My AUA ID is A09170218

Question 4:

1. Basic Idea: Push x=8 symbols for each ‘a’. We read and pop one for each ‘b’.

Let Q = {s, q1, q2, q3, q4, q5, q6, q7, t, f} be the states

∑ = {a, b} be the input alphabets

s from Q be start state.

F = {s, f} accept states.

………….

The transition relation is

((s, a, ε), (q1, a))  
((qi, ε, ε), (qi+1, a)) for each i from {1, 2, 3, 4, 5, 6}  
((q7, ε, ε), (t, a))  
((t, a, ε), (q1, a))  
((t, b, a), (f, ε))

((f, b, a), (f, ε))

1. Very similar to a)  
   We will push 1 ‘a’ each time we read ‘a’ and pop 8 for each ‘b’.

The relation is as follows

((s, a, ε), (t, a))

((t, a, ε), (t, a))

((t, b, a), (q1, ε))  
((qi, ε, a), (qi+1, ε)) for each i from {1, 2, 3, 4, 5, 6}  
((q7, ε, a), (f, a))

((f, b, a), (q1, ε))

1. For L\_a a grammar is:

G = (V, ∑, R, S)  
where V={S} is the non-terminal symbols and ∑={a, b} is the terminal symbols and S is the start variable.

The rules are:  
S 🡪 a S bbbbbbbb  
S 🡪 ε

1. For L\_b a grammar is:

G = (V, ∑, R, S)  
where V={S} is the non-terminal symbols and ∑={a, b} is the terminal symbols and S is the start variable.

The rules are:  
S 🡪 aaaaaaaa S b  
S 🡪 ε

……………………………………………………..

Question 5:

Part 1:

There are 6 possible problems in the 3-peg Hanoi Towers.

1. Appling the algorithm for special case of 1 disc:

to the special case of 1 disc:

1 disc from 0 to 1 🡪 print 01

1 disc from 0 to 2 🡪 print 02

1 disc from 1 to 0 🡪 print 10

1 disc from 1 to 2 🡪 print 12

1 disc from 2 to 0 🡪 print 20

1 disc from 2 to 1 🡪 print 21

1. Appling the algorithm for special case of N discs:

N discs from 0 to 1 🡪 (N – 1) discs from 0 to 2; print 01; (N – 1) discs from 2 to 1

N discs from 0 to 2 🡪 (N – 1) discs from 0 to 1; print 02; (N – 1) discs from 1 to 2

N discs from 1 to 0 🡪 (N – 1) discs from 1 to 2; print 10; (N – 1) discs from 2 to 0

N discs from 1 to 2 🡪 (N – 1) discs from 1 to 0; print 12; (N – 1) discs from 0 to 2

N discs from 2 to 0 🡪 (N – 1) discs from 2 to 1; print 20; (N – 1) discs from 1 to 0

N discs from 2 to 1 🡪 (N – 1) discs from 2 to 0; print 21; (N – 1) discs from 0 to 1

Now we introduce 6 variables:

A = from 0 to 1

B = from 0 to 2

C = from 1 to 0

D = from 1 to 2

E = from 2 to 0

F = from 2 to 1

Finally, plugin the introduced variables in the algorithm above omitting “print” and the indicated numbers of discs:

|  |  |  |
| --- | --- | --- |
| *A* → 01  *B* → 02  *C* → 10  *D* → 12  *E* → 20  *F* → 21 | *A* → *B*01*F B* → *A*02*D C* → *D*10*E D* → *C*12*B E* → *F*20*C F* → *E*21*A* | |
| Thus, the required context-free grammar becomes *G* = {Σ, *NT*, *R*, *S*}, where | |
| Σ = {0, 1, 2}; | |  | |

*NT* = {*A*, *B*, *C*, *D*, *E*, *F*};  
*R* = { *A* → 01, *A* → *B*01*F*, *B* → 02, *B* → *A*02*D*, *C* → 10, *C* → *D*10*E*, *D* → 12, *D* → *C*12*B*, *E* → 20, *E* → *F*20*C*, *F* → 21, *F* → *E*21*A*};  
*S* ∈ *NT* depends on the initial *i* and final *k* pegs specified by the ID#.

Based on my id that is A09170218 🡺

i = 28 % 3 = 1 and k is (i + 2) % 3 = 0 🡺

The start symbol is S = { N discs from 1 to 0 } == { *C* → 10 }

……………………………………….

Part 2:

Let \*\* mean to the power of.

It is easy to see from the requirement of the synchronous parsing of all non-terminals that  
*N* discs are moved in 2\*\**N* – 1 steps. Indeed, the number of non-terminals is doubled with  
each application of a rule of type *X* → *YabZ*, and there is exactly one pair of terminals  
between each pair of them. Therefore, the tot al number of non-terminals and terminal pairs  
is 2\*\**N* – 1.

To identify a specific step there is no need to derive all the moves (their quantity is

exponentially large). The total number of moves 2\*\*N – 1 = 2\*\*N–1 + 1 + 2\*\*N–1, where with the

middle move the largest disc is transferred from the initial peg to the final one. Therefore,

it corresponds to the base case. Formally, given a problem for N discs X  YikZ, where i

and k are the initial and the final pegs respectively, the requested move m can be identified

as following:

- If m = 2\*\*N–1 and, therefore, m is the middle move in X, then m = ik;

- If m < 2\*\*N–1, then it is also the mth move in Y, which corresponds to N – 1 discs and,

therefore, requires only N – 1 derivation steps;

- If m > 2\*\*N–1, then it is also the (m – 2\*\*N–1)th move in Z, which corresponds to N – 1 discs

and, therefore, requires only N – 1 derivation steps;

Part 3:

Let’s assume the discs are transferred from 0 to 2 ⇒ the starting nonterminal *S* = *B* in the  
grammar. The construction of the equivalent pushdown automaton *A* = {*Q*, Σ, Γ, *s*, Δ, *F*}  
is straightforward:  
*- Q* = {*s*, *f*}, *F* = {*f*}  
*-* Σ = {0, 1, 2}  
*-* Γ = {0, 1, 2, *A*, *B*, *C*, *D*, *E*, *F*}  
*-* Δ:

1. Transitions that implement the rules

|  |
| --- |
| 1. ((*s*, *e*, *e*), (*f*, *B*)) 2. ((*f*, *e*, *A*), (*s*, 01)) 3. ((*f*, *e*, *A*), (*s*, *B*01*F*)) 4. ((*f*, *e*, *B*), (*s*, 02)) 5. ((*f*, *e*, *B*), (*s*, *A*02*D*)) 6. ((*f*, *e*, *C*), (*s*, 10)) 7. ((*f*, *e*, *C*), (*s*, *D*10*E*)) 8. ((*f*, *e*, *D*), (*s*, 12)) 9. ((*f*, *e*, *D*), (*s*, *C*12*B*)) 10. ((*f*, *e*, *E*), (*s*, 20)) 11. ((*f*, *e*, *E*), (*s*, *F*20*C*)) 12. ((*f*, *e*, *F*), (*s*, 21)) 13. ((*f*, *e*, *F*), (*s*, *E*21*A*)) |

1. Transition when reading the terminals

14. ((f, 0, 0), (s, e))

15. ((f, 1, 1), (s, e))

16. ((f, 2, 2), (s, e))