3.1.1

<float> <id, limitedSquaare> <(> <id, x> <)> <{>

<float> <id, x>

<return> <(> <id, x> <op,"<="> <num, -10.0> <op, "||"> <id, x> <op, ">="> <num, 10.0> <)> <op, "?"> <num, 100> <op, ":"> <id, x> <op, "\*"> <id, x>

<}>

3.1.2

<text, "Here is a photo of"> <nodestart, b> <text, "my house"> <nodeend, b>

<nodestart, p> <selfendnode, img> <selfendnode, br>

<text, "see"> <nodestart, a> <text, "More Picture"> <nodeend, a>

<text, "if you liked that one."> <nodeend, p>

3.3.2

1)以a开头和结尾,中间有a或b二者中一个或者多个

2)以a或者空开头后是0或多个b的重复包括0次以上的字符串

3)长度为3或者4,第二个字符为a,其余可是a和b的字符串

4)bbb中每个位置至多插入一个a

5)aa或bb或空开头,后可跟ab或ba开头aa或bb结尾的子串多次

3.3.3

1)前缀 n+1

2)后缀 n+1

3)真前缀 n

4)子串 [n\*(n+1)]/2

5)子序列 ∑(i=1,n)C(n,i)

3.3.4

[sS][eE][lL][eE][cC][tT]

3.3.5

1)

Str -> other\* a\* (other|a)\* e\* (other|e)\* i\* (other|i)\* o\* (other|o)\* u\* (other|u)\*

Other -> [^aeiou]

2)

a\*b\*c\*……….z\*

3)

/\\*([^\\*\”]|\”[^\”]\*\”)\*\\*/

4)

want -> 0|A?0?1(A0?1|01)\*A?0?|A0?

A -> 0?2(02)\*

5)

want -> (FE\*G|(aa)\*b)(E|FE\*G)

E -> b(aa)\*b

F -> a(aa)\*b

G -> b(aa)\*ab|a

F -> ba(aa)\*b

6)

(aab|aba|baa)\*

7)

8)

9)

3.3.6

1) [a-jA-J]

2) [bcd....除去aeiou]

3) [0-9A-F]

4) [.?!]

3.3.7

\”\\

3.3.9

r{m,n}等价于r(m个)|r(m+1个)....

3.3.10

1) 位于中括号内第一个是补作用

2） 可以

3.4.1

1) a(a|b)\*a

![3 4 1-1-nfa](data:None;base64,)

2) ((ε|a)b\*)\*

![C:\Users\acer\Desktop\未标题-1.png](data:None;base64,)

3) (a|b)\*a(a|b)(a|b)

![C:\Users\acer\Desktop\图表1.png](data:None;base64,)

4) a\*ba\*ba\*ba\*

![3 4 1-4](data:None;base64,)

3.4.3

1) abababaab

失效函数: 0,0,1,2,3,4,5,1,2

2) aaaaaa

0,1,2,3,4,5

3) abbaabb

0,0,0,1,1,2,3

3.4.4:

#### 证明

1. 已知 f(1) = 0
2. 在第 1 次 for 循环时，计算 f(2) 的值，当第5行代码 b\_2 == b\_1 成立时，代码进入到第7行得出 f(2) = 1，不成立时，则代码进入第9行得出 f(2) = 0。显然，这次循环正确的计算出了 f(2) 。
3. 假设在第 i-1 次进入循环时，也正确的计算出了 f(i)，也有 f(i) = t (无论 t 是大于 0 还是等于 0)
4. 那么在第 1 次进入循环时，分两种情况进行考虑：
   1. t == 0

这种情况比较简单，直接从第 5 行开始，当 b\_i+1 == b\_1 时，f(i+1) = 1，否则 f(i+1) = 0

* 1. t > 0 while 循环会不断缩小 t 值，试图找出最大可能的使得 b\_i+1 == b\_t+1 成立的 t 值，如果找到了，则进入第 5 行执行，得到 f(i+1) = t+1；或者直到 t == 0 时也没有找到，则跳出循环，这时进入第 5 行执行，过程类似于前一种情况。

3.4.5

当为aaa..aaaaaaaaa时即成立

3.4.6

1) true 2) false

3.4.7

1) a b a b a a

0 0 1 2 3 1

abababaab

|

ababaa

abababaab

|

ababaa

2) abababbaa

|

ababaa

abababbaa

|

ababaa

abababbaa

|

ababaa

3.4.8

因此，整个过程中j最多加了n个1。于是，j最多只有n次减小的机会（j值减小的次数当然不能超过n，因为j永远是非负整数）。这告诉我们，while循环总共最多执行了n次。按照摊还分析的说法，平摊到每次for循环中后，一次for循环的复杂度为O(1)。整个过程显然是O(n)的。这样的分析对于后面P数组预处理的过程同样有效，同样可以得到预处理过程的复杂度为O(m)。

3.4.9

1) a_{n}=\frac{\sqrt{5}}{5} \cdot \left[\left(\frac{1 + \sqrt{5}}{2}\right)^{n} - \left(\frac{1 - \sqrt{5}}{2}\right)^{n}\right]

2) s6 = abaababa

failure = [ 0, 0, 1, 1, 2, 3, 2, 3 ]

1. s7 = abaababaabaab

failure = [ 0, 0, 1, 1, 2, 3, 2, 3, 4, 5, 6, 4, 5 ]

4)

5) n次

3.4.11

1)



3.4.12

建树o(n), 失效o(n) 查找o(n) 故为o(n)

3.5.1

1) while {return (WHILE);}

2) “!=”{yylval=NE; return(RELOP);}

3) letter[a-zA-Z\_]

4) String {“.\*”}

{String} {yylval=(int) installstr(); return(STRING);}

int installstr(){

替换转义字符

放入符号表返回指针

}

3.5.2~5

github.com/qq53