

CHAPTER 04

CÁC THUẬT TOÁN XỬ LÝ XÂU KÝ TỰ

Design by Minh An

Email: anvanminh.hau@gmail.com

Các thuật toán xử lý chuỗi ký tự

Bài toán 1:

Cho chuỗi ký tự S. Hãy liệt kê các chuỗi con khác nhau có độ dài K của S.

Ví dụ:

S = "abccaba", và K = 3 thì ta có các chuỗi các con khác nhau: "abc", "aba", "abb", "acc", "aca", "acb", "aab", "aaa", "bcc", "bca",....

Design by Minh An

Bài toán 1

```
void ThucHien(int i) {
    char tam[100]; //Túi để "soi" các ký tự ở vị trí i
    tam[0] = '\0';
    if (strlen(xc) == k) {
        cout<<xc<<endl;
        dem++;
    }
    else {
        for (int j = i; j < n; j++) {
            if (comat(s[j], tam) == false){
                them(tam, s[j]); //thêm s[j] vào "túi" tam
                them(xc, s[j]); //thêm s[j] vào chuỗi con xc
                ThucHien(j + 1); //thực hiện tiếp |xc|=j+1
                xoa(xc); //xóa ký tự cuối cùng của chuỗi xc
            }
        }
    }
}
```

Design by Minh An

Các thuật toán xử lý chuỗi ký tự

Bài toán 2:

Cho 2 chuỗi ký tự T, và P chỉ gồm các ký tự {'0'...'9', 'a'...'z', 'A'...'Z'}.

Yêu cầu:

Kiểm tra xem P Có phải là một chuỗi con của T hay không?

Phương pháp đơn giản:

Sử dụng thuật toán lặp

Design by Minh An

Thuật toán lặp

```
int indexOf(const char *p, const char *t){
    int m = strlen(p);
    int n = strlen(t) - m;
    for (int i = 0; i < n; i++){
        int j = 0;
        while (j < m && t[i + j] == p[j]){
            j++;
        }
        if (j == m){
            return i;
        }
    }
    return -1;
}
```

Design by Minh An

Bài toán 2: Thuật toán Boyer Moore Horspool

Phương pháp:

- So sánh ngược từ ký tự cuối của P trở về đầu.
- Giả sử vị trí bắt đầu so sánh trong T là i, và vị trí cuối cùng của P là v (v = len(P)).
- Ta sẽ so sánh T[i] với P[v] và dịch chuyển về đầu.
- Nếu việc khớp từng ký tự vượt qua được P[0] thì P có mặt trong T.
- Ngược lại:
 - Nếu ngay từ đầu T[i] không có trong P, thì $i = i + v$
 - Ngược lại: gọi x là vị trí mà T[x] không có trong P, thì $i = i + v - x - 1$

Design by Minh An

Thuật toán Boyer Moore Horspool

Ví dụ 1:

- Xâu T = "mothaibabonnamsau"b a y
- Xâu P = "namsau"
- $v = \text{strlen}(P) = 6$
- Do vậy ta bắt đầu từ vị trí $i = 5$ trong T

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
m	o	t	h	a	b	a	b	o	n	n	a	m	s	a	u	b	a	y	
n	a	m	s	a	u														

Design by Minh An

Ví dụ 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
m	o	t	h	a	i	b	a	b	o	n	n	a	m	s	a	u	b	a	y

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
m	o	t	h	a	i	b	a	b	o	n	n	a	m	s	a	u	b	a	y

Design by Minh An

Thuật toán Boyer Moore Horspool

Ví dụ 2:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T

Không có 'A' trong P
→ $i = i + 6 = 5 + 6 = 11$

Có 'D' tại $x = 0$ trong P
→ $i = i + 6 - 0 - 1 = 9 + 5 = 14$

Có 'D' tại $x = 0$ trong P
→ $i = i + 6 - 0 - 1 = 14 + 5 = 19$

Design by Minh An

Thuật toán Boyer Moore Horspool

```

int Boyer_Moore_Horspool() {
    int dem = 0, i = strlen(P), v = strlen(P);
    while (i < strlen(T)) {
        int x = v - 1, j = i - 1;
        while (T[j] == P[x] && x > -1) {
            j--; x--;
        }
        if (x < 0) { dem++; i = i + v; }
        else {
            x = char_in_string(T[j], P);
            if (x < 0) i = i + v;
            else i = i + v - x - 1;
        }
    }
    return dem;
}

```

Design by Minh An

Thuật toán Z

- Cho một xâu ký tự S chỉ gồm các chữ cái và chữ số.
- Định nghĩa:** Xâu tiền tố của S là một xâu con của S tính từ vị trí đầu tiên (prefix - substring).
- Phương pháp: Thuật toán Z là xây dựng một mảng Z[...] với ý nghĩa Z[i] là độ dài của xâu tiền tố bắt đầu từ vị trí i.
- Ví dụ S = "ACBDABACBAC" thì:
Mảng Z = {1, 0, 0, 0, 1, 0, 3, 0, 0, 2, 0}

A	C	B	D	A	B	A	C	B	A	C
-1	0	0	0	1	0	3	0	0	2	0

Xâu con dài nhất giống phần đầu, từ vị trí này có độ dài 1 (A)

Xâu con dài nhất giống phần đầu, từ vị trí này có độ dài 3 (ACB)

Xâu con dài nhất giống phần đầu, từ vị trí này có độ dài 2 (AC)

Design by Minh An

Thuật toán Z

- Ứng dụng:** Tìm trong xâu T xem có bao nhiêu xâu con P trong đó.
- Thuật toán:** Xâu tiền tố của S là một xâu con của S tính từ vị trí đầu tiên (prefix - substring).
 - Ta sẽ tạo một xâu mới $S = P + \$ + T$ (trong đó ký tự '\$' không có trong các xâu T và P).
 - Sau đó áp dụng thuật toán Z để tìm các tiền tố của S.
 - Độ dài tiền tố nào bằng độ dài của P, thì đó chính là xâu con P.

Design by Minh An

Thuật toán Z

Ví dụ:

- T = "Ban Viet la nguoi Viet Nam"
- P = "Viet"
- S = "Viet\$Ban Viet o Viet Nam".
- Khi đó có mảng Z như sau:

V	i	e	t	\$	B	a	n		V	i	e	t		o		V	i	e	t		N	a	m
*	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	4	0	0	0	0	0	0	0

Design by Minh An

Thuật toán Z

Phương pháp:

- Duyệt chuỗi S: i từ 1 đến n - 1 (mảng bắt đầu từ 0).
- Mỗi vị trí i ta quản lý một đoạn [left, right] với right lớn nhất có thể sao cho xâu con từ left tới right là tiền tố của xâu S
- Ban đầu left = right = 0.
- Giả sử ở i ta đã có đoạn [left, right] của vị trí i - 1 và giả sử đã tính được tất cả các giá trị Z trước đó.
- Chia trường hợp để cập nhật left, right, và Z[i] như sau:
 - Nếu $i > r$, trong trường hợp này không có tiền tố nào bắt đầu trước i và kết thúc sau i:
 - Vì vậy ta gán lại left = i
 - Cho r chạy từ i trở lên để tìm đoạn [left, right] mới.
 - Sau đó tính $Z[i] = \text{right} - \text{left} + 1$.

Design by Minh An

Phương pháp

2. Trường hợp ngược lại, $i \leq \text{right}$.

- Đặt $k = i - \text{left}$, ta thấy xâu $S[k...]$ và xâu $S[i...]$ giống nhau ở ít nhất $\text{right} - i + 1$ phần tử đầu, vì vậy có thể tận dụng $Z[k]$ để tính $Z[i]$.
- Ta có: $Z[i] \geq \min(Z[k], \text{right} - i + 1)$.
- Nếu $Z[k] < \text{right} - i + 1$ thì $Z[i] = Z[k]$
- Nếu $Z[k] \geq \text{right} - i + 1$ thì:
 - gán lại left = i và cho r tiếp tục tăng để tìm đoạn [left, right] mới.
 - Sau đó cũng có $Z[i] = \text{right} - \text{left} + 1$ như trên.

Design by Minh An

Thuật toán Z

```
void z_algo(const char *s, int *z) {
    int n = strlen(s), l = 0, r = 0;
    for (int i = 1; i < n; i++) {
        if (i > r) {
            l = r = i;
            while (r < n && s[r - l] == s[r]) r++;
            z[i] = r - l; r--;
        } else if (z[i - l] < r - i + 1)
            z[i] = z[i - l];
        else { l = i;
            while (r < n && s[r - l] == s[r]) r++;
            z[i] = r - l; r--;
        }
    }
}
```

Design by Minh An

Tin xâu con chung dài nhất

- Định nghĩa:** Xâu ký tự A được gọi là xâu con của xâu ký tự B nếu ta có thể xóa đi một số ký tự trong xâu B để được xâu A.
- Bài toán:** Cho biết hai xâu ký tự A và B, hãy tìm xâu ký tự C có độ dài lớn nhất và là con của cả A và B.
- Phương pháp:** Sử dụng quy hoạch động.
 - Gọi $L_{i,j}$ là độ dài xâu con chung dài nhất của xâu A_i gồm i kí tự đầu của A ($A_i = A[1..i]$) và xâu B_j gồm j kí tự phần đầu của B ($B_j = B[1..j]$).
 - Ta có công thức quy hoạch động như sau:
 - $L_{0,j} = L_{i,0} = 0$ // Một trong 2 xâu là rỗng
 - $L_{i,j} = L_{i-1,j-1} + 1$ nếu $A[i] = B[j]$.
 - $L_{i,j} = \max(L_{i-1,j}, L_{i,j-1})$ nếu $A[i] \neq B[j]$.

Design by Minh An

Tin xâu con chung dài nhất

- Cài đặt:** Sử dụng bảng phương án là một mảng 2 chiều $L[\text{len}(A)][\text{len}(B)]$ để lưu các giá trị của hàm QHĐ $L(i, j)$.

```
algo_1() {
    for (int i = 0; i < len(A); i++)
        L[i][0] = 0;
    for (int j = 0; j < len(B); j++)
        L[0][j] = 0;
    for (int i = 0; i < len(A); i++)
        for (int j = 0; j < len(B); j++)
            if (A[i] == B[j])
                L[i][j] = L[i-1][j-1] + 1;
            else
                L[i][j] = max(L[i-1][j], L[i][j-1]);
}
```

Design by Minh An

Tìm xâu con chung dài nhất

- **Cài đặt:** Sử dụng bảng phương án là một mảng 2 chiều $L[\text{len}(A)][\text{len}(B)]$ để lưu các giá trị của hàm QHĐ $L(i, j)$.

```
algorithm() {  
    for (int i = 0; i < len(A); i++)  
        L[i][0] = 0;  
    for (int j = 0; j < len(B); j++)  
        L[0][j] = 0;  
    for (int i = 0; i < len(A); i++)  
        for (int j = 0; j < len(B); j++)  
            if (A[i] == B[j])  
                L[i][j] = L[i-1][j-1] + 1;  
            else  
                L[i][j] = max(L[i-1][j], L[i][j-1]);  
}
```

Design by Minh An