

序列和集合的算法IV

Instructor: Shizhe Zhou

Course Code:00125401

Kmp for string matching

$A = xyxxyxyxyxyxyxyxyx$, $B = xyxyxyxyx$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	x	y	x	x	y	x	y	x	y	y	x	y	x	y	x	y	y	x	y	x	y	x	x	
1:	x	y	x	y
2:	.	x
3:	.	.	x	y
4:	.	.	.	x	y	x	y	y
5:	x
6:	x	y	x	y	y	x	y	x	y	x	x
7:	x
8:	x	y	x
9:	x
10:	x
11:	x	y	x	y	y
12:	x
13:	x	y	x	y	y	x	y	x	y	x	x	.

图 6.20 字符串直接匹配的例子

串内自重复性质: $next(i)$

与 $B(i-1)$ 的前缀相等的最大后缀的长度:

$next(i)$ 等于使 $b_{i-j}b_{i-j+1}\cdots b_{i-1} = B(j)$ 的最大的 j ($0 < j < i-1$), 如果不存在这样的 j , 则返回 0。

$B =$	x	y	x	y	y	x	y	x	y	x	x
		x	\cdot	\cdot	\cdot						
			x	y	x	\cdot	\cdot	\cdot			
				x	\cdot	\cdot	\cdot				
					x	\cdot	\cdot	\cdot			
						x	y	x	y	y	
							x	\cdot	\cdot	\cdot	
								x	y	x	

模式自匹配

$i =$	1	2	3	4	5	6	7	8	9	10	11
$B =$	x	y	x	y	y	x	y	x	y	x	x
$next =$	-1	0	0	1	2	0	1	2	3	4	3

图 6.22 $next$ 表中的值

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
x	y	x	x	y	x	y	x	y	y	x	y	x	y	x	y	y	x	y	x	y	x	x
x	y	x	y	y	x	y	x	y	x	x												

举i=8,16的例子说明kmp运行机制

Compute next

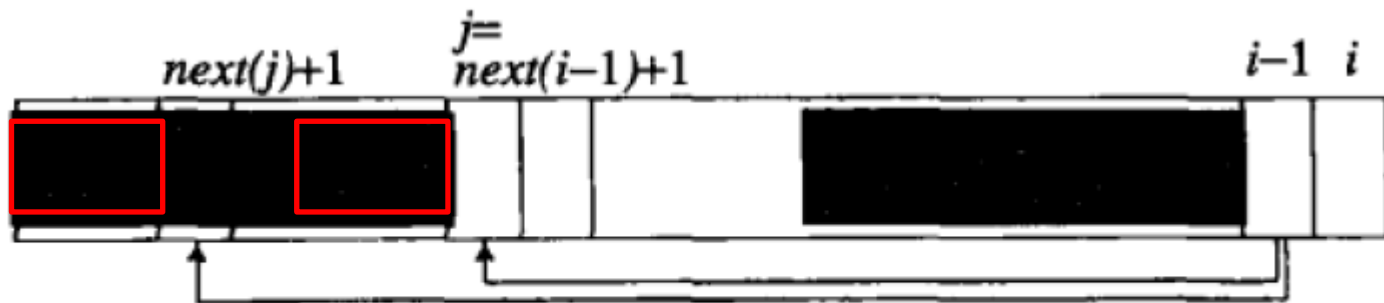


图 6.24 计算 $next(i)$

算法 C_1

输入: B (长度为 m 的字符串)

输出: $next$ (大小为 m 的数组)

begin

$next(1) := -1;$

$next(2) := 0;$

for $i := 3$ **to** m **do**

$j := next(i-1) + 1;$

while $b_{i-1} \neq b_j$ **and** $j > 0$ **do**

$j := next(j) + 1;$

$next(i) := j$

end

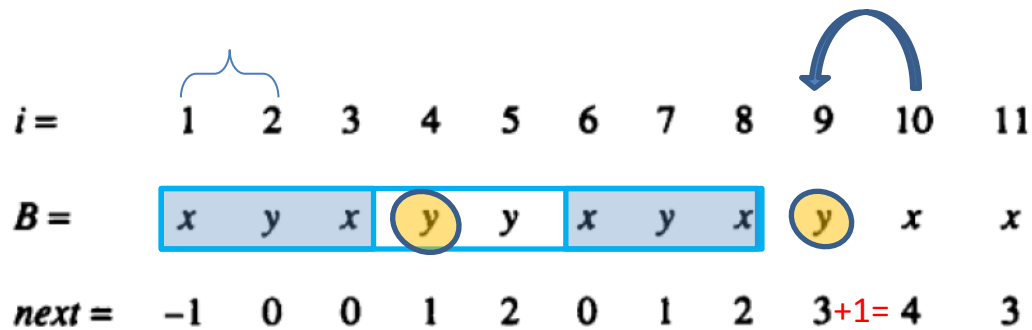


图 6.22 $next$ 表中的值

图 6.25 算法 $Compute_Next$

Next函数的递归计算

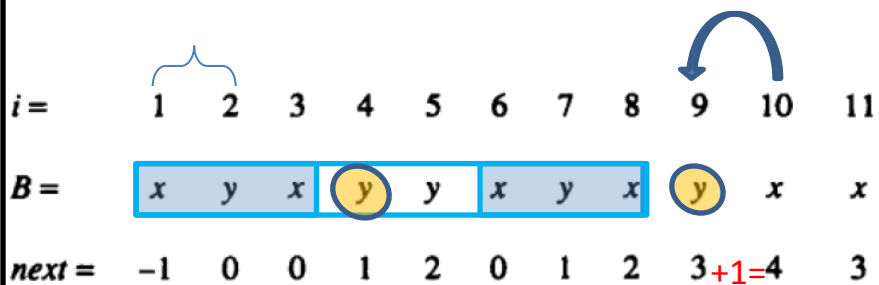


图 6.22 next 表中的值

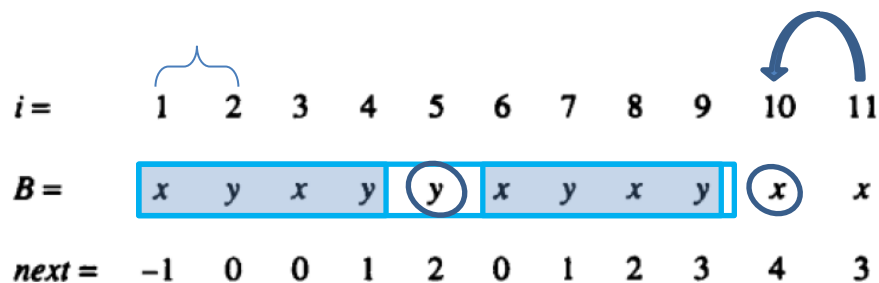


图 6.22 next 表中的值

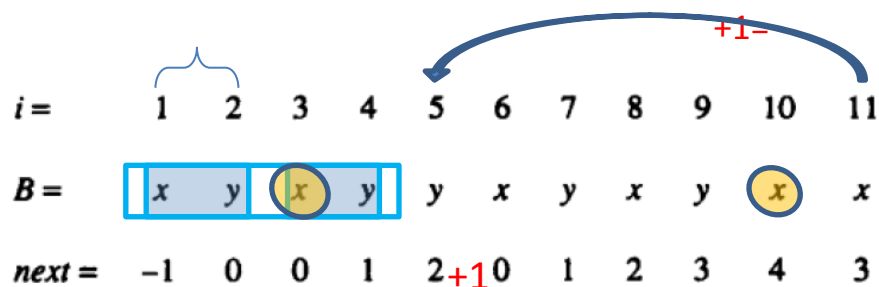


图 6.22 next 表中的值

Kmp for string matching

算法 *String_Match* (A, n, B, m)

输入: A (长度为 n 的字符串) 和 B (长度为 m 的字符串)

{设 $next$ 值已经算好, 见图 6.25}

输出: $Start$ (使得 B 是 A 中从 $A[Start]$ 开始的子串的第一个下标)

begin

$j := 1; i := 1;$

$Start := 0;$

while $Start = 0$ and $i \leq n$ **do**

if $B[j] = A[i]$ **then**

$j := j + 1;$

$i := i + 1$

else

$j := next[j] + 1;$ Move B right by $(j - next[j] - 1).$

if $j = 0$ **then**

$j := 1;$

$i := i + 1;$

if $j = m + 1$ **then** $Start := i - m$ Succeed!

end

Next值越小, 移动量越大;
注意 $next(j) < j - 1$

Minimum edit

- Edit string A to become B with fewest steps: insertion, deletion or replace.

abbc → babb

Min cost of modify
An to be Bm :

$$C(n, m) = \min \begin{cases} C(n-1, m) + 1 & \text{(删除 } a_n \text{)} \\ C(n, m-1) + 1 & \text{(插入与 } b_m \text{ 匹配的某个字符)} \\ C(n-1, m-1) + c(n, m) & \text{(替换或者匹配 } a_n \text{)} \end{cases}$$

$$c(i, j) = \begin{cases} 0 & \text{如果 } a_i = b_j \\ 1 & \text{如果 } a_i \neq b_j \end{cases}$$

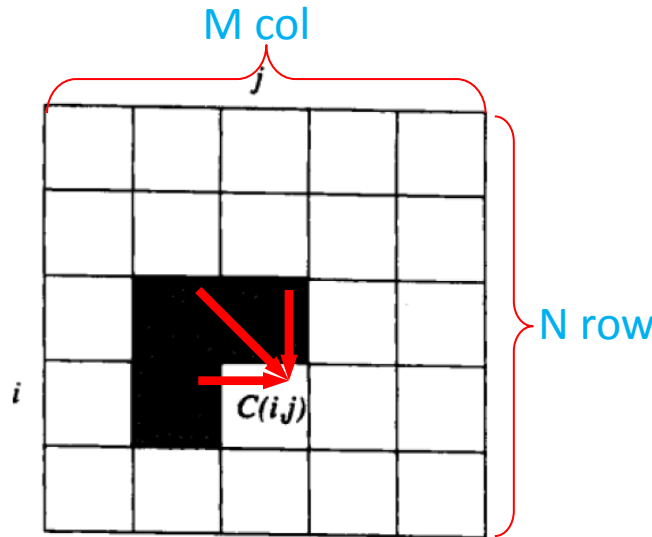


图 6.26 $C(i, j)$ 的依赖关系

Randomized algorithm

- Monte carlo → the chance of giving wrong answer is neglectable.
- Las vegas → always gives a true output, costly.

Find a number in larger half using monte carlor sampling 100 seeds: $1 - (\frac{1}{2})^{100}$

Random number generator

- 线性同余法 $r(i) = (r(i-1) \cdot b + 1) \bmod t$
- Linear congruential generator: $X_{n+1} \equiv (aX_n + c) \pmod{m}$

t 应该相当大

b 应该比 t 少一位数, 最后三位应该是 $x21$, 其中 x 是偶数


```
#include "math.h"
#include "windows.h"

srand(GetTickCount());
rand(); /*[0,65535]*/
```


Longest increasing sequence

- x_1, x_2, \dots, x_n 中挑选子序列 $x_{i_1}, x_{i_2}, \dots, x_{i_k}$. 满足对任意 $j < k$, $x_j < x_{j+1}$. LIS 为这些子序列中最长者.


归纳假设1:



归纳假设 (首次尝试): 给定某个长度小于 m 的序列, 知道如何求它的某个最长的递增序列。



归纳假设 (第二次尝试): 给定某个长度小于 m 的序列, 知道如何求它的所有最长的递增序列。



归纳假设 (第三次尝试): 给定某个长度小于 m 的序列, 知道如何求它的某个最长的递增序列, 使得其他最长递增序列的末尾的数都不比这个序列末尾的数小。

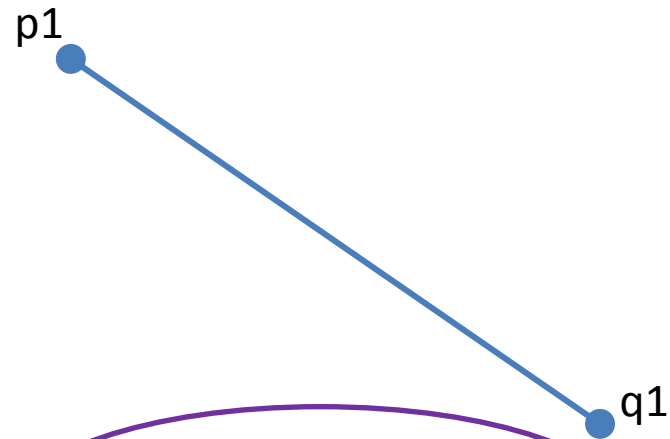
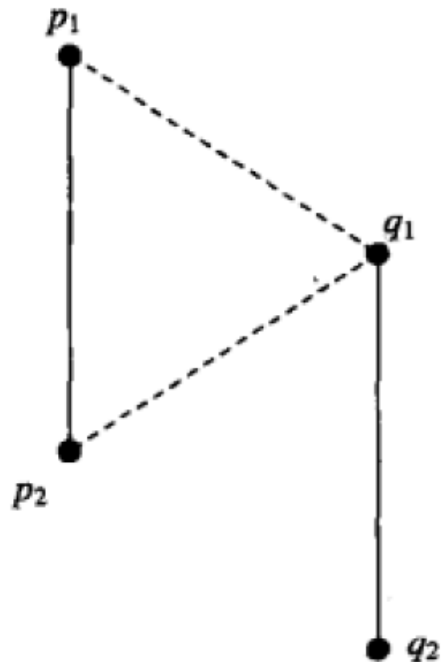
归纳假设 (第四次尝试): 给定某个长度小于 m 的序列, 知道如何对任意 $k < m-1$ 求出 $BIS(k)$, 如果存在的话。

- 令长度为n的LIS中，last元素最小者为BIS.
 - **Key observation:** $\text{BIS}(1).\text{last} < \text{BIS}(2).\text{last} < \dots < \text{BIS}(s).\text{last}$.
- Proof:** 反设若存在某个 $\text{BIS}(m-1).\text{last} > \text{BIS}(m).\text{last}$, 有 $\text{BIS}(m-1).\text{last} > \text{BIS}(m).\text{last} > \text{BIS}(m)[\text{last}-1]$
- 则 $\text{BIS}(m)$ 的 长度为m-1的前缀 形成一个 $\text{LIS}(m-1)$
- 满足: $\text{LIS}(m-1).\text{last} < \text{BIS}(m-1)$, 矛盾!

Code: <http://staff.ustc.edu.cn/~szhou/course/algorithmFundamentals/lis.zip>

Find the largest and the second largest number

- 剪枝: 延迟计算第二大数, 维护一个短的候选集



$$C = \{p_2^0 \text{ or } q_2^0, p_2^1 \text{ or } q_2^1, \dots, p_2^{s-1} \text{ or } q_2^{s-1}\}$$

最后一次分治之后,
从C中求得第二大数, 花费 $< \lg n$