DM - Problème MAX-SAT

2. NP-Complétude de MAX-2-SAT

1.

Supposons qu'il existe une valuation v telle que :

$$[\![x]\!]^v = 0 \text{ et } [\![l_1 \vee l_2 \vee l_3]\!]^v = 1$$

Comme les littéraux l_1, l_2, l_3 ne dépendent pas de x, on peut lui appliquer une valuation sans changer la valeur de vérité de ceux-ci.

$$\text{Alors, } v \text{ satisfait les clauses : } \begin{cases} l_2 \vee \neg x \\ l_3 \vee \neg x \\ l_1 \vee \neg x \end{cases}$$

(si $[\![x]\!]^v = 1$ v ne satisfait qu'une clause)

l_1	l_2	l_3	$(\overline{l_1} \vee \overline{l_2})$	$(\overline{l_1} \vee \overline{l_3})$	$(\overline{l_2} \vee \overline{l_3})$
0	0	1	1	1	1
0	1	1	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	0	0	0

Ce tableau nous informe que 4 clauses sont remplies au maximum.
Ainsi, comme $4 + 3 = 7$, le nombre maximum de clauses satisfaites est $\boxed{7}$.

2.

Supposons qu'il existe une valuation v telle que :

$$[\![l_1 \vee l_2 \vee l_3]\!]^v = 0$$

Pour maximiser le nombre de valuations on prend $[\![x]\!]^v = 0$ comme dans la question précédente, v satisfait 3 clauses
ainsi, comme $[\![l_1]\!]^v = [\![l_2]\!]^v = [\![l_3]\!]^v = 0$ alors,

$$[\![(\overline{l_1} \vee \overline{l_2})]\!]^v = [\![(\overline{l_1} \vee \overline{l_3})]\!]^v = [\![(\overline{l_2} \vee \overline{l_3})]\!]^v = 1$$

Donc, au maximum on a $\boxed{6}$ clauses de satisfaites.

3.

Définir une formule φ' de MAX-2-SAT de taille polynomiale en m et un seuil k tels que φ est satisfiable si et seulement s'il existe une valuation satisfaisant au moins k clauses de φ'.

On pose :

$$\boxed{\begin{aligned} \forall i \in [\![1, m]\!], \varphi' = \quad & l_{i,1} \wedge l_{i,2} \wedge l_{i,3} \wedge x \wedge (\overline{l_{i,1}} \vee \overline{l_{i,2}}) \wedge (\overline{l_{i,2}} \vee \overline{l_{i,3}}) \\ & \wedge(\overline{l_{i,1}} \wedge \overline{l_{i,3}}) \wedge (l_{i,1} \vee \neg x) \wedge (l_{i,2} \vee \neg x) \\ & \wedge(l_{i,3} \vee \neg x) \end{aligned}}$$

et $\boxed{k = 7}$
Soit

$$\varphi = \bigwedge_{i=1}^{m} (l_{i,1} \vee l_{i,2} \vee l_{i,3})$$

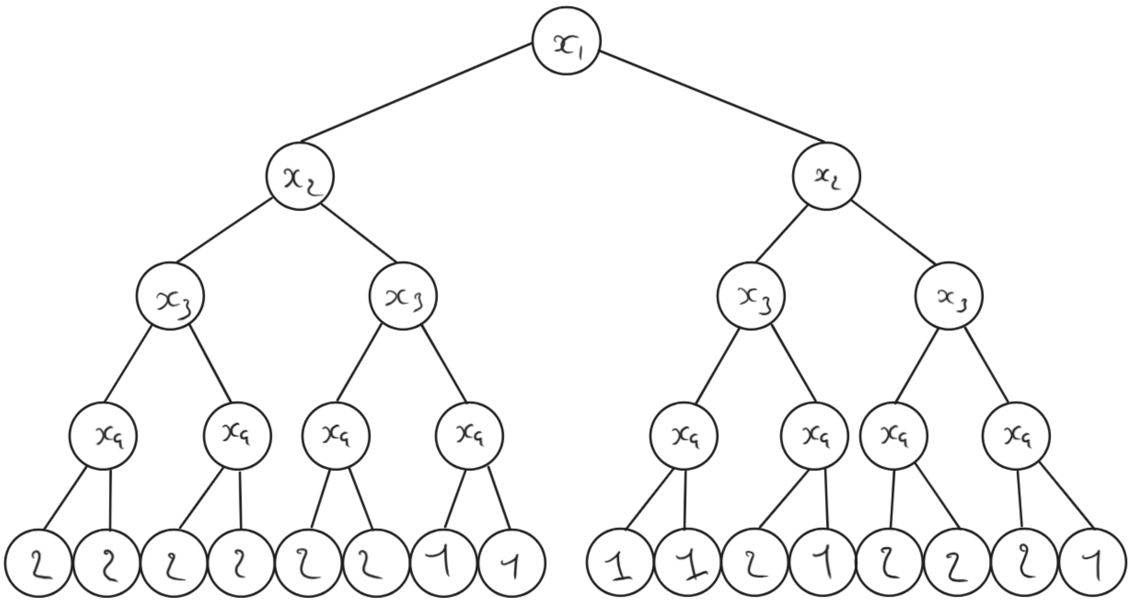
avec $l_{i,1}, l_{i,2}$ ou $l_{i,3} = \perp$ si on a que deux littéraux dans une clause.
Alors,
$\boxed{\Rightarrow}$:
Si φ est satisfiable, $\forall i \in [\![1, m]\!], l_{i,1} \vee l_{i,2} \vee l_{i,3}$ est satisfiable, alors il existe donc une valuation satisfaisant au moins 7 clauses de φ (c'est

même exactement) d'après la question 1.

$\boxed{\Leftarrow}$:

Réciproquement, par contraposition si φ n'est pas satisfiable, alors il existe $i \in [\![1, m]\!]$ tel que $l_{i,1} \vee l_{i,2} \vee l_{i,2}$ n'est pas satisfiable ie d'après la question 2 : pour toute valuation de φ', au plus 6 clauses sont satisfaites (ce qui est bien la négation de : il existe une valuation satisfaisant au moins 7 clauses de φ')

4.



5.

Comme il suffit de minimiser le nombre de clauses de φ_0 d'après l'arbre, la valuation définie par :

$$v(x_1) = 1 \text{ et } v(x_2) = v(x_3) = v(x_4) = 0$$

convient

6.

```
let phi_0 = [[1; 2; 3]; [1; -3; 4]; [1; -4]; [-1; 2; 3]; [-1; -2]; [-1; -3]; [-2; 3]; [2; -3]];;
```

7.

```
let nb_var (f:fnc) =
    let i = ref 0 in
        let rec aux1 phi =
            match phi with
            | [] -> ()
            | t1::s1 -> let rec aux2 cl =
                        match cl with
                        | [] -> ()
                        | t2::s2 -> (if (!i <
                        abs(t2)) then i:= abs(t2)
                        ; aux2 s2)
                        in (aux2 t1; aux1 s1)
            in aux1 f ;
    !i;;
```

8.

```
let v = [|false; true; false; false; true|];;
```

9.

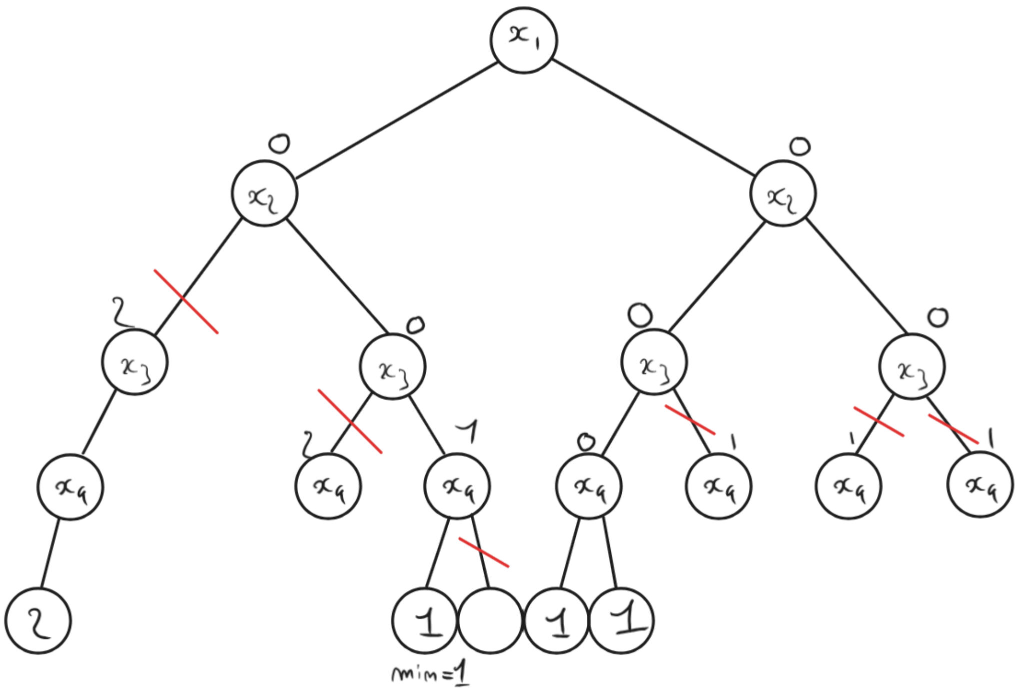
```
let insat_clause (v:bool array) (k:int) (phi:clause) =
    let rec aux (f:clause) =
        match f with
        | [] -> false
        | i::s -> ( if (abs(i) > k) then true
                    else if ((v.(abs(i)) && i>0)
                    || (not v.(abs(i)) && i <0))
                    then true else aux s)
    in aux phi;;
```

10.

```
let insat (v:bool array) (k:int) (f:fnc) =
    let c = ref 0 in
        let rec aux phi =
            match phi with
            | [] -> ()
            | t::s -> if (not (insat_clause v k
                      t)) then (c:=!c +1; aux s)
                      else aux s
        in (aux f; !c);;
```

11.

2 est la borne inférieure initiale de φ_0 alors,



12.

```
let maxSat (f:fnc) =
    let n = nb_var f in
    let v_init = (Array.make (n+1) true) in
    let v_max = ref (Array.make (n+1) true) in (*Tableau ou les valuations des littéraux vérifiant MAX-SAT seront
renvoyés*)
    let min = ref (insat v_init n f) in
        let rec aux v k =
            let v_true = Array.copy v in (*Tableau qui choisit la valuation true pour le litéral k*)
            let v_false = Array.copy v in (*Même chose pour false*)
            begin
```

```
                v_false.(k) <- false;
                let in_sat_true = insat v_true k f in (*borne inférieure de clauses non satisfiable pour
l'évaluation du litéral k à true*)
                let in_sat_false = insat v_false k f in (*Même chose pour false*)
                    if (k=n) then (*Condition d'arret : on atteint une feuille*)
                        (if in_sat_true < !min
                        then (min:= in_sat_true; v_max := v_true)
                        else if in_sat_false < !min then (min:= in_sat_false; v_max := v_false))
                    else (*On continue suivant la valeur de la borne inférieure et du minimum*)
                        if in_sat_true < !min then aux v_true (k+1);
                        if in_sat_false < !min then aux v_false (k+1)
            end;
    in (aux v_init 1; !v_max);;
```