

Các thuật toán xử lý xâu ký tự

Bài toán 1:

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

Ví du:

S = "abccaba", và K = 3 thì ta có các xâu 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 dễ 'soi" các ký tự ở vị trí i
    tam[0] = '\0';
    if (strlen(xc) == k) {
        cout<<xc<<end1;
        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 xâu con xc
                ThucHien(j + 1); //thực hiện tiếp |xc|=j+1
                      xoa (xc); //xoá kỳ tự cuối cùng của xâu xc
        }
    }
    }
}</pre>
```

Các thuật toán xử lý xâu ký tự

Bài toán 2:

Cho 2 xâu 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 xâu con của T hay không?

· Phương pháp đơn giản:

Sử dụng thuật toán lặp

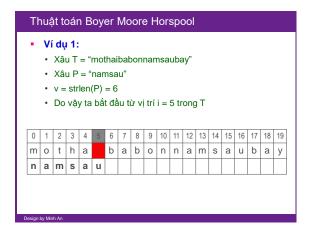
Design by Minh An

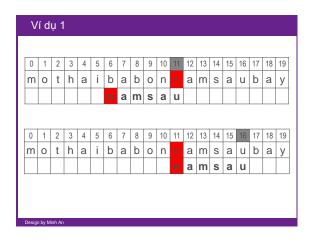
Bài toán 2: Thuật toán Boyer Moore Horspool

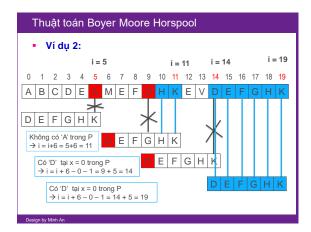
Phương pháp:

- 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







```
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;
}</pre>
```

```
Thuật toán Z
• Cho một xâu ký tư 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:
       Mång Z = \{1, 0, 0, 0, 1, 0, 3, 0, 0, 2, 0\}
       С
               В
                                 В
                                      Α
                                             С
                                                               С
                                                         Α
        0 0
                    0
                                 0
                                                               0
   Xâu con dài nhất giống
                           Xâu con dài nhất
giống phần đầu, từ vị
                                                   Xâu con dài nhất
                                                 giống phần đầu, từ vị
        đô dài 1 (A)
                             trí này có độ dài 3
                                                  trí này có độ dài 2
                                  (ACB)
                                                        (AC)
```

Thuật toán Z Ứng dụng: Tìm trong xâu T xem có bao nhiều 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 tim các tiền tố của S. Độ dài tiền tố nào bằng độ dài của P, thì đó chính là xâu con P.



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 lí 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] = right left + 1.

Design by Minh An

Phương pháp

- 2. Trường hợp ngược lại, i ≤ right.
- Đặt k = i left, ta thấy xâu S[k...] và xâu S[i...] giống nhau ở ít nhất right - i + 1 phần tử đầu, vì vậy có thể tận dụng Z[k] để tính Z[i].
- Ta có: Z[i] ≥ min(Z[k], right i + 1).
- Nếu Z[k] < r − i + 1 thì Z[i] = Z[k]
- Nếu Z[k] ≥ r 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] = right 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 --;
    }
}</pre>
```

Design by Minh A

Tìn 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 xoá 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_i = 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
 - $\begin{array}{ll} \text{-} & L_{i,\,j} = L_{i-1,\,j-1} + 1 & \text{n\'eu A[i]} = B[j]. \\ \text{-} & L_{i,\,j} = \text{max}(L_{i-1,\,j},\,L_{i,\,j-1}) & \text{n\'eu A[i]} \neq B[j]. \end{array}$

Docine by Minh /

Tìn xâu con chung dài nhất

 Cài đặt: Sử dụng bảng phương án là một mảng 2 chiều L[len(A)][len(B)] để lưu các giá trị của hàm QHD 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]);
}</pre>
```

Design by Minh

Tìn xâu con chung dài nhất

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

```
algorith() {
  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]);
}</pre>
```

Design by Minh An