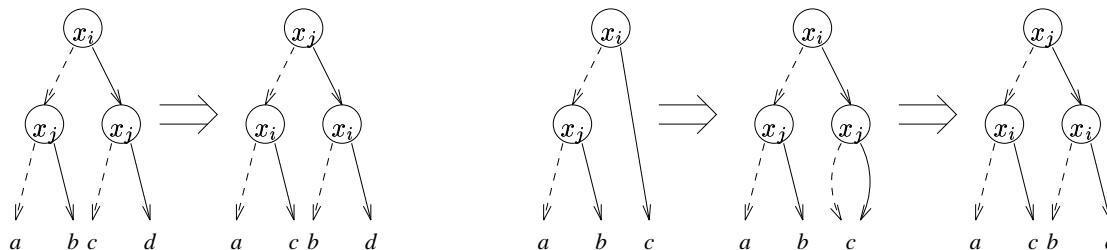


### Problem 3 (30%)

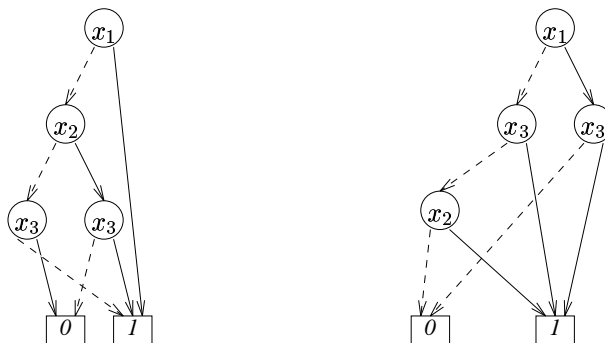
The size of ROBDDs depends heavily on the chosen variable ordering. In this problem you are asked to develop an algorithm, which changes the ordering of an ROBDD. The change is made by small steps called *level exchanges*:



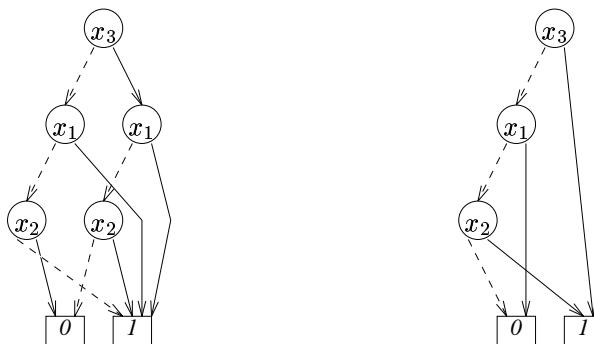
To the left,  $x_j$  is moved up by simply interchanging at the same time  $x_i$  with  $x_j$  and  $b$  with  $c$ . To the right, it is assumed that the ROBDD with root  $c$  does not contain variable  $x_j$ . Before the level exchange, we add a vertex with variable  $x_j$  and low- and high-son  $c$ . There is a symmetric case when the low-son of  $x_i$  does not contain variable  $x_j$ .

#### Question 3.1

Draw figures showing the result of moving  $x_3$  up to the root, using level-exchanges, in the two ROBDDs below. Remember that the result must be reduced.



**Solution:**



#### Question 3.2

Give an algorithm  $UP(j, b)$  which from  $b$  with ordering  $x_1 < \dots < x_n$  uses level

exchanges, to compute a new ROBDD with ordering  $x_j < x_1 < \dots < x_{j-1} < x_{j+1} < \dots < x_n$ , which is equivalent to  $b$ , i.e., is valid on the same truth-assignments as  $b$ . It can be advantageous to use MAKENODE from the BDD-notes.

**Solution:** For readability we have left out  $H$ ,  $max$ , and  $b'$  in the calls to MAKENODE in the algorithm below.

```

UP(j, b) =
  UP'(j, u) =
    if u is terminal then r ← u
    else if var(u) = j then      (* copy *)
      r ← MAKENODE(j, UP'(j, low(u)), UP'(j, high(u)))
    else
      l ← UP'(j, low(u))
      h ← UP'(j, high(u))
      if var(l) = j and var(h) = j then
        r ← MAKENODE(j, MAKENODE(var(u), low(l), low(h)),
                      MAKENODE(var(u), high(l), high(h)))
      else if var(l) = j then
        r ← MAKENODE(j, MAKENODE(var(u), low(l), h),
                      MAKENODE(var(u), high(l), h))
      else if var(h) = j then
        r ← MAKENODE(j, MAKENODE(var(u), l, low(h)),
                      MAKENODE(var(u), l, high(h)))
      else
        r ← MAKENODE(var(u), l, h)
    endif
  endif
  return r

H ← emptytable;  max ← 1
b'.root ← UP'(j, b.root)
return b'

```

### Question 3.3

What is the runtime of  $UP(j, u)$ ? Is it satisfactory? If yes, why? If no, how can it be improved?

**Solution:** The running time is exponential. With the use of dynamic programming (an array  $M$  with an entry for each node to store  $r$ ) it can be brought down to polynomial,  $O(|b|)$ , where  $|b|$  is the number of nodes in  $b$ .

### Question 3.4

Give an implementation of  $UP(j, b)$  which does not use level exchanges, but only uses operations from the BDD-notes.

**Solution:** The variable with index  $j$  can be “pulled up” to the root of an ROBDD  $b$  using the following operations:

$$b' \leftarrow \text{MK}(j, \text{RESTRICT}(b, j, 0), \text{RESTRICT}(b, j, 1))$$