Lab Assignment № 3

Data Structures & Algorithms (CS F211) 2nd Semester 2015-16

Problem

QBF Game Tree Evaluation

Input

You will be given a Boolean formula with $2n$ variables in CNF (Conjunctive Normal Form). The variables will be from the capital alphabet. The first variable selected by Player 1 will always be A. The second variable selected by Player 2 will always be B, and so on. We will always have the variables as the initial segment of the alphabet and the variables will follow the alphabetical order. For example, we can have the variables A, B or A, B, C, D, but not A, C or B, C, D, E. The language cannot take $\wedge$, $\vee$ or $\neg$ symbols as input. So, instead of these symbols we will use the familiar symbols: $*$ for $\wedge$ (AND), $+$ for $\vee$ (OR) and $-$ for $\neg$ (NOT). For example,

$$(A) \wedge (B \vee \neg C)$$

will be represented as

$$(A) * (B + -C)$$

A CNF formula f of $2n$ variables $x_1, x_2, ..., x_{2n}$ can be represented as follows:

$$f(x_1, x_2, ..., x_{2n}) = (C_1) \wedge (C_2) \wedge ... \wedge (C_m)$$

where each $C_k (1 \leq k \leq m)$ is a clause of the form

$$y_{k_1} \vee y_{k_2} \vee ... \vee y_{k_i}$$

where each $y_{k_j} (1 \leq j \leq i)$ is either a variable x_{k_j} or a negation of a variable $\neg x_{k_j}$, both without parentheses. The formula will have exactly m opening parentheses $'('$ and exactly m closing parentheses $')'$ as shown above. There will be no extra parentheses.

Some examples of formulas in CNF form and their equivalent input forms are:

$$(A) \wedge (B \vee \neg C) \equiv (A) * (B + -C)$$

$$(\neg A) \wedge (C) \wedge (\neg B \vee D) \equiv (-A) * (C) * (-B + D)$$

The input will always be in a correct form and a valid CNF formula with $2n$ variables. You are not required to check it for correctness.

Tasks

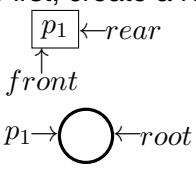
After taking the input, you will have to:

1. Construct the game tree equivalent to the input formula,
2. Evaluate the game tree, and
3. Output the evaluation of each node of the game tree in preorder (on a single line without spaces).

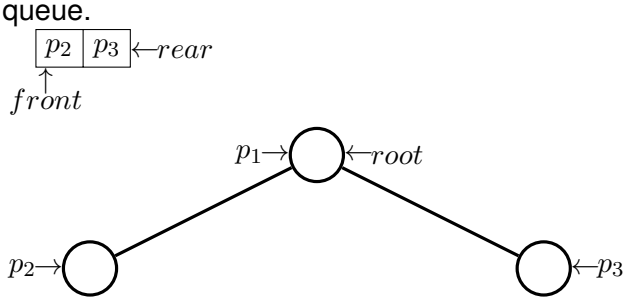
Constructing the Game Tree

You can use Level Order Traversal using a queue to construct a game tree. Example: Suppose the input is $(A + B)$.

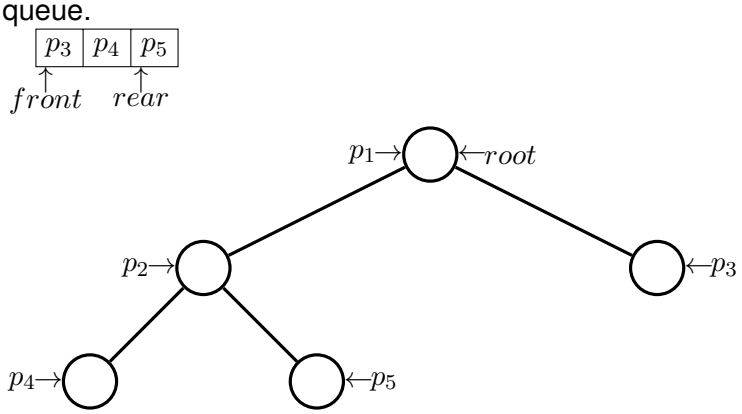
First, create a root node and insert its pointer in a queue.



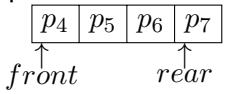
Remove p_1 from the queue, make its two children and insert the pointers to the children in the queue.

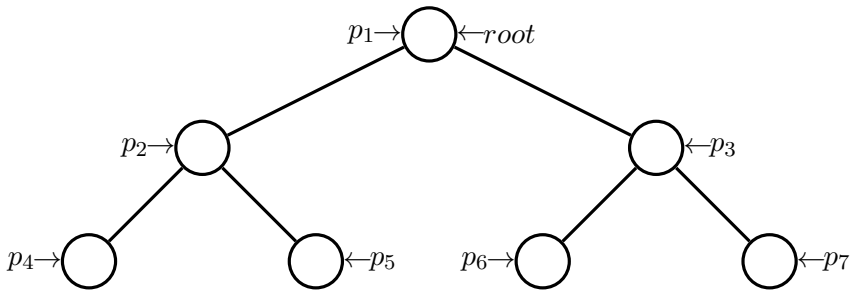


Remove p_2 from the queue, make its two children and insert the pointers to the children in the queue.



Remove p_3 from the queue, make its two children and insert the pointers to the children in the queue.



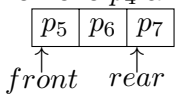


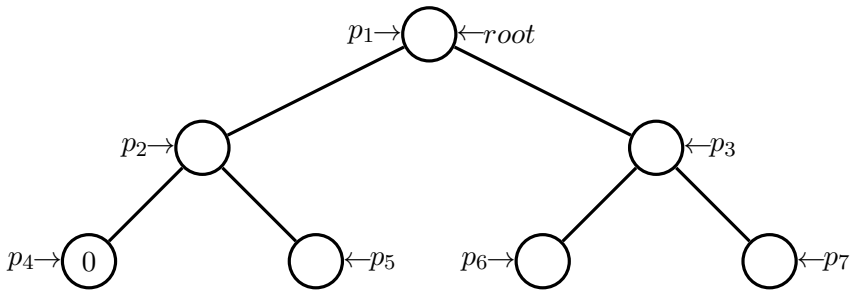
Now, the tree is constructed.

Evaluating the Game Tree

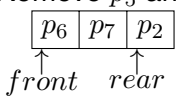
Now you can use Reverse Level Order Traversal using the same queue (with p_4, p_5, p_6 and p_7) to evaluate the game tree.

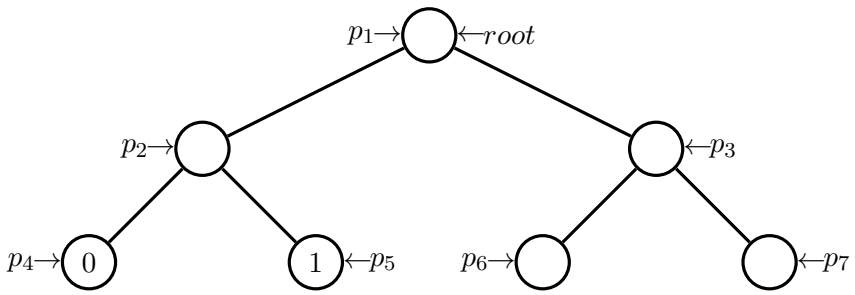
Remove p_4 and evaluate it. p_4 is a left child, so we continue.



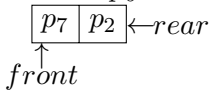


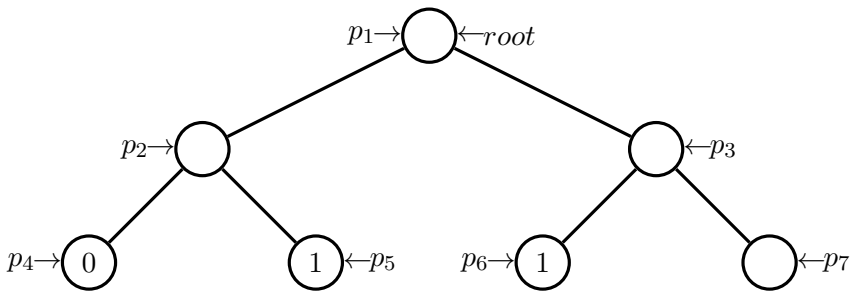
Remove p_5 and evaluate it. p_5 is a right child. So, we insert its parent p_2 in the queue.



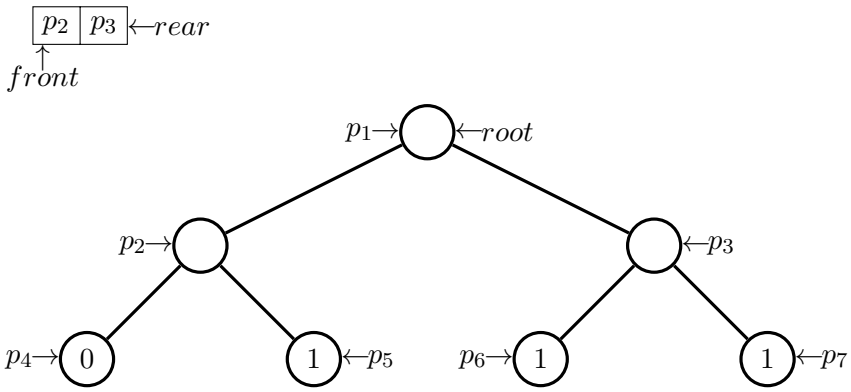


Remove p_6 and evaluate it. p_6 is a left child, so we continue.

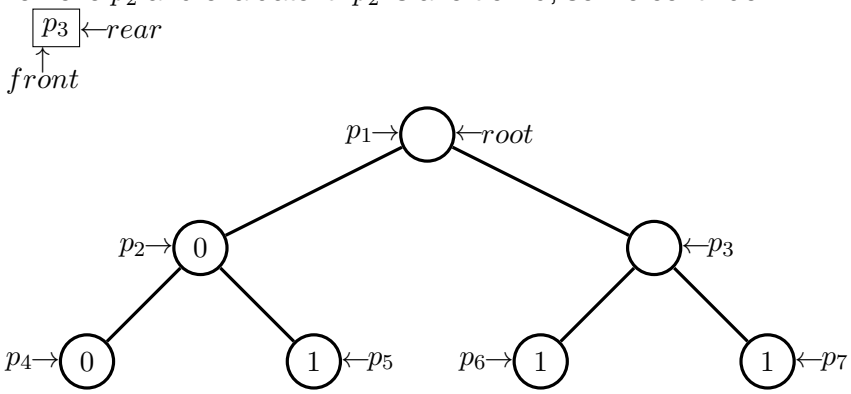




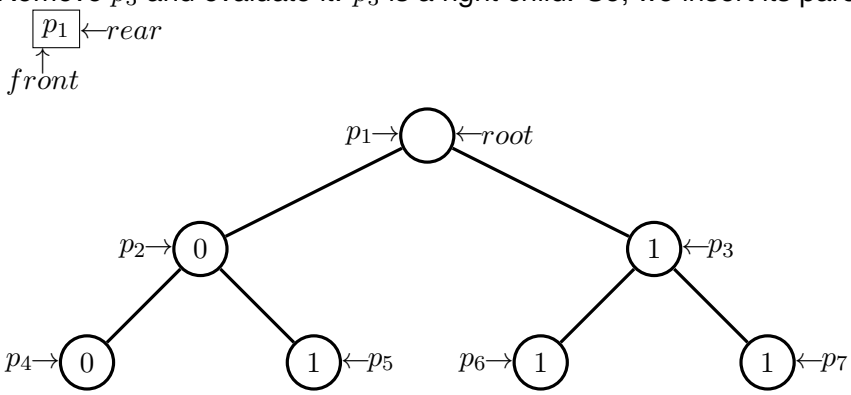
Remove p_7 and evaluate it. p_7 is a right child. So, we insert its parent p_3 in the queue.



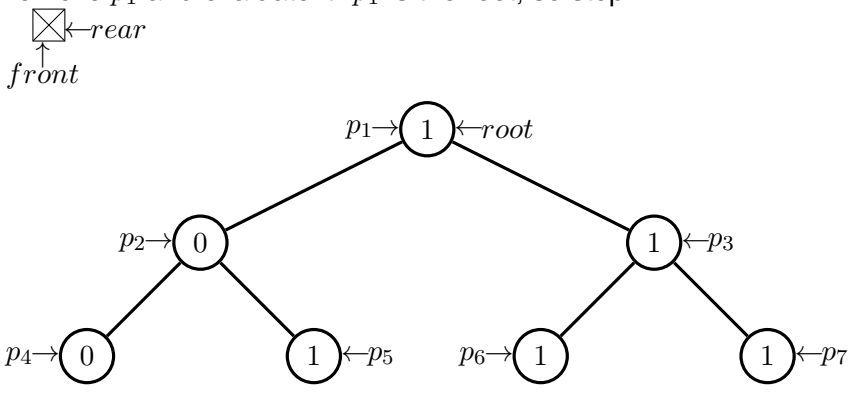
Remove p_2 and evaluate it. p_2 is a left child, so we continue.



Remove p_3 and evaluate it. p_3 is a right child. So, we insert its parent p_1 in the queue.



Remove p_1 and evaluate it. p_1 is the root, so stop.



Preorder Traversal

1001111

Sample Inputs and Corresponding Outputs

1. Input: $(A + B)$

   Output: 1001111

2. Input: $(A) * (B)$

   Output: 0000001

3. Input: $(-A) * (C) * (-B + D)$

   Output: $00100011100000010000000000000000$