có thể so sánh một lần nữa giữa các ký tự này.

Raita quan sát thấy rằng thuật toán của nó đã có một tác động tốt trong thực tế khi tìm kiếm mô hình trong các văn bản tiếng Anh và các tác động đến sự tồn tại của các ký tự phụ thuộc. Smith thực hiện một số thí nghiệm và kết luận rằng hiện tượng này có thể thay thế bỡi các kết quả của trình biên dịch khác.

Giai đoạn tiền xử lý của thuật toán Raita bao gồm trong tính toán các chức năng chuyển đổi xấu ký tự (xem chương 14). Nó có thể được thực hiện trong thời gian O (m + a) và sự phức tạp không gian O (a)

Giai đoạn tìm kiếm của thuật toán Raita có một trường hợp xấu là nhất bậc hai về sự phức tạp thời gian.

## Mã hóa

void preBmBc(char \*x, int m, int bmBc[]) {

int i;

for (i = 0; i < ASIZE; ++i)

bmBc[i] = m;

for (i = 0; i < m - 1; ++i) // mang bmBC chua vi tri xuat hien cuoi cung cua cac ky tu trong xau X, khong duyet Ky tu ben phai nhat (i < m-1)

bmBc[x[i]] = m - i - 1; // duyet xau tu trai qua phai, vi tri ky tu tinh' tu` phai qua trai. ( AGCG => A = 3; G = 2; C = 1)

}

void RAITA(char \*x, int m, char \*y, int n) {

int j, bmBc[ASIZE];

char c, firstCh, \*secondCh, middleCh, lastCh;

/\* Preprocessing \*/

preBmBc(x, m, bmBc);

firstCh = x[0];

secondCh = x + 1;

middleCh = x[m/2];

lastCh = x[m - 1];

/\* Searching \*/

j = 0;

while (j <= n - m) {

c = y[j + m - 1];

if (lastCh == c && middleCh == y[j + m/2] &&

firstCh == y[j] &&

memcmp(secondCh, y + j + 1, m - 2) == 0)

OUTPUT(j);

j += bmBc[c];

}

}

## Kiểm nghiệm

1. Bảng bad-character shift bmBc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | A | C | G | T |
| bmBc[a] | 1 | 6 | 2 | 8 |

1. Searching

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | 1 |  | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

1. Shift by: 1 (*bmBc*[A])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | 2 |  |  |  |  |  |  | 1 |  | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |

1. Shift by: 2 (*bmBc*[G])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | 2 |  |  |  |  |  |  | 1 |  | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

1. Shift by: 2 (*bmBc*[G])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 2 | 4 | 5 | 6 | 3 | 8 | 9 | 1 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

1. Shift by: 2 (*bmBc*[G])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | |

1. Shift by: 1 (*bmBc*[A])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | |

1. Shift by: 8 (*bmBc*[T])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 2 |  |  |  |  |  |  | 1 |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

1. Shift by: 2 (*bmBc*[G])

# Thuật toán Turbo-BM

## Những đặc điểm chính

* Đây là 1 biến thể của thuật toán Boyer-Moore
* Không yêu cầu pha tiền xử lý như thuật toán Boyer-Moore
* Cần không gian nhớ phụ như vớithuật toán Boyer-Moore;
* Pha tiền xử lý có độ phức tạp O(m+σ);
* Pha tìm kiếm có độ phức tạp thuật toán là O(n);

## Mô tả

Thuật toán Turbo-Bm là 1 thuật toán được cải thiện từ thuật toán Boyer-Moore. Thuật toán này không cần pha tiền xử lý mà chỉ cần cung cấp không gian nhớ phụ . Nó bao gồm việc ghi nhớ các thành phần của các kí tự khớp với hậu tố của các mẫu trong lần thử cuối cùng ( và chỉ thực hiện nếu có sự dịch hậu tố tốt được thực hiện).

Kỹ thuật này có 2 lợi ích:

- Có thể nhảy qua thành phần

- Có thể thực hiện dịch chuyển nhanh

Một dịch chuyển nhanh có thể thực hiện ra nếu trong quá trình xử lý hiện tại của các hậu tố của mẫu phù hợp với các kí tự ngắn hơn các xử lý trước đó. Trong trường hợp này chúng ta hãy gọi u là yếu tố nhớ và v là các hậu tố xuất hiện trong các xử lý hiện tại như vậy uzv là một hậu tố của x. Hãy để a và b là các phần tử không phù hợp trong các xử lý hiện tại trong các mẫu và các kĩ tự tương ứng. Sau đó, av là một hậu tố của x. Hai kí tự a,b xảy ra tại khoảng cách p trong chuỗi kí tự, và các hậu tố của x có chiều dài |uzv| có một đọ dài thời gian là p = |ZV| kể từ u là một biên giới của uzv, do đó nó không thể chồng lên nhau cả hai lần xuất hiện của hai nhân vật a và b khác nhau, tại khoảng cách p trong chuỗi kí tự. Sự dịch chuyển nhỏ nhất có thể có chiều dài |u| - |v|, mà chúng ta gọi là dịch chuyển nhanh (turbo-shift)  
Tuy nhiên trong trường hợp nơi | v | <| u | nếu chiều dài của sự thay đổi phần tử-tồi lớn hơn độ dài của sự dịch chuyển của hậu tố - tốt và độ dài của dịch chuyển nhanh sau đó chiều dài của sự chuyển đổi thực tế phải lớn hơn hoặc bằng |u| + 1.

Thật vậy(xemhình15.2), trong trường hợp nàyhai phần tửcvàdlà khác nhauvì chúng tagiả địnhrằngsự thay đổitrước đó là mộtsự dịch chuyển hậu tố tốt. Sau đó,một sự thay đổilớn hơndịch chuyển nhanh nhưng nhỏ hơn|u|+1sẽsắp xếpcvàd, vớimộtkí tự tương tựtrongv. Vì vậy, nếutrường hợp nàychiều dài củasự thay đổithực tế phảicó ít nhấtbằng |u|+1.  
Giai đoạn tiền xử lý có thể được thực hiện trong O(m+ σ) thời gian. Giai đoạn tìm kiếm làO(n). Các số so sánh phần tử trong đoạn mã được thực hiện bởi các thuật toánTurbo-BM được giới hạn bởi 2n.

## Mã hóa

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

#include <iostream>

using namespace std;

#define ASIZE 255

#define XSIZE 9

int bmBc[ASIZE]; //Mảng bmBc[ ] là mảng chứa vị trí xuất hiện cuối cùng của các ký tự trong xâu x

int bmGs[XSIZE]; //Mảng chứa vị trí có hậu tố trùng nhau hoặc có phần chung giữa 2 xâu

int suff[XSIZE]; //Mảng chứa giá trị xâu hậu tố dài nhất

void Init()

{

for(int i = 0; i < XSIZE; i++)

{

bmGs[i] = 999;

suff[i] = 999;

}

}

void preBmBc(char \*x, int m/\*, int bmBc[]\*/) {

int i;

for (i = 0; i < ASIZE; ++i)

bmBc[i] = m;

for (i = 0; i < m - 1; ++i) // mang bmBC chua vi tri xuat hien cuoi cung cua cac ky tu trong xau X, khong duyet Ky tu ben phai nhat (i < m-1)

bmBc[x[i]] = m - i - 1; // duyet xau tu trai qua phai, vi tri ky tu tinh' tu` phai qua trai. ( AGCG => A = 3; G = 2; C = 1)

}

void suffixes(char \*x, int m/\*, int \*suff\*/) { //xac dinh hau to chung dai nhat giua xau x[0 ... x] va x[0 ... m-1]

int f, g, i;

suff[m - 1] = m;

g = m - 1;

for (i = m - 2; i >= 0; --i) { //duyet tu cuoi xau ve dau xau

if (i > g && suff[i + m - 1 - f] < i - g)

suff[i] = suff[i + m - 1 - f];

else {

if (i < g)

g = i;

f = i;

while (g >= 0 && x[g] == x[g + m - 1 - f]) //neu trung nhau thi dua do dai trung vao

--g;

suff[i] = f - g;

}

}

}

void preBmGs(char \*x, int m/\*, int bmGs[]\*/) {

int i, j/\*, suff[XSIZE]\*/;

suffixes(x, m/\*, suff\*/);

for (i = 0; i < m; ++i)

bmGs[i] = m;

j = 0;

for (i = m - 1; i >= 0; --i)

if (suff[i] == i + 1)

for (; j < m - 1 - i; ++j)

if (bmGs[j] == m)

bmGs[j] = m - 1 - i;

for (i = 0; i <= m - 2; ++i)

bmGs[m - 1 - suff[i]] = m - 1 - i;

}

void TBM(char \*x, int m, char \*y, int n) {

int bcShift, i, j, shift, u, v, turboShift,

bmGs[XSIZE], bmBc[ASIZE];

/\* Preprocessing \*/

preBmGs(x, m/\*, bmGs\*/);

preBmBc(x, m/\*, bmBc\*/);

/\* Searching \*/

j = u = 0;

shift = m;

while (j <= n - m) {

i = m - 1;

while (i >= 0 && x[i] == y[i + j]) {

--i;

if (u != 0 && i == m - 1 - shift)

i -= u;

}

if (i < 0) {

cout<<(j);

shift = bmGs[0];

u = m - shift;

}

else {

v = m - 1 - i;

turboShift = u - v;

bcShift = bmBc[y[i + j]] - m + 1 + i;

shift = max(turboShift, bcShift);

shift = max(shift, bmGs[i]);

if (shift == bmGs[i])

u = min(m - shift, v);

else {

if (turboShift < bcShift)

shift = MAX(shift, u + 1);

u = 0;

}

}

j += shift;

}

}

void main()

{

char \*y = "GCATCGCAGAGAGTATACAGTACG"; //"ABABDABACDABABCABAB";

int n = strlen(y);

char \*x = "GCAGAGAG"; //"ABABCABAB";

int m = strlen(x);

TBM (x, m, y, n);

}

## Kiểm nghiệm

1. Pha tiền xử lý

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| c | A | C | G | T |
| bmBC[c] | 1 | 6 | 2 | 8 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| x[i] | G | C | A | G | A | G | A | G |
| suff[i] | 1 | 0 | 0 | 2 | 0 | 4 | 0 | 8 |
| bmGs[i] | 7 | 7 | 7 | 2 | 7 | 4 | 7 | 1 |

1. Pha tìm kiếm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | 1 |  | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

1. Shift by: 1 (*bmGs*[7]=*bmBc*[A]-8+8)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | 3 | 2 | 1 |  | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |

1. Shift by: 4 (*bmGs*[5]=*bmBc*[C]-8+6)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 6 | 5 | - | - | 4 | 3 | 2 | 1 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

1. Shift by: 7 (*bmGs*[0])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | 3 | 2 | 1 |  | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

1. Shift by: 4 (*bmGs*[5]=*bmBc*[C]-8+6)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | | 2 | 1 |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

1. Shift by: 7 (*bmGs*[6])

# Thuật toán Quick Search

## Những đặc điểm chính

* Phiên bản đơn giản của Boyer-Moore
* Chỉ sử dụng Bad-Character shift
* Dễ thực thi
* Giai đoạn tiền xử lý có độ phức tạp thời gian và không gian O(m+σ)
* Giai đoạn tìm kiếm có độ phức tạp thời gian O(mn)
* Rất nhanh trong thực thi với mẫu ngắn và bảng chữ cái lớn

## Mô tả

N/A

## Mã hóa

void preQsBc(char \*x, int m, int qsBc[]) {

int i;

for (i = 0; i < ASIZE; ++i)

qsBc[i] = m + 1;

for (i = 0; i < m; ++i)

qsBc[x[i]] = m - i;

}

void QS(char \*x, int m, char \*y, int n) {

int j, qsBc[ASIZE];

/\* Preprocessing \*/

preQsBc(x, m, qsBc);

/\* Searching \*/

j = 0;

while (j <= n - m) {

if (memcmp(x, y + j, m) == 0)

cout<<(j);

j += qsBc[y[j + m]]; /\* shift \*/

}

}

## Kiểm nghiệm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | A | C | G | T |
| qsBC[a] | 2 | 7 | 1 | 9 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
| 1 | 2 | 3 | 4 |  | | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 1 (*qsBc*[G])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | 1 |  | | | | | | | | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |

Shift by: 2 (*qsBc*[A])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | 1 |  | | | | | | | | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

Shift by: 2 (*qsBc*[A])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 9 (*qsBc*[T])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

Shift by: 7 (*qsBc*[C])

# Thuật toán Zhu-Takaoka:

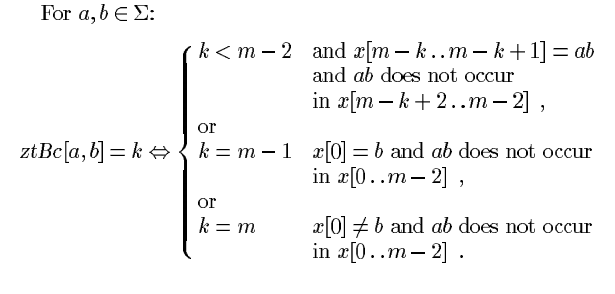
## Đặc điểm:

* Biến thể của thuật toán Boyer - moore
* Sử dụng 2 ký tự văn bản liên tiếp để tính toán sự thay đổi ký tự xấu
* Giai đoạn tiền xử lý trong 0(m + a2)thời gian và phức tạp không gian
* Tìm kiếm trong giai đoạn 0 (m x n) thời gian phức tạp

## Mô tả:

Zhu and takaoka thiết kế một thuật toán thực hiện sự thay đổi bằng cách xem xét các thay đổi xấu ký tự ( xem trang 14) cho 2 ký tự văn bản liên tiếp. Trong giai đoạn tìm kiếm so sánh được thực hiện từ phải sang trái và khi cửa sổ đặt trên nhân tố văn bản Y[j1] và một ghép đôi không xứng xảy ra giữa x[m — k] và y[j + m — k] khi đó x[m — k + I . . m — 1] = y[j + m — k + I . . j + m — 1] sự thay đổi là thực hiện với thay đổi ký tự xấu cho ký tự văn bản y[j + m — 2] và y[j + m — 1]. Bảng thay đổi good- sufEx cũng được dùng để tính toán sự thay đổi

Giai đoạn tiền xử lý của thuật toán bao gồm trong tính toán cho mỗi cặp ký tự (a,b) với a,b € ∑ xảy ra ở đầu bên phải của ab trong x[0..m-2]



Nó cũng bao gồm việc tính toán bang bmGs ( xem trang 14). Giai đoạn tiền xử lý là trong *0(m +**2) thời gian và phức tạp không gian*

Giai đoạn tìm kiếm có một trường hợp xấu bậc 2

## Mã hóa

void preZtBc(char \*x, int m, int ztBc[ASIZE][ASIZE]) {

int i, j;

for (i = 0; i < ASIZE; ++i)

for (j = 0; j < ASIZE; ++j)

ztBc[i][j] = m;

for (i = 0; i < ASIZE; ++i)

ztBc[i][x[0]] = m - 1;

for (i = 1; i < m - 1; ++i)

ztBc[x[i - 1]][x[i]] = m - 1 - i;

}

void ZT(char \*x, int m, char \*y, int n) {

int i, j, ztBc[ASIZE][ASIZE], bmGs[XSIZE];

/\* Preprocessing \*/

preZtBc(x, m, ztBc);

preBmGs(x, m, bmGs);

/\* Searching \*/

j = 0;

while (j <= n - m) {

i = m - 1;

while (i < m && x[i] == y[i + j])

--i;

if (i < 0) {

cout<<(j);

j += bmGs[0];

}

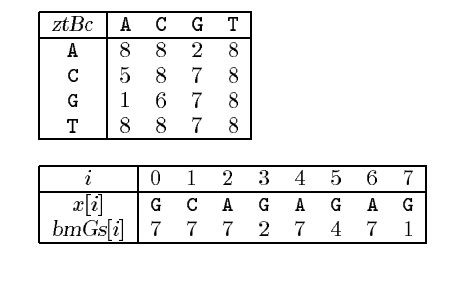
else

j += max(bmGs[i], ztBc[y[j + m - 2]][y[j + m - 1]]);

}

}

## Kiểm nghiệm



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | 1 |  | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 5 (*ztBc*[C][A])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 7 (*bmGs*[0])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | 3 | 2 | 1 |  | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

Shift by: 7 (*bmGs*[6])

# Thuật toán Berry – Ravindran

## Phát biểu thuật toán

* Sự pha trộn của thuật toán Quick Search và thuật toán Zhu-Takaoka
* Giai đoạn tiền xử lý độ phức tạp *O(m + 2)*
* Giai đoạn tìm kiếm độ phức tạp *O(m x n)*

## Mô tả

N/A

## Mã hóa

void preBrBc(char \*x, int m, int brBc[ASIZE][ASIZE]) {

int a, b, i;

for (a = 0; a < ASIZE; ++a)

for (b = 0; b < ASIZE; ++b)

brBc[a][b] = m + 2;

for (a = 0; a < ASIZE; ++a)

brBc[a][x[0]] = m + 1;

for (i = 0; i < m - 1; ++i)

brBc[x[i]][x[i + 1]] = m - i;

for (a = 0; a < ASIZE; ++a)

brBc[x[m - 1]][a] = 1;

}

void BR(char \*x, int m, char \*y, int n) {

int j, brBc[ASIZE][ASIZE];

/\* Preprocessing \*/

preBrBc(x, m, brBc);

/\* Searching \*/

y[n + 1] = '\0';

j = 0;

while (j <= n - m) {

if (memcmp(x, y + j, m) == 0)

cout<<(j);

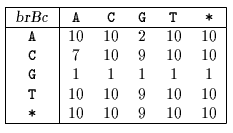
j += brBc[y[j + m]][y[j + m + 1]];

}

}

## Kiểm nghiệm

Tiền xử lý



Tìm kiếm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
| 1 | 2 | 3 | 4 |  | | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 1 (*brBc*[G][A])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | 1 |  | | | | | | | | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |

Shift by: 2 (*brBc*[A][G])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | 1 |  | | | | | | | | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

Shift by: 2 (*brBc*[A][G])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 10 (*brBc*[T][A])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  |

Shift by: 7 (*brBc*[G][0])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 1 |  | | | | | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

Shift by: 10 (*brBc*[0][0])

# Thuật toán Colussi

## Đặc điểm

* Sàng lọc lại thuật toán Knutt-Morris-Pratt;
* Phân vùng tập các vị trí mẫu thành 2 tập con rời nhau; các vị trí trong tập đầu tiên được từ trái qua phải và khi không có sự phù hợp xảy ra các vị trí trong tập con thứ 2 sẽ được quét từ phải qua trái;
* Pha tiền xử lý có độ phức tạp không gian và thời gian là O(m);
* Pha tìm kiếm có độ phức tạp thời gian là O(n);
* Trong trường hợp xấu nhất phải thực hiện so sánh ký tự văn bản

## Mô tả

Việc thiết kế thuật toán Colussi tuân theo một phân tích có tính chặt chẽ của thuật toán Knutt-Morris-Pratt

Tập các vị trí mẫu được phân chia thành 2 tập con rời nhau. Sau đó, mỗi mẫu thử bao gồm 2 pha:

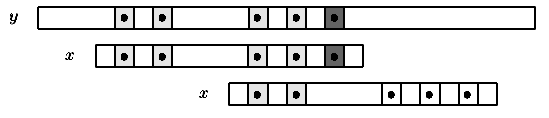
* Trong pha đầu tiên, các so sánh được thực hiện từ trái qua phải với các ký tự văn bản phù hợp với vị trí mẫu mà giá trị của hàm kmpNext hoàn toàn lớn hơn -1. Những vị trí đó được gọi là ***noholes;***
* Pha thứ 2 bao gồm việc so sánh các vị trí còn lại (được gọi là ***holes***) từ phải qua trái.

Chiến lược này có 2 ưu điểm:

* Khi một không phù hợp xảy ra trong pha đầu tiên, sau khi dịch chuyển thích hợp không cần thiết phải so sánh ký tự văn bản phù hợp với noholes được so sánh trong suốt mẫu thử trước;
* Khi một không phù hợp xảy ra trong pha thứ 2 điều đó có nghĩa là một hậu tố của mẫu thử phù hợp với một nhân tố của văn bản, sau khi dịch chuyển tương ứng một tiền tố của mẫu thử cũng sẽ vẫn phù hợp với một nhân tố của văn bản, do đó không cần thiết phải so sánh lại với nhân tố đó nữa.

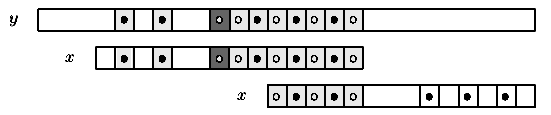
**Định nghĩa thuật toán:**

* For 0 leq *i* leq *m*-1:
* Khi *kmin* neq 0 một chu kỳ kết thúc tại vị trí *i* trong *x*.
* For 0 < *i* < *m*:
* Lấy *nd*+1 là số lượng của noholes trong *x*.
* Bảng *h* chứa *nd+1* noholes đầu tiên theo thứ tự tăng dần và tiếp đó là *m-nd-1* holes theo thứ tự giảm dần:
  + for *0 leq i leq nd*, *h[i]* là một nohole và *h[i] < h[i+1]* với *0 leq i<nd*;
  + for *nd < i < m*, *h[i]* là một hole và *h[i] > h[i+1]* với *nd < i < m-1*
* Nếu *i* là một hole thì *rmin[i]* là chu kỳ nhỏ nhất của *x* lớn hơn *i*
* Giá trị của *first[u]* là số nguyên nhỏ nhất *v* mà *u* leq *h*[*v*]
* Tiếp theo, giả sử rằng *x* phù hợp với *y[j .. j+m-1]*. Nếu *x[h[k]]=y[j+h[k]]* với *0 leq k < r < nd* và *x[h[r]] neq y[j+h[r]]*. Lấy *j’ = j+kmin[h[r]]*. Tiếp đó không có sự xuất hiện của *x* bắt đầu trong *y[j .. j’]* và *x* có thể được dịch *kmin*[h[r]] vị trí sang phải .
* Ngoài ra *x[h[k]]=y[j’+h[k]]* với *0 leq k < first[h[r]-kmin[h[r]]]* có nghĩa rằng việc so sánh có thể được tiếp tục với *x[h[first[h[r]-kmin[h[r]]]]]* và *y[j’+h[first[h[r]-kmin[h[r]]]]]*.



*Hình 1: Không phù hợp với một nohole. Noholes là các vòng tròn màu đen và được so sánh từ trái qua phải*

* Nếu *x[h[k]]=y[j+h[k]]* với *0 leq k < r* và *x[h[r]] neq y[j+h[r]]* với *nd leq r < m*. Lấy *j’=j+rmin[h[r]]*. Tiếp đó, không có sự xuất hiện nào của *x* bắt đầu trong *y[j .. j’]* và *x* có thể được dịch *kmin[h[r]]* vị trí sang phải.
* Ngoài ra *x[0 .. m-1-rmin[h[r]]]=y[j’ .. j+m-1]* có nghĩa rằng việc so sánh có thể được tiếp tục với *x[h[first[m-1-rmin[h[r]]]]]* và *y[j’+h[first[m-1-rmin[h[r]]]]]*.



*Hình 2: Không phù hợp với một hole. Noholes là các vòng tròn màu đen và được so sánh từ trái qua phải, trong khi holes là các vòng tròn màu trắng và được so sánh từ phải qua trái*

* Để tính toán giá trị *kmin*, một bảng *hmax* được sử dụng và được định nghĩa như sau: *hmax[k]* thỏa mãn *x[k .. hmax[k]-1]=x[0 .. hmax[k]-k-1]* và *x[hmax[k]] neq x[hmax[k]-k]*.
* Giá trị của *nhd0[i]* là số lượng các noholes chắc chắn nhỏ hơn *i*
* Chúng ta định nghĩa 2 hàm *shift* và *next* như sau:
  + *shift[i]=kmin[h[i]] và next[i]=nhd0[h[i]-kmin[h[i]]] với i < nd;*
  + *shift[i]=rmin[h[i]] và next[i]=nhd0[m-rmin[h[i]]] với nd leq i < m;*
  + *shift[m]=rmin[0] và next[m]=nhd0[m-rmin[h[m-1]]].*
* Do đó, trong suốt một lần thử, khi cửa sổ được đặt ở vị trí trên nhân tố văn bản *y[j .. j+m-1]*, khi một không phù hợp xuất hiện giữa *x[h[r]] và y[j+h[r]]* cửa số phải được dịch đi *shift[r]* và những so sánh có thể được tiếp tục với mẫu ởi vị trí *h[next[r]]*.

Pha tiền xử lý có thể hoàn thành trong một O(m) không gian và thời gian. Pha tìm kiếm có thể hoàn thành trong O(n) độ phức tạp thời gian và hơn nữa tối đa 3/2n lần việc so sánh ký tự văn bản được thực hiện trong suốt pha tìm kiếm.

## Mã hóa

int preColussi(char \*x, int m, int h[], int next[],

int shift[]) {

int i, k, nd, q, r, s;

int hmax[XSIZE], kmin[XSIZE], nhd0[XSIZE], rmin[XSIZE];

/\* Computation of hmax \*/

i = k = 1;

do {

while (x[i] == x[i - k])

i++;

hmax[k] = i;

q = k + 1;

while (hmax[q - k] + k < i) {

hmax[q] = hmax[q - k] + k;

q++;

}

k = q;

if (k == i + 1)

i = k;

} while (k <= m);

/\* Computation of kmin \*/

memset(kmin, 0, m\*sizeof(int));

for (i = m; i >= 1; --i)

if (hmax[i] < m)

kmin[hmax[i]] = i;

/\* Computation of rmin \*/

for (i = m - 1; i >= 0; --i) {

if (hmax[i + 1] == m)

r = i + 1;

if (kmin[i] == 0)

rmin[i] = r;

else

rmin[i] = 0;

}

/\* Computation of h \*/

s = -1;

r = m;

for (i = 0; i < m; ++i)

if (kmin[i] == 0)

h[--r] = i;

else

h[++s] = i;

nd = s;

/\* Computation of shift \*/

for (i = 0; i <= nd; ++i)

shift[i] = kmin[h[i]];

for (i = nd + 1; i < m; ++i)

shift[i] = rmin[h[i]];

shift[m] = rmin[0];

/\* Computation of nhd0 \*/

s = 0;

for (i = 0; i < m; ++i) {

nhd0[i] = s;

if (kmin[i] > 0)

++s;

}

/\* Computation of next \*/

for (i = 0; i <= nd; ++i)

next[i] = nhd0[h[i] - kmin[h[i]]];

for (i = nd + 1; i < m; ++i)

next[i] = nhd0[m - rmin[h[i]]];

next[m] = nhd0[m - rmin[h[m - 1]]];

return(nd);

}

void COLUSSI(char \*x, int m, char \*y, int n) {

int i, j, last, nd,

h[XSIZE], next[XSIZE], shift[XSIZE];

/\* Processing \*/

nd = preColussi(x, m, h, next, shift);

/\* Searching \*/

i = j = 0;

last = -1;

while (j <= n - m) {

while (i < m && last < j + h[i] &&

x[h[i]] == y[j + h[i]])

i++;

if (i >= m || last >= j + h[i]) {

OUTPUT(j);

i = m;

}

if (i > nd)

last = j + m - 1;

j += shift[i];

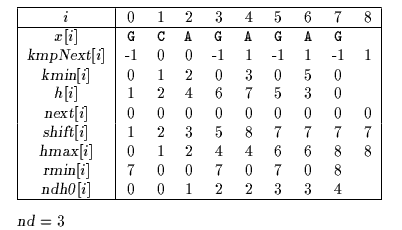
i = next[i];

}

}

## Kiểm nghiệm

Tiền xử lý



Tìm kiếm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | 1 | 2 |  | 3 |  | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 3 (*shift*[2])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | 1 | 2 |  | | | | | | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

Shift by: 2 (*shift*[1])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 8 | 1 | 2 | 7 | 3 | 6 | 4 | 5 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 7 (*shift*[8])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | 1 |  | | | | | | | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

Shift by: 1 (*shift*[0])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | |

Shift by: 1 (*shift*[0])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

Shift by: 1 (*shift*[0])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 1 |  | | | | | | |
|  | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  |

Shift by: 1 (*shift*[0])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eighth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | 1 | 2 |  | 3 |  | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

Shift by: 3 (*shift*[2])

# Thuật toán Galil-Giancarlo

## Đặc điểm

* Sàng lọc thuật toán Colussi;
* Pha tiền xử lý có độ phức tạp không gian và thời gian là O(m);
* Pha tìm kiếm có độ phức tạp thời gian là O(n);
* Trong trường hợp xấu nhất phải thực hiện so sánh ký tự văn bản

## Mô tả

Các thuật toán Galil-Giancarlo là một biến thể của thuật toán Colussi (xem chương 9). Sự thay đổi can thiệp vào pha tìm kiếm. Phương pháp này được áp dụng khi x không phải là một lũy thừa của một ký tự đơn. Như vậy x ≠ *cm* với c € Σ. Lấy l là chỉ số cuối cùng trong patern sao cho 0 ≤ i ≤ l , x [0] = [i] x và x [0] ≠ x [l + 1]. Giả sử trong lần thử trước đó tất cả các noholes đã được so khớp và một hậu tố của pattern đã được so khớp nghĩa là sau sự thay đổi tương ứng một tiền tố của pattern sẽ vẫn khớp với một phần của văn bản. Do đó các ô được đặt ở vị trí trên các nhân tố văn bản *y*[j ... j + m -1] và phân ra *y*[j ... cuối cùng] khớp với x [0 ... cuối cùng - j]. Sau đó lần thử tiếp theo thuật toán sẽ quét các ký tự văn bản bắt đầu với y [last + 1] cho đến khi hoặc kết thúc của văn bản là đạt hoặc một ký tự x[0] ≠ y [j + k] được tìm thấy. Trong trường hợp sau này hai subcases có thể phát sinh:

x [l + 1] ≠ y [j + k] hoặc quá ít x [0] đã được tìm thấy (k ≤ l) sau đó các ô được chuyển và định vị trên các nhân tố văn bản y [k + 1 ... k + m), quá trình quét của văn bản được khôi phục lại (như trong thuật toán Colussi) với nohole đầu tiên và tiền tố ghi nhớ của pattern là những từ rỗng.

x [l + 1] = y [j + k] và đầy đủ của x [0] đã được tìm thấy (k>l) sau đó các ô được chuyển và định vị trên các nhân tố văn bản y [k - l -1 ... k - l + m - 2], quá trình quét của văn bản được khôi phục lại (như trong thuật toán Colussi) với nohole thứ hai (x [l + 1] là đầu tiên) và tiền tố ghi nhớ của pattern là x [0 ... l + 1].

Pha tiền xử lý là chính xác giống như trong thuật toán Colussi (chương 9) và có thể được thực hiện trong O(m) không gian và thời gian. Pha tìm kiếm sau đó có thể thực hiện độ phức tạp về thời gian O(n) và hơn nữa tối đa là 4/3n so sánh ký tự văn bản được thực hiện trong giai đoạn tìm kiếm.

## Mã hóa

void GG(char \*x, int m, char \*y, int n) {

int i, j, k, ell, last, nd;

int h[XSIZE], next[XSIZE], shift[XSIZE];

char heavy;

for (ell = 0; x[ell] == x[ell + 1]; ell++);

if (ell == m - 1)

/\* Searching for a power of a single character \*/

for (j = ell = 0; j < n; ++j)

if (x[0] == y[j]) {

++ell;

if (ell >= m)

OUTPUT(j - m + 1);

}

else

ell = 0;

else {

/\* Preprocessing \*/

nd = preCOLUSSI(x, m, h, next, shift);

/\* Searching \*/

i = j = heavy = 0;

last = -1;

while (j <= n - m) {

if (heavy && i == 0) {

k = last - j + 1;

while (x[0] == y[j + k])

k++;

if (k <= ell || x[ell + 1] != y[j + k]) {

i = 0;

j += (k + 1);

last = j - 1;

}

else {

i = 1;

last = j + k;

j = last - (ell + 1);

}

heavy = 0;

}

else {

while (i < m && last < j + h[i] &&

x[h[i]] == y[j + h[i]])

++i;

if (i >= m || last >= j + h[i]) {

OUTPUT(j);

i = m;

}

if (i > nd)

last = j + m - 1;

j += shift[i];

i = next[i];

}

heavy = (j > last ? 0 : 1);

}

}

}

## Kiểm nghiệm

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **i** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| x[i] | G | C | A | G | A | G | A | G |  |
| kmpNext[i] | -1 | 0 | 0 | -1 | 1 | -1 | 1 | -1 | 1 |
| kmin[i] | 0 | 1 | 2 | 0 | 3 | 0 | 5 | 0 |  |
| h[i] | 1 | 2 | 4 | 6 | 7 | 5 | 3 | 0 |  |
| next[i] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| shift[i] | 1 | 2 | 3 | 5 | 8 | 7 | 7 | 7 | 7 |
| hmax[i] | 0 | 1 | 2 | 4 | 4 | 6 | 6 | 8 | 8 |
| rmin[i] | 7 | 0 | 0 | 7 | 0 | 7 | 0 | 8 |  |
| ndh0[i] | 0 | 0 | 1 | 2 | 2 | 3 | 3 | 4 |  |

Pha tìm kiếm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | 1 | 2 |  | 3 |  | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 3 (*shift*[2])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | 1 | 2 |  | | | | | | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

Shift by: 2 (*shift*[1])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 8 | 1 | 2 | 7 | 3 | 6 | 4 | 5 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 7 (*shift*[8])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | 1 |  | | | | | | | | | |
|  | | | | | | | | | | | | - | G |  | | | | | | | | | |

Shift by: 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

Shift by: 1 (*shift*[0])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 1 |  | | | | | | |
|  | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  |

Shift by: 1 (*shift*[0])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | 1 | 2 |  | 3 |  | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

Shift by: (*shift*[2])

# Thuật toán Apostolico - Crochemore

## Đặc điểm

* Thực hiện từ trái sang phải
* Pha tiền xử lý có độ phức tạp về không gian và thời gian là
* Pha tìm kiếm có độ phức tạp về thời gian là
* Trong trường hợp xấu nhất, thuật toán thực hiện so sánh ký tự

## Mô tả

**Input:**

* Xâu mẫu X = (, ,…, ), độ dài m.
* Văn bản Y = (, ,…, ), độ dài n

**Output:**

* Đưa ra mọi vị trí xuất hiện của X trong Y.

**Formats:** Apostolico-Crochemore(X,m,Y,n);

**Actions:**

**Bước 1** (Tiền xử lý):

* Xây dựng bảng dịch để tính toán bước dịch: **preKMP(x, m, kmpNext)**. //sử dụng thuật toán xây dựng bảng dịch của phương pháp Knuth-Morris-Pratt.
* Tìm vị trí ký tự khác nhau đầu tiên trong xâu mẫu X, ký hiệu là ***l***.
  + ***l***= 0 nếu x chỉ chứa 1 loại ký tự, vd: aaaaaa.
  + ***l***= vị trí của ký tự đầu tiền của X khác với ký tự x0.

**Bước 2** (Tìm kiếm):

* Việc so sánh được tính toán trên mô hình các vị trí theo thứ tự sau:

***l*, *l* + 1, …, *m* – 2, *m* – 1, 0, 1, …, *l* – 1.**

* Xét bộ ba ***(i, j, k)*** thỏa mãn:
  + Cửa sổ chạy được xác định bởi factor **[, , …, ]**
  + **0 *kl*và [, , …, ] = [, , …, ]**
  + ***l im*và [, , …, ] = [, , …, ]**
* Cách tính toán bộ ba ***(i, j, k)***tiêp theo:

If ( ***i* = l** ) {

If(  **= )**{

Bộ 3 tiếp theo là **(*i* + 1,*j*, *k*)**

}else{

Bộ 3 tiếp theo là **(*l*, *j+1*, max{0, *k – 1}*)**

}

}else if( ***l < i < m*** ) {

If(  **= )**{

Bộ 3 tiếp theo là **(*i* + 1,*j*, *k*)**

}else{

If( **kmpNext[i]*l*)**{

Bộ 3 tiếp theo là **(*l*,*i* + *j – kmpNext[i]*, max{0, *kmpNext[i]*, *}*))**

}else{

Bộ 3 tiếp theo là **(*kmpNext[i]*, *i* + *j – kmpNext[i]*, *l*)**

}

}

}else if( i = m ){

If ( ***kl***&& **=** ){

Bộ 3 tiếp theo là **(*i*,*j*, *k +* 1)**

}else{

If ( k = l ){

<tìm thấy mẫu ở vị trí j – i >

}

Tính bộ 3 theo trường hợp ***l < i < m.***

}

}

## Mã hóa

//The function preKmp is given chapter Knuth, Morris and Pratt algorithm.

void AXAMAC(char \*x, int m, char \*y, int n) {

int i, j, k, ell, kmpNext[XSIZE];

/\* Preprocessing \*/

preKmp(x, m, kmpNext);

for (ell = 1; x[ell - 1] == x[ell]; ell++);

if (ell == m)

ell = 0;

/\* Searching \*/

i = ell;

j = k = 0;

while (j <= n - m) {

while (i < m && x[i] == y[i + j])

++i;

if (i >= m) {

while (k < ell && x[k] == y[j + k])

++k;

if (k >= ell)

OUTPUT(j);

}

j += (i - kmpNext[i]);

if (i == ell)

k = MAX(0, k - 1);

else

if (kmpNext[i] <= ell) {

k = MAX(0, kmpNext[i]);

i = ell;

}

else {

k = ell;

i = kmpNext[i];

}

}

}

## Kiểm nghiệm



l = 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | 1 | 2 | 3 |  | | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 4 (*i*-*kmpNext*[*i*]=3- -1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 1 |  | | | | | | | | | | | | | | | | | |
|  | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | |

Shift by: 1 (*i*-*kmpNext*[*i*]=1-0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 8 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 7 (*i*-*kmpNext*[*i*]=8-1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | 1 |  | | | | | | | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

Shift by: 1 (*i*-*kmpNext*[*i*]=1-0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | |

Shift by: 1 (*i*-*kmpNext*[*i*]=1-0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

Shift by: 1 (*i*-*kmpNext*[*i*]=1-0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 1 |  | | | | | | |
|  | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  |

Shift by: 1 (*i*-*kmpNext*[*i*]=1-0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eighth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | 1 | 2 | 3 | 4 |  | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

Shift by: 3 (*i*-*kmpNext*[*i*]=4-1)

# Thuật toán Not So Naive

## Đặc điểm

• Thực hiện từ trái sang phải.

• Pha tiền xử lý có độ phức tạp hằng số.

• Độ phức tạp về không gian là hằng số

• Pha tìm kiếm có độ phức tạp thuật toán là O(n.m);

## Mô tả

**Input :**

• Xâu mẫu X =(x0, x1,..,xm), độ dài m.

• Văn bàn Y=(y1, y2,..,yn) độ dài n.

**Output:**

• Đưa ra mọi vị trí xuất hiện của X trong Y.

**Formats:** k = NSN(X, m, Y, n);

**Actions:**

Bước 1 (Tiền xử lý): Xác định số bước dịch chuyển SubY tren Y

- Gọi:

+ SubY: chuỗi con gồm m phần tử (y[j.. j+m-1]) của Y

+ k: số bước dịch khi x[1] != SubY[1]

+ l: số bước dịch khi x[1] = SubY[1]

- Xác định số dịch chuyển chuỗi con SubY khi chưa thỏa mãn:

+ Nếu (x[0] = x[1] && x[1] != y[j+1]) --> x[0] != y[j+1] --> SubY dịch chuyển 2 đơn vị j = j+2 --> k =2

+ Nếu (x[0] != x[1] && x[1] = y[j+1]) --> x[0] != y[j+1] --> SubY dịch chuyển 2 đơn vị j = j+2 --> l =2

+ Còn lại SubY dịch chuyển 1 đơn vị j = j+1

/\* Preprocessing \*/

if (x[0] == x[1]) {

k = 2; // dịch chuyển 2 nếu x[1] != suby[1]

l = 1; // ngược lại dịch chuyển 1

}

else {

k = 1; // dịch chuyển 1 nếu x[1] != suby[1]

l = 2; // ngược lại dịch chuyển 2

}

Bước 2 (Tìm kiếm):

• Ký tự của SubY được so sánh với chuỗi mẫu X theo thứ tự 1, 2, ... , m-2, m-1, 0

• Nếu SubY = chuỗi mẫu thì in ra j (vị trí bắt đầu của SubY trên Y)

• Ngược lại di chuyển SubY trên Y

**EndActions.**

## Mã hóa

void NSN(char \*x, int m, char \*y, int n) {

int j, k, ell;

/\* Preprocessing \*/

if (x[0] == x[1]) {

k = 2;

ell = 1;

}

else {

k = 1;

ell = 2;

}

/\* Searching \*/

j = 0;

while (j <= n - m)

if (x[1] != y[j + 1])

j += k;

else {

if (memcmp(x + 2, y + j + 2, m - 2) == 0 &&

x[0] == y[j])

OUTPUT(j);

j += ell;

}

}

## Kiểm nghiệm

Pha tiền xử lý

*k=1* and ell=2

Pha tìm kiếm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | 1 | 2 | 3 |  | | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 2 (*l*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | 1 |  | | | | | | | | | | | | | | | | | | | |
|  | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | |

Shift by: 1 (*k*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | 1 | 2 |  | | | | | | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

Shift by: 2 (*l*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 8 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 2 (*l*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | 1 |  | | | | | | | | | | | | | | |
|  | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | |

Shift by: 1 (*k*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | 1 |  | | | | | | | | | | | | | |
|  | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | | |

Shift by: 1 (*k*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | 1 |  | | | | | | | | | | | | |
|  | | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | | |

Shift by: 1 (*k*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eighth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | 1 |  | | | | | | | | | | | |
|  | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | | | |

Shift by: 1 (*k*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ninth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | 1 |  | | | | | | | | | | |
|  | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | | |

Shift by: 1 (*k*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | 1 |  | | | | | | | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

Shift by: 1 (*k*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eleventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | |

Shift by: 1 (*k*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Twelfth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

Shift by: 1 (*k*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Thirteenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 1 |  | | | | | | |
|  | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  |

Shift by: 1 (*k*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourteenth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | 1 | 2 | 3 | 4 |  | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

Shift by: 2 (*l*)

# Thuật toán Colussi Reverse

## Đặc điểm

* Được tinh chỉnh lại từ thuật toán Boyer-Moore
* Phân chia tập các vị trí mẫu thành hai tập nhỏ hơn tách rời nhau
* Pha tiền xử lý có độ phức tạp O(*m2*) về thời gian và O(*m* x *σ*) về không gian
* Pha tìm kiếm có độ phức tạp thuật toán là O(*n*)

## Mô tả

N/A

## Mã hóa

void preRc(char \*x, int m, int h[],

int rcBc[ASIZE][XSIZE], int rcGs[]) {

int a, i, j, k, q, r, s,

hmin[XSIZE], kmin[XSIZE], link[XSIZE],

locc[ASIZE], rmin[XSIZE];

/\* Computation of link and locc \*/

for (a = 0; a < ASIZE; ++a)

locc[a] = -1;

link[0] = -1;

for (i = 0; i < m - 1; ++i) {

link[i + 1] = locc[x[i]];

locc[x[i]] = i;

}

/\* Computation of rcBc \*/

for (a = 0; a < ASIZE; ++a)

for (s = 1; s <= m; ++s) {

i = locc[a];

j = link[m - s];

while (i - j != s && j >= 0)

if (i - j > s)

i = link[i + 1];

else

j = link[j + 1];

while (i - j > s)

i = link[i + 1];

rcBc[a][s] = m - i - 1;

}

/\* Computation of hmin \*/

k = 1;

i = m - 1;

while (k <= m) {

while (i - k >= 0 && x[i - k] == x[i])

--i;

hmin[k] = i;

q = k + 1;

while (hmin[q - k] - (q - k) > i) {

hmin[q] = hmin[q - k];

++q;

}

i += (q - k);

k = q;

if (i == m)

i = m - 1;

}

/\* Computation of kmin \*/

memset(kmin, 0, m \* sizeof(int));

for (k = m; k > 0; --k)

kmin[hmin[k]] = k;

/\* Computation of rmin \*/

for (i = m - 1; i >= 0; --i) {

if (hmin[i + 1] == i)

r = i + 1;

rmin[i] = r;

}

/\* Computation of rcGs \*/

i = 1;

for (k = 1; k <= m; ++k)

if (hmin[k] != m - 1 && kmin[hmin[k]] == k) {

h[i] = hmin[k];

rcGs[i++] = k;

}

i = m-1;

for (j = m - 2; j >= 0; --j)

if (kmin[j] == 0) {

h[i] = j;

rcGs[i--] = rmin[j];

}

rcGs[m] = rmin[0];

}

void RC(char \*x, int m, char \*y, int n) {

int i, j, s, rcBc[ASIZE][XSIZE], rcGs[XSIZE], h[XSIZE];

/\* Preprocessing \*/

preRc(x, m, h, rcBc, rcGs);

/\* Searching \*/

j = 0;

s = m;

while (j <= n - m) {

while (j <= n - m && x[m - 1] != y[j + m - 1]) {

s = rcBc[y[j + m - 1]][s];

j += s;

}

for (i = 1; i < m && x[h[i]] == y[j + h[i]]; ++i);

if (i >= m)

OUTPUT(j);

s = rcGs[i];

j += s;

}

}

## Kiểm nghiệm

Pha tiền xử lý

Tính locc[a]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *a* | A | C | G | T |
| *locc*[*a*] | 6 | 1 | 5 | -1 |

Tính rcBc thông qua locc

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *rcBc* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| A | 8 | 5 | 5 | 3 | 3 | 3 | 1 | 1 |
| C | 8 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| G | 2 | 2 | 2 | 4 | 4 | 2 | 2 | 2 |
| T | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |

Tính rcGs[i], h[i] thông qua hmin, kmin, rmin

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| *x*[*i*] | G | C | A | G | A | G | A | G |  |
| *link*[*i*] | -1 | -1 | -1 | -1 | 0 | 2 | 3 | 4 |  |
| *hmin*[*i*] | 0 | 7 | 3 | 7 | 5 | 5 | 7 | 6 | 7 |
| *kmin*[*i*] | 0 | 0 | 0 | 2 | 0 | 4 | 7 | 1 | 0 |
| *rmin*[*i*] | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 8 | 0 |
| *rcGs*[*i*] | 0 | 2 | 4 | 7 | 7 | 7 | 7 | 7 | 7 |
| *h*[*i*] |  | 3 | 5 | 6 | 0 | 1 | 2 | 4 |  |

Pha tìm kiếm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | 1 |  | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 1 (*rcBc*[A][8])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | 2 |  |  |  | 1 |  | | | | | | | | | | | | | | |
|  | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | |

Shift by: 2 (*rcGs*[2])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | 2 |  |  |  | 1 |  | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

Shift by: 2 (*rcGs*[2])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 5 | 6 | 7 | 2 | 8 | 3 | 4 | 1 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 7 (*rcGs*[9])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 2 |  |  |  | 1 |  | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

Shift by: 2 (*rcGs*[2])

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | | | | 1 |  | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

Shift by: 5 (*rcBc*[A][2])

# Thuật toán String Matching on Ordered

## Đặc điểm

* Không có giai đoạn tiền xử lý;
* Đòi hỏi phải có một bảng chữ cái đúng;
* Độ phức tạp trong giai đoạn tìm kiếm O (n);
* Thực hiện 6n + 5 so sánh đối tượng văn bản trong trường hợp xấu nhất.

## Mô tả

N/A

## Mã hóa

/\* Compute the next maximal suffix. \*/

void nextMaximalSuffix(char \*x, int m,

int \*i, int \*j, int \*k, int \*p) {

char a, b;

while (\*j + \*k < m) {

a = x[\*i + \*k];

b = x[\*j + \*k];

if (a == b)

if (\*k == \*p) {

(\*j) += \*p;

\*k = 1;

}

else

++(\*k);

else

if (a > b) {

(\*j) += \*k;

\*k = 1;

\*p = \*j - \*i;

}

else {

\*i = \*j;

++(\*j);

\*k = \*p = 1;

}

}

}

/\* String matching on ordered alphabets algorithm. \*/

void SMOA(char \*x, int m, char \*y, int n) {

int i, ip, j, jp, k, p;

/\* Searching \*/

ip = -1;

i = j = jp = 0;

k = p = 1;

while (j <= n - m) {

while (i + j < n && i < m && x[i] == y[i + j])

++i;

if (i == 0) {

++j;

ip = -1;

jp = 0;

k = p = 1;

}

else {

if (i >= m)

OUTPUT(j);

nextMaximalSuffix(y + j, i+1, &ip, &jp, &k, &p);

if (ip < 0 ||

(ip < p &&

memcmp(y + j, y + j + p, ip + 1) == 0)) {

j += p;

i -= p;

if (i < 0)

i = 0;

if (jp - ip > p)

jp -= p;

else {

ip = -1;

jp = 0;

k = p = 1;

}

}

else {

j += (MAX(ip + 1,

MIN(i - ip - 1, jp + 1)) + 1);

i = jp = 0;

ip = -1;

k = p = 1;

}

}

}

}

## Kiểm nghiệm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
| 1 | 2 | 3 | 4 |  | | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | 1 |  | | | | | | | | | | | | | | | | | | |
|  | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | |

Shift by: 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 9

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 |  | | | | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

Shift by: 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | 1 |  | | | | | | | |
|  | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  |

Shift by: 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 1 |  | | | | | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

Shift by: 1

# Thuật toán Two way

## Đặc điểm

* Tìm kiếm mẫu từ một vị trí cụ thể
* Yêu cầu có sẵn 1 bảng chữ cái alphabet
* Có pha tiền xử lý với độ phức tạp O(m)
* Tìm kiếm pha với thời gian O(n)
* Thực hiện so sánh 2n-m ký tự văn bản trong trường hợp xấu nhất

## Mô tả

**Input:**

* Xâu mẫu X=(x0, x1,…,xm), độ dài m
* Văn bản Y=(y0, y1,…,yn ), độ dài n

**Output:**

* Đưa ra mọi vị trí xuất hiện của X trong Y

**Formats:** k=TW (X,m,Y,n)

**Actions:**

* Bước 1 ( tiền xử lý ):

i= maxSuf (x, m, &p);

j=maxSufTilde (x, m, &q);

if (i>j){

ell=i;

per=p;

}

else{

ell=j;

per=q;

}

* Bước 2 (tìm kiếm):

## Mã hóa

/\* Computing of the maximal suffix for <= \*/

int maxSuf(char \*x, int m, int \*p) {

int ms, j, k;

char a, b;

ms = -1;

j = 0;

k = \*p = 1;

while (j + k < m) {

a = x[j + k];

b = x[ms + k];

if (a < b) {

j += k;

k = 1;

\*p = j - ms;

}

else

if (a == b)

if (k != \*p)

++k;

else {

j += \*p;

k = 1;

}

else { /\* a > b \*/

ms = j;

j = ms + 1;

k = \*p = 1;

}

}

return(ms);

}

/\* Computing of the maximal suffix for >= \*/

int maxSufTilde(char \*x, int m, int \*p) {

int ms, j, k;

char a, b;

ms = -1;

j = 0;

k = \*p = 1;

while (j + k < m) {

a = x[j + k];

b = x[ms + k];

if (a > b) {

j += k;

k = 1;

\*p = j - ms;

}

else

if (a == b)

if (k != \*p)

++k;

else {

j += \*p;

k = 1;

}

else { /\* a < b \*/

ms = j;

j = ms + 1;

k = \*p = 1;

}

}

return(ms);

}

/\* Two Way string matching algorithm. \*/

void TW(char \*x, int m, char \*y, int n) {

int i, j, ell, memory, p, per, q;

/\* Preprocessing \*/

i = maxSuf(x, m, &p);

j = maxSufTilde(x, m, &q);

if (i > j) {

ell = i;

per = p;

}

else {

ell = j;

per = q;

}

/\* Searching \*/

if (memcmp(x, x + per, ell + 1) == 0) {

j = 0;

memory = -1;

while (j <= n - m) {

i = MAX(ell, memory) + 1;

while (i < m && x[i] == y[i + j])

++i;

if (i >= m) {

i = ell;

while (i > memory && x[i] == y[i + j])

--i;

if (i <= memory)

OUTPUT(j);

j += per;

memory = m - per - 1;

}

else {

j += (i - ell);

memory = -1;

}

}

}

else {

per = MAX(ell + 1, m - ell - 1) + 1;

j = 0;

while (j <= n - m) {

i = ell + 1;

while (i < m && x[i] == y[i + j])

++i;

if (i >= m) {

i = ell;

while (i >= 0 && x[i] == y[i + j])

--i;

if (i < 0)

OUTPUT(j);

j += per;

}

else

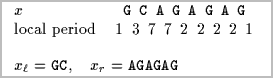
j += (i - ell);

}

}

}

## Kiểm nghiệm



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | 1 | 2 |  | | | | | | | | | | | | | | | | | | | |
| G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | | | |

Shift by: 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Second attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | 1 |  | | | | | | | | | | | | | | | | | | |
|  | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | | |

Shift by: 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 1 |  | | | | | | | | | | | | | | | | | |
|  | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | | |

Shift by: 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fourth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | 1 |  | | | | | | | | | | | | | | | | |
|  | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | | |

Shift by: 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fifth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | 7 | 8 | 1 | 2 | 3 | 4 | 5 | 6 |  | | | | | | | | | | |
|  | | | | | G | C | A | G | A | G | A | G |  | | | | | | | | | | |

Shift by: 7

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sixth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | 1 | 2 |  | | | | | | | |
|  | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | | | |

Shift by: 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seventh attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | 1 | 2 |  | | | | | |
|  | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |  | |

Shift by: 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eighth attempt | | | | | | | | | | | | | | | | | | | | | | | |
| G | C | A | T | C | G | C | A | G | A | G | A | G | T | A | T | A | C | A | G | T | A | C | G |
|  | | | | | | | | | | | | | | | | | | 1 | 2 | 3 |  | | |
|  | | | | | | | | | | | | | | | | G | C | A | G | A | G | A | G |

Shift by: 3