Hw 15

Page579: 32.1-2

Page583: 32.2-2，32.2-3

Page587：32.3-2

Page593：32.4-3，32.4-7

32.1

Page579: 32.1-2

32. 1-2 假设在模式 P 中所有字符都不相同。试说明如何对一段 n 个字符的文本 T 加速过程 NAIVE-STRING-MATCHER 的执行速度，使其运行时间达到 $O(n)$。

T[1..n]，P[1..m]

修改后：

```
i = 0
while i<=n-m:
    isMatch = True
    for j = 1..m:
        if T[i+j]!=P[j]:
            # 情况(1)
            isMatch = False
            if j==1:
                # 情况(1.1)
                i = i+1
            else:
                # 情况(1.2)
                i = i+j-1
            break
    if isMatch:
        # 情况(2)
        print('match')
        i = i+m
```

两大类情况，三小种情况：

（1）第i次循环匹配失败，T[i+j]!=P[j]时：

（1.1）j==1时，同naive算法，i往后移一位重新开始比较；

（1.2）j>1时，下次应从T[i+j]开始比较（i跳到i+j-1），因为T[i+k]==P[k]!=P[1]，k=2..j-1。

（2）第i次循环匹配成功时，下次应从T[i+m+1]开始比较（i跳到i+m），因为T[i+k]==P[k]!=P[1]，k=2..m。

综上，最坏情况：T[1]比较1次，T[2..n-m]比较两次，T[n-m+1..n]比较1次，如T=aa..a，P=ab，时间O(n)。

32.2

32.2-2 如何扩展 Rabin-Karp 算法，使其能解决如下问题：如何在文本字符串中搜寻出给定的 k 个模式中的任何一个出现？起初假设所有 k 个模式都是等长的，然后扩展你的算法以适用于不同长度的模式。

Step 1 of 3

Rabin – karp algorithm can extend to search the text string for an occurrence of given set of K – patterns, by taking P as pointer to double dimensional array of size $K \times M$ in its input parameter list So double dimensional array will holds K patterns and each of length M characters .

Step 2 of 3

Now the modified algorithm is

Rabin – karp – matcher (T, P, d, q)

1. n ← length [T]
2. m ← length [P]
3. h ← $d^{m-1} \bmod q$
4. for J ← 1 to K
5. do P ← 0
6. to ← 0
7. for i ← 1 to m
8. do P ← $(dp + P[J][i]) \bmod q$
9. to ← $(dt_0 \text{ to} + T[i]) \bmod q$
10. for s ← 0 to n – m
11. do if P = t_s
12. then if P[J][1...m] = T [s + 1.....s + m]
13. then print "pattern occurs with shift" S
14. if s < n – m
15. then $t_{s+1} \leftarrow (d(t_s - T[s+1]h) + T[s+m+1]) \bmod q$

Step 3 of 3

We can generalize this algorithm to allow the patterns of different lengths by placing 2,3 lines after the 6th line (within outer loop) and before the 7^{th} line so that each time m takes the length of corresponding patter in two dimentional array and continues inner loop m times

32.2-3 试说明如何扩展 Rabin-Karp 算法用于处理以下问题：在一个 $n \times n$ 的二维字符数组中搜索一个给定的 $m \times m$ 的模式。（该模式可以在水平方向和垂直方向移动，但是不可以旋转。）

The Rabin – karp method can extend to handle the problem of looking for a given m × m pattern in an n × n array of characters as follows

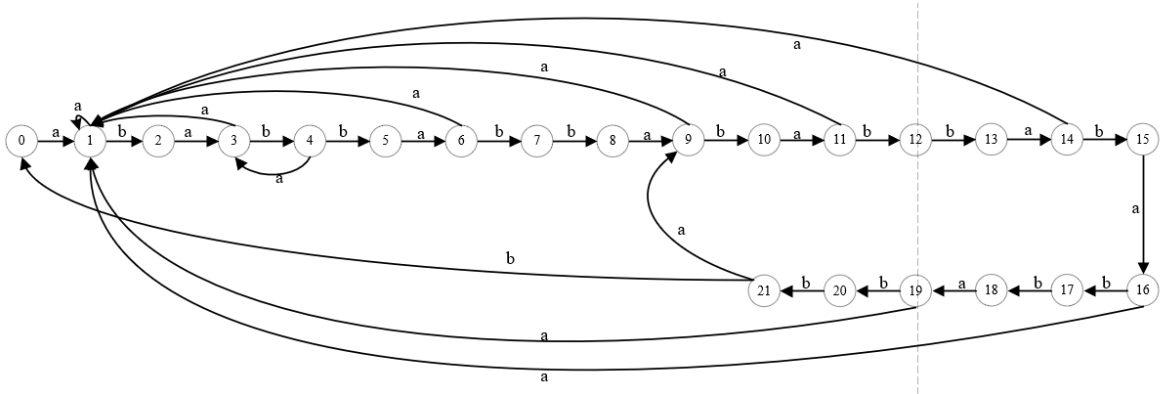
```
Rabin – karp – matcher ( T, P, d, q)
1.  For K ← 1 to n
2.  l_T ← length [T[K]]
3.  For J ←1 to m
4.  l_p ←length [P]
5.  h ← d^(m-1) mod q
6.  P ← 0
7.  to ← 0
8.  for i ← 1 to m
9       do P ← (dp + P[j] [i]) mod q
10          t_0 ← (dt_0 + t [K] [i]) mod q
11. for s ←0 to n – m
12.     do if P = t_s
13.         then if (P[J] [1.....m] = T [K] [S+1.....S+m]
14.                 then print "pattern occurs with shift" S
15.  if S < n – m
16.  then t_(S+1) ← d(t_s – T[K][S+1]h) + T[K][S+m+1] mod q
```

32.3

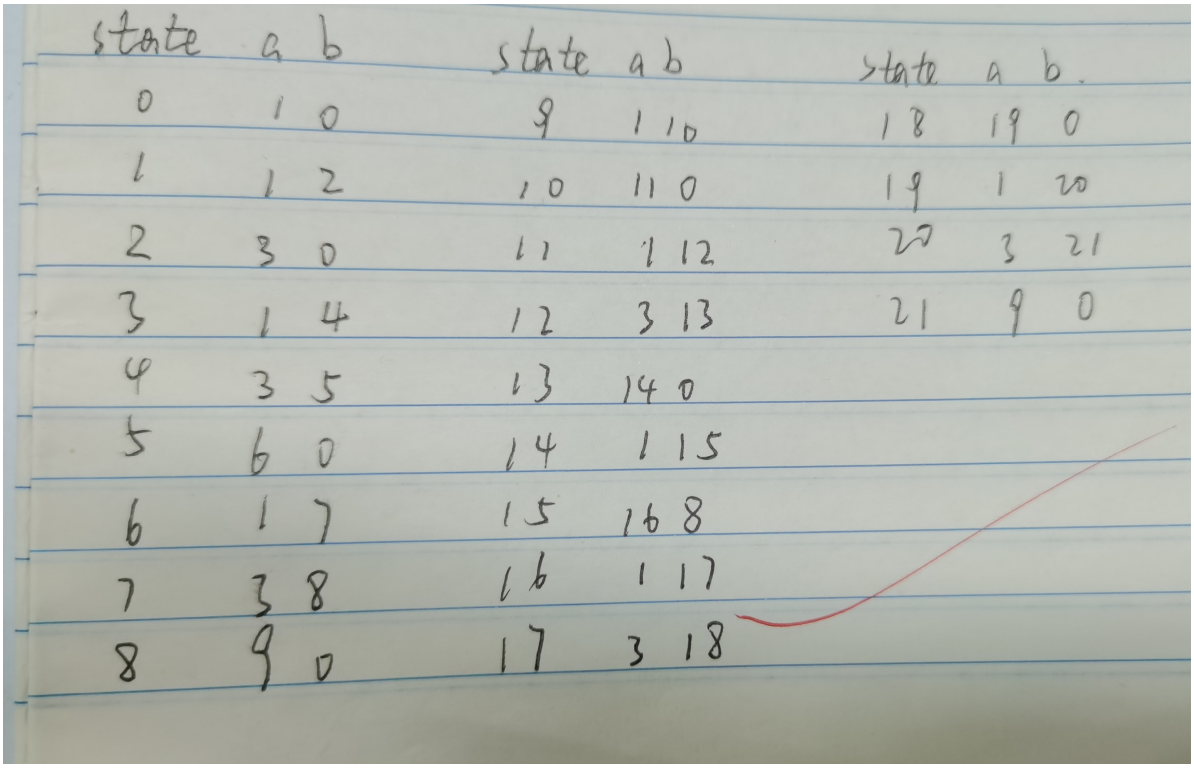
Page587：32.3-2

32. 3-2　对字母表∑＝{a，b}，画出与模式 ababbabbababbababbabb 对应的字符串匹配自动机的状态转换图。

给出图和表均给分。



by#54



32.4

Page593：32.4-3，32.4-7

32.4-3 试说明如何通过检查字符串 PT（由 P 和 T 连结形成的长度为 $m+n$ 的字符串）的 π 函数来确定模式 P 在文本 T 中的出现位置。

答：模式P在文本T中出现的位置集合如下：

只要给出Π[q]或Π*[q]这种均给分。q是PT的下标，这里要确定模式P在文本T中的出现位置，需要减去2m。

$$M = \{q - 2m | m \epsilon \Pi^*[q] \ and \ q \geq 2m\}$$

32.4-7 写出一个线性时间的算法，以确定文本 T 是否是另一个字符串 T' 的循环旋转。例如 arc 和 car 是彼此的循环旋转。

主要思想：将两个文本串T连结形成TT，使用KMP算法判断T'是否是TT的子串，若T'是TT的子串，则T是T'的循环旋转；否则，不是。
算法：

```
Check_rot(string T, string T'):
    if T.length != T'.length:
        return false
    else:
        TT = concat(T,T)
        # 采用KMP算法判断T'是否是TT的子串
        if KMP(TT, T')==true:
            return true
        return false
```