Wednesday, May 11, 2022

Assignment 9

CSCI 411

1.

$$0^* (0 + 101)^* + (110)^*$$

Let $A = 0^* (0 + 101)^*$ and $B = (110)^*$

Reverse of A $= \varepsilon 0^* + \varepsilon[(101)^* + 0^*]\, \varepsilon 0^*$

$= 0^* + [(101)^* + 0^*]\, 0^*$

Reverse of B $= (011)^*$

Hence, the reverse is A + B, **$0^* + [(101)^* + 0^*]\, 0^* + (011)^*$.**

2.

$$0^* + (11 + 01 + 10 + 00)^*$$

Let $A = 0^*$ and $B = (11 + 01 + 10 + 00)^*$

Complement of A $= 0^*\, 1(0 + 1)^*$

Complement of B $= (0 + 1)\, (00 + 01 + 10 + 11)^*$

Hence, the reverse is A + B, **$0^*\, 1(0 + 1)^* + (0 + 1)\, (00 + 01 + 10 + 11)^*$.**

3.

Outcome - {0, 1}*

S → S_1 1S_2

S1 → 0S_0 1 | 1 S_1 0 | S_1 S_1 | 1 S_1 |

4.

S → AB | BC | a

A → CA | c

B → AA | BB | a | c

C → AB | b

a. bccbcac

A, A, A, A, S, S, S, S, C, C, C, C, S, A, A, S, S, C, C, A, A, A						
S, C, S, S, S, C, C	S, S, S, S, B, B, B, B, C, C, C, C, S, S, A, A, S, B, C, A, A, B, B					
A, S, C, A	S, B, C, S, S, S, B, C	S, B, B, B, B				
S	S, B, C, A	S, B	A, S, C, A			
S, B, C	S, S	B	S, C	S, B, C, A, B		
A	S, B, B, C	S	A	S, B, C	B	
C	A, B	A, B	C	A, B	S, B	A, B
b	**c**	**c**	**b**	**c**	**a**	**c**

CFG recognizes the string because the top cell is non-empty.

b. ccaabb

S, S, S, S, S, S, S					
S, S, S, B, B, C, C, B, B, B, B, B	S, S				
S, S, B, C, B, B	S, B, C, B	S			
S, B, B, C	S, B, C	B	S		
A, B	A, B	S, B	S, B	C	C
c	**c**	**a**	**a**	**b**	**b**

CFG does not recognizes the string because the top cell is empty.

5.

S → AA | BB | BA | ε

A → AB | 1

B → BA | 0

a. 0001110

	S					
	S	S, B, S				
	S	S, B				
	S	S, B		S		
S	S	S, B	S	S	A	
B	B	B	A	A	A	B
0	**0**	**0**	**1**	**1**	**1**	**0**

CFG does not recognizes the string because the top cell is empty.

b. 1011000

A, S, A, A, A						
A, S, A, A	S, B, S					
A, S, A	S, B, S	S				
A, S	S, B, S	S	A			
A, S	S, B	S	A			
A	S, B	S	A	S	S	
A	B	A	A	B	B	B
1	**0**	**1**	**1**	**0**	**0**	**0**

CFG recognizes the string because the top cell is non-empty.

6.

a. ababa

q_0 → q_1	B a b a b a B B
q_2 → q_2	B X b a b a B B
q_3 → q_7	B X b a b a X B
q_7 → q_7	B X b a b X B B
q_7 → q_0	B X b a b X B B
q_4 → q_5	B X X a b X B B
q_5 → q_6	B X X a b X B B
q_7 → q_7	B X X a X X B B
q_0 → q_1	B X X X X X B B
	q_8 is final state

Hence, ababa is accepted.

b. babb

$q_0 \rightarrow q_4$	B B X a b b B B
$q_5 \rightarrow q_5$	B B X a b b B B
$q_5 \rightarrow q_6$	B B X a b b B B
$q_7 \rightarrow q_0$	B B X a b X B B
$q_1 \rightarrow q_2$	B B X X b X B B
q_3	B B X X b X B B
	Transition to b not found.

Hence, babb is rejected.

7.

Sudokus can be solved by computers in O(1) time for any valid fixed input. One of the attributes of NP-complete is that it has variable input size so one can analyze the running time of an algorithm as that input size grows asymptotically.

Sudoku runtime depends on the grid size. So as the grid size increase , the time will grow exponentially because DFS has to go deeper and deeper, as well as having to consider more possibilities at each DFS level.

8.

According to the theorum, if G = (V, E) is a graph, then S is an independent set

$\Rightarrow$ V – S is a vertex cover

Suppose S is an independent set, and let e = (u, v) be some edge. Only one of u, v can be in S. Hence, at least one of u, v is in V – S. So, V – S is a vertex cover.

Suppose V – S is a vertex cover, and let u, v $\in$ S. There can't be an edge between u and v (otherwise, that edge wouldn't be covered in V – S). So, S is an independent set.

By our previous theorem, S is an independent set iff V – S is a vertex cover: then S must be an independent set of size ≥ k.

9.

Using vertex cover,

$$G = (V, E) \text{ is a subset } V' \subseteq V$$

We can reduce any instance of decision-vertex-cover for G = (V, E) to an instance of decision-set-cover by letting U = E and letting the family of sets equal to S_v where v belongs to V, and such that S_v contains all edges in E adjacent to v. Hence, we can dissolve this reduction in O(|E| + |V|) polynomial time.

Since, U = E we can find set cover for U and family of sets by selecting family from family of all sets corresponding to v' belonging to V'. This simultaneously proves that we can find V' by taking vertices corresponding to S_v belonging to vertex cover if the vertex cover exists.

10.

Knapsack problem:

Given a knapsack with a weight limit of W, a collection of n items x_1, x_2, x_3…x_n with values v_1, v_2, v_3…v_n and weights w_1, w_2, w_3…w_n. Find the maximum value of the items that can be added to the knapsack such that the weight does not exceed the weight limit W.

Input:

Length of x = log (n)

Length of W = log (W)

each x for w[i] size log (W[i])

each x for v[i] size log (v[I])

This problem can decomposed into Subset-sum problem for a boolean solution. And we know that subset-sim problem is NP-Complete. Therefore, the knapsack problem can be reduced to the Subset-Sum problem in polynomial time.