BÀI 5

Email: anvanminh.haui@gmail.com

Các thuật toán xử lý xâu ký tự

1

Các thuật toán xử lý xâu ký tự

1. Một số bài toán cơ bản
2. Thuật toán Boyer Moore Horspool
3. Thuật toán Z
4. Tìm xâu con chung dài nhất

2

Một số bài toán cơ bản

- Bài toán 1:
  - Cho xâu ký tự s. Hãy cho biết số loại ký tự có trong xâu s và số lần xuất hiện của mỗi loại ký tự trong xâu s.
- Ví dụ:
  - s = "abccabdad123", có 7 loại ký tự, ký tự a xuất hiện 3 lần, b: 2 lần, c: 2 lần, d: 2 lần, 1: 1 lần, 2: 1 lần và 3: 1 lần.

3

Một số bài toán cơ bản

- Bài toán 2:
  - Cho 2 xâu ký tự T, và P chỉ gồm các ký tự {'0' ... '9', 'a' ... 'z', 'A' ... 'Z'}.
  - Kiểm tra xem P Có phải là một xâu con của T hay không?
- Phương pháp đơn giản:
  - Sử dụng thuật toán vét cạn.

4

Thuật toán vét cạn

```
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;
}
```

5

Thuật toán Boyer Moore Horspool

- Bài toán 3:
  - Cho 2 xâu ký tự T, và P chỉ gồm các ký tự {'0' ... '9', 'a' ... 'z', 'A' ... 'Z'}.
  - Kiểm tra xem P Có phải là một xâu con của T hay không?
- Phương pháp:
  - Sử dụng thuật toán Boyer Moore Horspool.

6

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 P[v] (v = strlen(P) - 1).
  - 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 (có T[i] != P[v]):
    - + Nếu T[i] không có trong P, thì i = i + v
    - + Ngược lại (T[i] có trong P): gọi x là vị trí xuất hiện đầu tiên của T[i] trong P (T[i] = P[x]), thì i = i + v – x – 1

7

Thuật toán Boyer Moore Horspool

- Ví dụ 1:
  - Xâu T = "mothaibabonnamsaubay"
  - Xâu P = "namsau"
  - v = 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	i	b	a	b	o	n	n	a	m	s	a	u	b	a	y
n	**a**	**m**	**s**	**a**	**u**														

8

Ví dụ 1

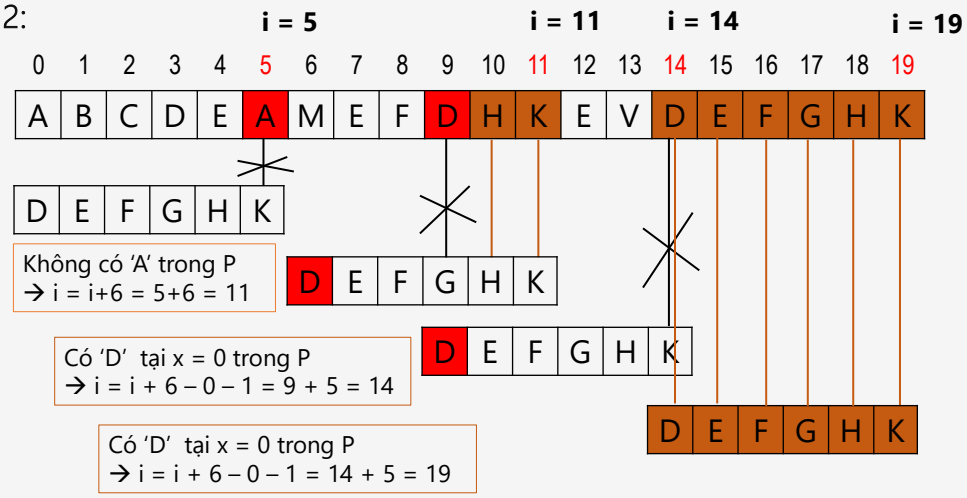
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
m	o	t	h	a	i	b	a	b	o	n	n	a	m	s	a	u	b	a	y
						n	**a**	**m**	**s**	**a**	**u**								

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
											n	**a**	**m**	**s**	**a**	**u**			

9

Thuật toán Boyer Moore Horspool

- Ví dụ 2:



10

Thuật toán Boyer Moore Horspool

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

11

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}

12

Thuật toán Z

- 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)

13

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).
  - 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.

14

Thuật toán Z

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

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
V	i	e	t	$	B	a	n		V	i	e	t		o		V	i	e	t		N	a	m
-1	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	4	0	0	0	0	0	0	0

15

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 2 trường hợp để cập nhật left, right, và Z[i].

16

Thuật toán Z

- Chia 2 trường hợp để cập nhật left, right, và Z[i] như sau:
  - Trường hợp 1: Nếu i > right, trong trường hợp này không có tiền tố nào bắt đầu trước i và kết thúc sau i.
    - Gán left = i;
    - Cho right chạy từ i trở lên để tìm đoạn [left, right] mới;
    - Sau đó tính Z[i] = right – left;

17

Thuật toán Z

- Trường hợp 2, 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] < right – i + 1 thì Z[i] = Z[k]
  - Nếu Z[k] ≥ right – i + 1 thì:
    - gán lại left = i và cho right tiếp tục tăng để tìm đoạn [left ,right] mới.
    - Sau đó cũng có Z[i] = right – left như trên.

18

Thuật toán Z

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

19

Tìm 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_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] ≠ B[j].:

20

Thuật toán quy hoạch động

- Cài đặt: Sử dụng bảng phương án là một mảng 2 chiều L[len(A) + 1][len(B) + 1] để lưu các giá trị của hàm quy hoạch động 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 = 1; i <= len(A); i++)
      for (int j = 1; j <= len(B); j++)
         if (A[i-1] == B[j-1]
            L[i][j] = L[i-1][j-1] + 1;
         else
            L[i][j] = max(L[i-1][j], L[i][j-1]);
   return L[m][n];
}
```

21

Truy vết tìm xâu con chung dài nhất

```
xau_con() {
   int i = len(A), j = len(B), k = 0;
   while (L[i][j] != 0) {
      if (A[i-1] == B[j-1]) {
            xc[k] = A[i-1];
            k++; i--; j--;
      }
      else {
            if (L[i-1][j] > L[i][j-1]) i--;
            else j--;
      }
   }
   xc[k] = '\0'; strrev(xc);
}
```

22