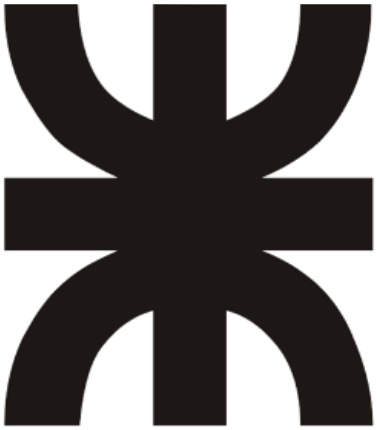


UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL RESISTENCIA

Ingeniería en Sistemas de Información

Materia: Complejidad y Técnicas de Diseño de Algoritmos

Trabajo Práctico Integrador

Grupo N° 6:

- ***Maldonado Leandro Arian***
- ***Ojeda Delio Brian Bautista***
- ***Schefer Mauricio Nicolas***
- ***Velazco Gez Schegtel Juan Ignacio***

“Paréntesis de Expresiones Booleanas”

El problema que vamos a encarar es la “parentesización” de expresiones booleanas (también conocido simplemente como “Evaluación Booleana”), consta de encontrar todas las distintas maneras de agregar paréntesis a una proposición compuesta por operandos verdaderos y falsos, con operadores de conjunción (AND), disyunción (OR) y disyunción exclusiva (XOR) de manera que la expresión sea verdadera.

Supongamos que tenemos una proposición de la forma **P=(T|F)((AND|OR|XOR)(T|F))$^{n-1}$**, es decir, una proposición de n operandos, con n >= 1, todos ellos separados por operadores lógicos.

Definimos dos subconjuntos:
Por un lado, tenemos los **T(i,j)**, conjunto de soluciones para una sub proposición dentro de la proposición original, desde el elemento i-ésimo hasta el j-ésimo, de manera que esa subexpresión sea verdadera.
A su vez, también tenemos los **F(i,j)**, el conjunto de soluciones para una sub proposición dentro de la proposición original, desde el elemento i-ésimo hasta el j-ésimo, de forma que esa subexpresión sea falsa.

Nuestro objetivo es encontrar todos los **T(i,j)** de manera que obtengamos el conjunto **T(1,n)**, que contiene todas las posibles formas de agregar paréntesis a nuestra proposición de longitud n de manera que esta sea verdadera.

Por ejemplo, consideremos la expresión:

T *OR* T *AND* F *XOR* T

Para esta expresión, tenemos cuatro formas de agregar paréntesis de manera que la expresión sea verdadera:

T *OR* ((T *AND* F) *XOR* T)
T *OR* (T *AND* (F *XOR* T))
(T *OR* T) *AND* (F *XOR* T)
((T *OR* T) *AND* F) *XOR* T

Y una forma de agregar paréntesis para que la expresión sea falsa

(T *OR* (T *AND* F)) *XOR* T

Para facilitar la solución del problema, estos conjuntos **T(i,j)** y **F(i,j)** se pueden descomponer, cada uno, en varios otros conjuntos de soluciones para particiones más pequeñas de la proposición original. Supongamos que una partición **P(i,j)** se puede separar en dos sub proposiciones de la forma:

P(i,j) = **P(i,k)** * **P(k,j)**

donde el asterisco representa un operador lógico. En función de cuál sea el operador que una las sub proposiciones, los conjuntos **T(i,j)** y **F(i,j)** se pueden simplificar de distintas formas

Hipótesis

Dado un cadena de 4 valores de verdad obtendremos la cantidad de configuraciones diferentes de paréntesis que permiten que dicha cadena de como resultado verdadero. Se espera que la resolución utilizando programación dinámica sea más rápida que la técnica de backtracking.

Resolución con Programación Dinámica

Para esta técnica, se arman las matrices True y False, poniendo en las filas y columnas los valores de verdad. Cada uno de los elementos de las matrices triangulares representa una combinación de valores de verdad, y el número en cada casilla representa el número de combinaciones de paréntesis que le dan a la expresión valor verdadero (o falso para la matriz F).

Matriz T (Cuenta de True):

1	0	2	5
0	0	1	2
0	0	1	1
0	0	0	0

Matriz F (Cuenta de False):

0	1	0	0
0	1	0	0
0	0	0	0
0	0	0	1

Expresión de entrada: True AND False OR True XOR False
Número de formas de parentizar: 5
(True AND (False OR (True XOR False)))
(True AND ((False OR True) XOR False))
((True AND False) OR (True XOR False))
((True AND (False OR True)) XOR False)
(((True AND False) OR True) XOR False)

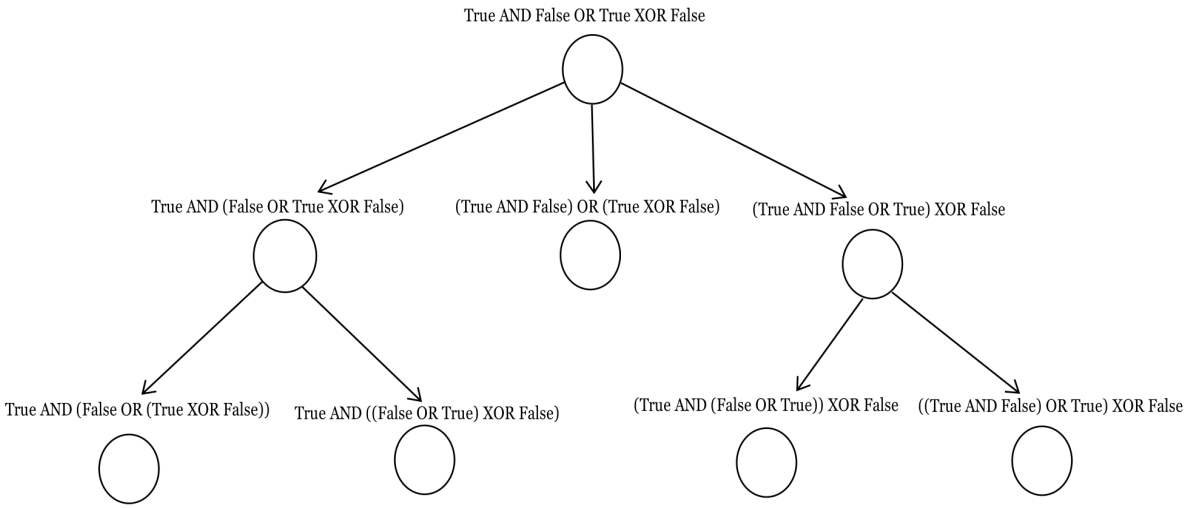
Tabla para combinaciones que devuelven Verdadero

	AND	OR	XOR	
TRUE	T	F	T	F
T	T=1	T&F = 0	T&F|T = 2	T&F|T^F = 5
F	0	F=0	F|T =1	F|T^F = 2
T	0	0	T=1	F^T =1
F	0	0	0	F=0

Como se ve, las expresiones analizadas aumentan en longitud a medida que nos movemos de izquierda a derecha, de abajo hacia arriba.

Resolución con Backtracking

En la técnica de backtracking, partimos de la expresión completa, y vamos dividiendo en expresiones de dos en dos más pequeñas, hasta agotar las opciones. Al llegar a cada opción final de configuración de paréntesis, se analiza su valor de verdad y se devuelve el resultado, sumando ese resultado al conjunto de soluciones de ser verdadero.



Conclusiones

Encontramos que Backtraking es una buena forma de entender y reflexionar acerca del programa, al poder apoyarnos en un método visual para confeccionar o encontrar las posibles soluciones, con ayuda de un árbol. Sin embargo, en expresiones muy extensas

esta técnica no es eficiente y allí es cuando utilizar la programación dinámica se convierte en una necesidad.

Con respecto a la elección de las técnicas, decidimos que no íbamos a abordar este problema con Algoritmos Voraces porque nuestro objetivo es encontrar todas las posibles configuraciones de paréntesis para evaluar la expresión *booleana,* y los Algoritmos Voraces o *Greedy* están diseñados para encontrar solamente una solución.