

ĐỐI SÁNH CHUỖI

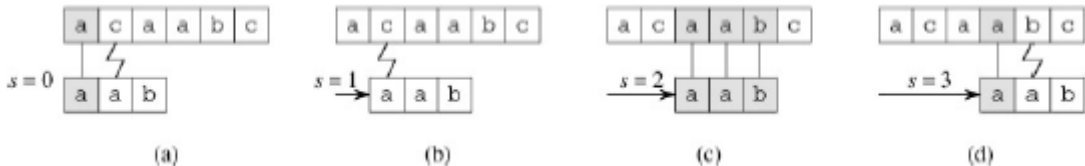
MỤC TIÊU

Hoàn tất bài thực hành này, sinh viên có thể:

- Hiểu và cài đặt được các thuật toán đối sánh chuỗi tiêu biểu:
 - + Brute-Force
 - + Morris-Pratt
 - + Knutt-Morris-Pratt
- Áp dụng đối sánh chuỗi: tìm kiếm trong văn bản

Thời gian thực hành: **từ 120 phút đến 400 phút**

TÓM TẮT

- Đối sánh chuỗi:
Kiểm tra sự tồn tại của 1 chuỗi kí tự (mẫu, pattern) trong 1 chuỗi kí tự có kích thước lớn hơn nhiều (văn bản, text)
Nếu tồn tại trả về 1 hoặc nhiều vị trí xuất hiện.
- Quy ước:
 - + Mẫu cần tìm : P(chiều dài m)
 - + Văn bản T: (chiều dài n)
 - + P và T có cùng tập hữu hạn kí tự Σ ($\Sigma = \{0,1\}; \Sigma = \{A,...,Z\}; ...$)
 - + $m \leq n$
- Tóm tắt các thuật toán:
 - + Brute-Force:
 - Ý tưởng:
Lần lượt kiểm tra điều kiện $P[0..m-1] = T[i..i+m-1]$
Ví dụ:

 - + Morris-Pratt:
 - Ý tưởng:
 - Ghi nhận những phần của T đã trùng với P trước đó
 - Tại 1 vị trí j nào đó xảy ra sự không trùng khớp, thay vì gán j=0, ta gán j bằng 1 số thích hợp

- Giải pháp:

Xây dựng bảng Next: bảng chứa các giá trị j_1 tương ứng với các giá trị j

+ Knutt-Morris-Pratt:

- Ý tưởng:
 - Cải tiến Morris-Pratt bằng cách thay đổi cách tính bảng Next
 - Bổ sung điều kiện $a \neq c$ để tránh trường hợp “không trùng khớp” ngay vị trí đầu tiên sau khi dịch chuyển j
 - Thao tác tìm kiếm không thay đổi

NỘI DUNG THỰC HÀNH

Căn bản

Sinh viên đọc kỹ phát biểu bài tập và thực hiện theo hướng dẫn:

- Cho trước 2 chuỗi T, P
- + Thực hiện tìm kiếm chuỗi P trong T
- + In kết quả ra màn hình

Chương trình mẫu

(Ghi chú: sử dụng 2 thuật toán Brute-Force và Morris-Pratt)

```
#include<string.h>
#include<stdio.h>
#include<conio.h>
void initNextMP(char *P, int NEXT[])
{
    int i, j;
    int m;
    m= strlen(P);
    i = 0;
    j = NEXT[0] = -1;
    while (i < m)
    {
        if ((j == -1) || (P[i] == P[j]))
        {
            i++;
            j++;
            if (i >= m)
                break;
            NEXT[i] = j;
        }
        else j = NEXT[j]; //(3)
    }
}

int MPMatcher(char *P, char *T)
{
    int n = strlen(T);
    int m = strlen(P);
    int *NEXT=new int[m];
    initNextMP(P, NEXT);
    int j = 0, i = 0;
    for (i = 0; i < n; i++)
    {
```

```

        while (j > -1 && T[i] != P[j])
            j = NEXT[j];
        j++;
        if (j >= m)
            return (i-j+1);
    }
    return -2;
}
int bruteForceMatcher(char *T, char *P)
{
    int m, n, j;
    m = strlen(P);
    n = strlen(T);
    for (int i = 0; i <= n - m; i++) { //(1)
        j = 0; //(2)
        while ((j < m) && (P[j] == T[i + j])) j++;
        if (j == m) return i;
    }
    return -1;
}
void main()
{
    int luachon;
    char *T, *P;
    int ketqua;
    T=new char[256];
    P=new char[128];
    printf("nhập chuỗi T");
    fflush();
    gets(T);
    fflush();
    printf("nhập chuỗi P");
    gets(P);
    do
    {
        printf("\nNhan 0 de thoat\n");
        printf("nhap 1: de chon Brute Force\n");
        printf("nhap 2: de chon Morris-Pratt\n");
        scanf("%d", &luachon);
        switch(luachon)
        {
            case 1:{
                ketqua=bruteForceMatcher(T,
P);
                printf("vị trí chuỗi P xuất
hien trong T: %d", ketqua);
            }
            break;
            case 2:{
                ketqua=MPMatcher(P,T);
                printf("vị trí chuỗi P xuất
hien trong T: %d", ketqua);
            }
            break;
        }
    }while(luachon!=0);
    getch();
}

```

Yêu cầu

1. Biên dịch đoạn chương trình trên
2. Kiểm tra kết quả in ra màn hình khi người dùng nhập vào giá trị 2 chuỗi P và T
3. Giải thích ý nghĩa của 1 số dòng lệnh trong đoạn chương trình:
 - (1) trong vòng lặp for, tại sao giá trị i chỉ cần thỏa điều kiện : $i < n - m$
 - (2) j được gán: $j = 0$ khi nào và tại sao cần khởi gán lại giá trị này cho j
 - (3) j được gán: $j = \text{NEXT}[j]$ khi nào và tại sao cần khởi gán lại giá trị này cho j
4. Viết hàm tạo bảng Next theo ý tưởng của thuật toán Knutt-Morris-Pratt: Sử dụng lại hàm `initNextMP` trong chương trình trên bằng cách cập nhật lại đoạn code như sau:

```
initNEXT (Morris-Pratt):  
...  
NEXT[i] = j;  
}  
  
initNEXT (Knuth-Morris-Pratt):  
...  
if (p[i] != p[j]) NEXT[i] = j;  
else NEXT[i] = NEXT[j];
```

Viết 1 chương trình như sau: nhận 1 chuỗi P, in ra màn hình giá trị trong bảng Next ứng với 2 thuật toán Morris-Pratt và Knutt-Morris-Pratt. So sánh, nhận xét kết quả.

5. Sử dụng lại chương trình trên, thay hàm `void initNextMP(char *P, int NEXT[])` bằng hàm vừa được tạo ở câu 4. Biên dịch chương trình để thực hiện tìm kiếm chuỗi P trong T.

BÀI TẬP THÊM

- Viết chương trình dạng Console tên `StrMatching`. Chương trình được thi hành với cú pháp dòng lệnh:

StrMatching <option> <paras>

- + Tùy theo <option> mà chương trình sẽ hoạt động như mô tả bên dưới:

- <option> là `-mp`: Khi đó <paras> có dạng:

<T> <P>

Chương trình sẽ thực hiện tìm kiếm nội dung file <P> trong <T> dùng thuật toán Morris-Pratt

- <option> là `-kmp`: Khi đó <paras> có dạng:

<T> <P>

Chương trình sẽ thực hiện tìm kiếm nội dung file <P> trong <T> dùng thuật toán Knutt-Morris-Pratt

- + Với cả hai option chương trình đều in ra cửa sổ Console các thông tin sau:

- Thông tin bảng Next
- Tất cả các vị trí mà P xuất hiện trong T nếu có, ngược lại thông báo không tồn tại

Ví dụ:

StrMatching -mp T.txt P.txt: tìm P trong T bằng thuật toán Morris-Pratt