- 书面作业讲解
  - TJ第8章练习6、7、8、9、11、13、18、19、21、22、23

- 教材讨论
  - TC第32章

问题1： naive

- *32.1-4*
  Suppose we allow the pattern P to contain occurrences of a ***gap character*** $\diamond$ that can match an *arbitrary* string of characters (even one of zero length). For example, the pattern ab$\diamond$ba$\diamond$c occurs in the text cabccbacbacab as

  c ab cc ba cba c ab
  ab $\diamond$ ba $\diamond$ c

  and as

  c ab ccbac ba c ab .
  ab $\diamond$ ba $\diamond$ c

  Note that the gap character may occur an arbitrary number of times in the pattern but not at all in the text. Give a polynomial-time algorithm to determine whether such a pattern P occurs in a given text T, and analyze the running time of your algorithm.

- 分段匹配

问题1： naïve (续)

- 我们稍稍改一改题目
  - P occurs in a given text T → P matches T（即必须与整个T匹配）
  - ◊ → ?和*

- 动态规划
  - if: P[i]==T[j] || P[i]=='?'&&T[j]!=EMPTY
    - ans[i-1,j-1]
  - if: P[i]=='*'
    - ans[i,j-1] | ans[i-1,j] | ans[i-1,j-1]

问题2：Rabin-Karp

- 这是对naïve和Rabin-Karp的另一种叙述方式，你能理解吗？与TC相比，你有什么新的收获？

```
1.  function NaiveSearch(string s[1..n], string sub[1..m])
2.     for i from 1 to n-m+1
3.        for j from 1 to m
4.           if s[i+j-1] ≠ sub[j]
5.              jump to next iteration of outer loop
6.        return i
7.     return not found
```

```
1.  function RabinKarp(string s[1..n], string sub[1..m])
2.  hsub := hash(sub[1..m]);  hs := hash(s[1..m])
3.  for i from 1 to n-m+1
4.    if hs = hsub
5.      if s[i..i+m-1] = sub
6.        return i
7.    hs := hash(s[i+1..i+m])
8.  return not found
```

- 你觉得Rabin-Karp效率通常高于naïve的关键步骤是什么？
  - rolling hash
- hash的副作用是什么？副作用有多大？

问题2： Rabin-Karp (续)

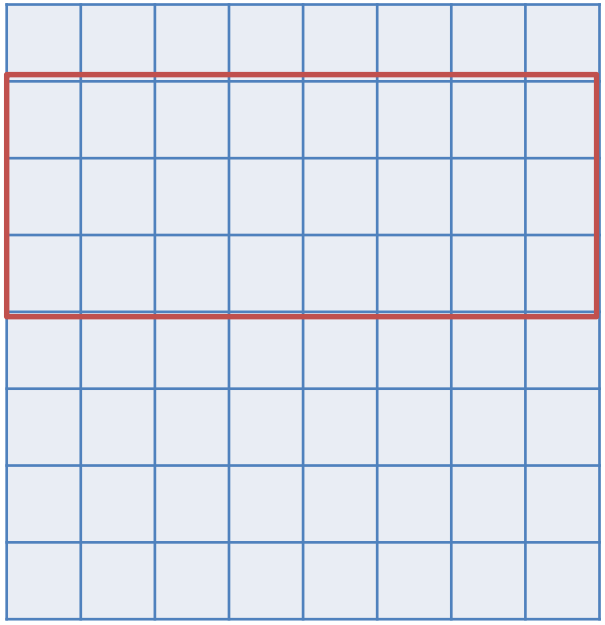
- *32.2-2*
  How would you extend the Rabin-Karp method to the problem of searching a text string for an occurrence of any one of a given set of k patterns? Start by assuming that all k patterns have the same length. Then generalize your solution to allow the patterns to have different lengths.

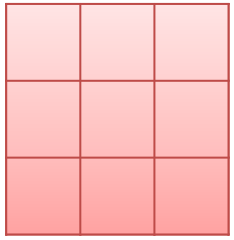
- hash table

问题2： Rabin-Karp (续)

- *32.2-3*
  Show how to extend the Rabin-Karp method to handle the problem of looking for a given $m \times m$ pattern in an $n \times n$ array of characters. (The pattern may be shifted vertically and horizontally, but it may not be rotated.)

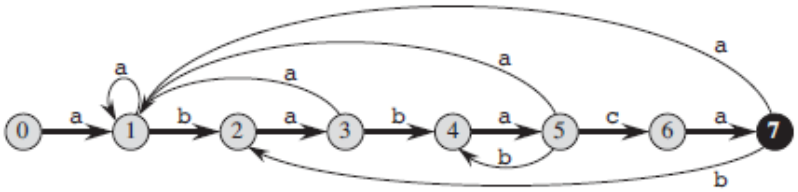
- mxn和mxm → 1xn和1xm → string matching
- rolling hash





问题3：automaton

- 自动机的5个组成部分是什么？



- Q is a finite set of ***states***,
- $q_0 \in Q$ is the ***start state***,
- $A \subseteq Q$ is a distinguished set of ***accepting states***,
- Σ is a finite ***input alphabet***,
- δ is a function from $Q \times \Sigma$ into Q, called the ***transition function*** of M.

- 粗线和细线分别表示什么意思？

- ***32.3-3***
  We call a pattern P ***nonoverlappable*** if $P_k \sqsupset P_q$ implies $k = 0$ or $k = q$. Describe the state-transition diagram of the string-matching automaton for a nonoverlappable pattern.

问题3：automaton (续)

- 这个自动机始终维护的invariant是什么含义？

$$\phi(T_i) = \sigma(T_i)$$

- 用数学归纳法证明上述invariant，你能解释每一个步骤吗？

$$\begin{aligned}
\phi(T_{i+1}) &= \phi(T_i a) && \text{(by the definitions of } T_{i+1} \text{ and } a\text{)} \\
&= \delta(\phi(T_i), a) && \text{(by the definition of } \phi\text{)} \\
&= \delta(q, a) && \text{(by the definition of } q\text{)} \\
&= \sigma(P_q a) && \text{(by the definition (32.4) of } \delta\text{)} \\
&= \sigma(T_i a) && \text{(by Lemma 32.3 and induction)} \\
&= \sigma(T_{i+1}) && \text{(by the definition of } T_{i+1}\text{)}
\end{aligned}$$