


SUFFIX STRUCTURES

Lecture 01

Problem

- Cho 2 chuỗi phân biệt **s** và **t**.
- Hãy chuyển chuỗi **s** thành chuỗi **t** thông qua các phép biến đổi:
 1. Xóa 1 ký tự bất kỳ trong chuỗi **s**
 2. Hoán đổi vị trí của hai ký tự bất kỳ trong chuỗi **s**
- Xác định:
 - **s** transform  **t**?
 - Nếu chuyển được thì sử dụng phép biến đổi nào ?
 - Biết rằng số lần biến đổi là không giới hạn

Input

- $|s| > 0$
- $|t| > 0$
- $s \neq t$
- s và t chỉ gồm chữ cái latin in thường

Output

- “**need tree**” nếu không thể biến s thành t
- “**automaton**” chuyển s thành t được thông qua phép biến đổi số 1
- “**array**” chuyển s thành t được thông qua phép biến đổi số 2
- “**both**” nếu cần thực hiện cả 2 phép biến đổi

Sample test 1

Input

s = 'automaton'

t = 'tomat'

Output

automaton

Giải thích:

s = 'automaton'

t = 'tomat'

Sample test 2

Input

```
s = 'array'  
t = 'arary'
```

Output

```
array
```

Giải thích:

s = 'ar**ay**' → **swap** **r** với **a** ta được s = 'ar**ay**'

t = 'ar**ay**'

Sample test 3

Input

s = 'both'

t = 'hot'

Output

both

Giải thích:

s = '~~b~~oth' → **xóa b** ta được s = '**oth**' → **swap** (o,h) thì s = '**hto**'

→ **swap** (t, o) và s = 'hot' = t

Sample test 4

Input

s = 'need'

t = 'tree'

Output

need tree

Giải thích: để biến s thành t thì trong s phải có ít nhất 1 chữ r và 1 chữ t → không có cách nào biến đổi s thành t được.

Solution

- TH1: Nếu 1 kí tự nào đó có trong t mà không có trong s → **'need tree'**
- TH2: Nếu 1 kí tự có trong s mà không có trong t thì xóa kí tự đó đi → **'automaton'**
- TH3: Nếu thứ tự xuất hiện của các ký tự trong t không khớp với s → **'array'**
- TH4: Kết hợp TH2 + TH3 → **'both'**

Solution

Để giải quyết trường hợp **TH1** ('**need tree**') và **TH2**('automaton'):

- Tạo mảng đếm tần suất của các kí tự trong chuỗi s và t: cnt_s và cnt_t
- Đối với từng kí tự x trong bảng chữ cái latin (26 chữ cái):
 - Nếu $\text{cnt_t}[x] > \text{cnt_s}[x] \rightarrow$ '**need tree**'
 - Ngược lại: $\text{cnt_t}[x] < \text{cnt_s}[x] \rightarrow$ '**automaton**'

Solution

Input

s = 'automaton'

t = 'tomat'

Output

automaton

Minh họa mảng cnt_s và cnt_t:

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
cnt_s	2												1	1	2					2	1					

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
cnt_t	1												1		1					2						

Solution

- Để giải quyết trường hợp **TH3 'array'**:
 - Gọi match là vị trí xuất hiện của ký tự trước đó của t trong s
 - Duyệt qua từng ký tự t[i] trong chuỗi t
 - Tìm vị trí xuất hiện đầu tiên của t[i] trong s mà xuất hiện sau match
 - Nếu các vị trí này tăng dần nghĩa là các ký tự này đã đúng vị trí, ta không cần đổi chỗ. Ngược lại ta kết luận "array".
- Trường hợp **TH4 'both'**: nếu cần cả '**automaton**' và '**array**'
- Độ phức tạp: **$O(\max(\text{length}(s), \text{length}(t)))$**

Pseudo code (phần nhập xuất, khởi tạo)

```
read s, t
cnt_s = cnt_t = [0] * 26
for i = 0 .. len(s) - 1
    cnt_s[s[i] - 'a'] += 1
for i = 0 .. len(t) - 1
    cnt_t[t[i] - 'a'] += 1
need_tree = automaton = array = false
```

Pseudo code (phần xử lý logic)

```
for i = 0 .. 25
  if cnt_t[i] > cnt_s[i]
    need_tree = true //Case 1
  else if cnt_t[i] < cnt_s[i]
    automaton = true //Case 2
match = -1
for i = 0 .. len(t) - 1
  index = first occurrence of t[i] in s[match + 1, len(s) - 1]
  if found index
    match = index
  else
    array = true //Case 3
    break
```

Pseudo code (in kết quả)

```
if need_tree
    print 'need tree'
else automaton and array
    print 'both'
else if automaton
    print 'automaton'
else
    print 'array'
```