ASSIGNMENT4 QUESTION1

1. Boolean operators NAND and NOR are defined as follows

| NAND | *true* | *false* |
|---|---|---|
| *true* | *false* | *true* |
| *false* | *true* | *true* |

| NOR | *true* | *false* |
|---|---|---|
| *true* | *false* | *false* |
| *false* | *false* | *true* |

You are given a boolean expression consisting of a string of the symbols *true*, *false*, separated by operators AND, OR, NAND and NOR but without any parentheses. Count the number of ways one can put parentheses in the expression such that it will evaluate to *true*. (20 pts)

**Answer:**

We have a Boolean expression consisting of a string [i:j], and for every part of string, we want to know the numbers of way that it be true and false.

Let T[i , j] be the number of ways of parenthesizing between the string [i : j] evaluates to TRUE.

F[i , j] be the number of ways of parenthesizing between between the string [i : j] evaluates to False. Include I j. (I $\leq$ j)

Then the total[i , j] = T[i, j] + F[i , j]

T[i , j] and F[i , j] can be expressed by the following recursive formulations:

$$T[i,j] = \sum_{k=i}^{j-1} \begin{cases} T[i,k] \cdot T[k+1,j] & if\ S[k] = AND \\ \text{total}[i,k] \cdot \text{total}[k+1,j] - F[i,k] \cdot F[k+1,j] & if\ S[k] = OR \\ \text{total}[i,k] \cdot \text{total}[k+1,j] - T[i,k] \cdot T[k+1,j] & if\ S[k] = NAND \\ F[i,k] \cdot F[k+1,j] & if\ S[k] = NOR \end{cases}$$

$$F[i,j] = \sum_{k=i}^{j-1} \begin{cases} \text{total}[i,k] \cdot \text{total}[k+1,j] - T[i,k] \cdot T[k+1,j] & if\ S[k] = AND \\ F[i,k] \cdot F[k+1,j] & if\ S[k] = OR \\ T[i,k] \cdot T[k+1,j] & if\ S[k] = NAND \\ \text{total}[i,k] \cdot \text{total}[k+1,j] - F[i,k] \cdot F[k+1,j] & if\ S[k] = NOR \end{cases}$$

And we have this base case:

T[i , i] = 1 if S[i] = True

T[i , i] = 0 if S[i] = False

F[i , i] = 0 if S[i] = True

F[i , i] = 1 if S[i] = False

The final answer will be given by T[1,n] since we use all string [1:n]. and we want the string evaluate to true.

The time complexity for this question should be $O(n^3)$ since we have $O(n^2)$ sub-problems and $O(n)$ to solve each problems .