1. Soit $L$ un langage calculable en temps polynomial, c'est à dire il existe une machine de Turing déterministe $M$ et un polynôme $p(n)$ tels que
   - $M$ accepte $L$,
   - sur chaque mot de longueur $n$, la machine s'arrête en $p(n)$ de pas.

   Montrer que $L^*$ est également calculable en temps polynomial, où par définition

$$L^* = \{w_1 \dots w_l : l \geq 0 \text{ and } w_i \in L \text{ for } 1 \leq i \leq l\}.$$

**Solution:** On construira un algorithme qui procède par programmation dynamique. Sur l'entrée $x_1 \dots x_n$, où $x_i \in \{0,1\}$, on va créer un tableau $T$ de dimension $n \times n$, où par définition, pour $i \leq j$:

$$T(i,j) = \begin{cases} 1 & \text{si } x_i \dots x_j \in L^* \\ 0 & \text{sinon.} \end{cases}$$

Il suffira de montrer qu'on peut trouver la valeur $T(1,n)$ en temps polynomial. On a la récurrence suivante sur $j-i$ pour les valeurs $T(i,j)$:

$$T(i,i) = \begin{cases} 1 & \text{si } x_i \in L \\ 0 & \text{sinon,} \end{cases}$$

et

$$T(i,j) = \begin{cases} 1 & \text{s'il existe } i \leq k < j \text{ tel que } T(i,k) = 1 \text{ et } T(k+1,j) = 1, \text{ ou } x_i \dots x_j \in L \\ 0 & \text{sinon,} \end{cases}$$

pour $i < j$.
On peut évaluer les entrées du tableau dans l'ordre suivant:

(a) pour $i = 1$ à $n$ calculer la valeur $T(i,i)$,

(b) pour $i = 1$ à $n-1$
pour $d = 1$ à $n-i$
calculer la valeur $T(i,j)$.

Il y a $O(n^2)$ entrées à calculer. Pour calculer une entrée, on regarde $O(n)$ valeurs déjà calculés du tableau et on fait un calcul polynomial pour décider si $x_i \dots x_j \in L$. C'est pourquoi le temps de calcul de l'algorithme est $O(n^3 p(n))$ où $p(n)$ est un polynôme tel que $L$ est calculable en temps $O(p(n))$.

2. Une séquence d'entiers $(a_0, a_1, \ldots, a_{n-1})$ est appelée *unimodale* s'il existe un entier $0 \leq t \leq n-1$ tel que la séquence $(a_t, a_{t+1}, \ldots, a_{t+n-1})$ est d'abord strictement croissante et après strictement décroissante, où les calculs dans les indices sont performś modulo $n$. Donner un algorithme en temps $O(\log n)$ qui détermine la valeur maximale dans une séquence unimodale.
**Suggestion:** Considérer d'abord le cas $t = 0$.

**Solution:** Une séquence unimodale appartient à une des 4 catégories suivantes:

(a) la séquence d'abord croît, après décroît ($t = 0$),

(b) la séquence d'abord décroît, après croît,

(c) la séquence d'abord décroît, après croît, après décroît,

(d) la séquence d'abord croît, après décroît, après croît.

On peut décider facilement à laquelle de ces 4 catégories $a$ appartient en comparant ses deux premiers et ses deux derniers éléments.

(a) Si $a_1 < a_2$ et $a_{n-2} > a_{n-1}$ alors $a$ appartient à la catégorie (a). Dans ce cas-là, on utilise une méthode similaire à la recherche dichotomique. On compare deux éléments consécutifs $a_i$ et $a_{i+1}$ au milieu de la séquence. Si $a_i < a_{i+1}$, on cherche le maximum récursivement dans la moitié droite de $a$ (y compris $a_{i+1}$), sinon dans la moitié gauche de $a$ (y compris $a_i$). On s'arrête quand il ne reste que deux éléments: le plus grand des deux est le maximum. Le nombre de comparaisons est

$$T(n) = T(n/2) + O(1) = O(\log n).$$

(b) Si $a_1 > a_2$ et $a_{n-2} < a_{n-1}$ alors $a$ appartient à la catégorie (b). Dans ce cas-là le maximum est le plus grand de $a_0$ et de $a_{n-1}$, par conséquent le nombre de comparaisons est $O(1)$.

(c) Si $a_1 > a_2$ et $a_{n-2} > a_{n-1}$ alors $a$ appartient à la catégorie (c). Appelons $b$ la première partie décroissante de $a$, appelons $c$ sa partie croissante, et appelons $d$ sa deuxième partie décroissante. On fait de nouveau une sorte de recherche dichotomique en comparant deux éléments consécutifs $a_i$ et $a_{i+1}$ au milieu de $a$. Si $a_i < a_{i+1}$ alors ils sont en $c$. Dans ce cas-là on fait une recherche comme en cas (a) sur la moitié droite de $a$ (y compris $a_{i+1}$). Si $a_i > a_{i+1}$, ils sont ou bien en $b$ ou en $d$. Cela se décide en comparant $a_i$ avec par exemple avec $a_0$, le premier élément de l'intervalle (en observant que n'importe quel élément en $b$ est plus petit que n'importe quel élément en $d$, sinon la séquence ne serait pas unimodale). Si $a_i < a_0$ alors $a_i$ est aussi en $b$, et on fait récursivement le cas (c) avec la moitié droite de $a$. Sinon $a_i$ est en $d$, et on fait récursivement le cas (c) avec la moitié gauche de $a$.

(d) Si $a_1 < a_2$ et $a_{n-2} < a_{n-1}$ alors $a$ appartient à la catégorie (d). Ce cas est analogue au cas précédent.

3. On rappelle la définition des formules booléennes à $n$ variables qu'on notera $\mathcal{F}_n$:

- $x_i \in \mathcal{F}_n$ for $i = 1, \ldots, n$.
- Si $\phi \in \mathcal{F}_n$ et $\psi \in \mathcal{F}_n$ alors $\bar{\phi} \in \mathcal{F}_n$, $\phi \vee \psi \in \mathcal{F}_n$ et $\phi \wedge \psi \in \mathcal{F}_n$.

Soit $\mathbb{N}$ l'ensemble des entiers naturels. On définit le langage $MAJSAT$ et la fonction $\#SAT : \mathcal{F}_n \to \mathbb{N}$ de la façon suivante:

$$MAJSAT = \{\phi \in \mathcal{F}_n : \phi \text{ est satisfaite par la majorité (plus de } 2^{n-1}) \text{ assignations}\},$$

et

$$\#SAT(\phi) = \text{nombre d'assignations satisfaisantes pour } \phi.$$

(a) Montrer que pour tout $n > 0$, pour tout $0 < j \leq 2^n$, il existe une formule $\psi$ à $n$ variables et de taille polynomiale telle que le nombre d'assignations satisfaisantes pour $\psi$ est $j$.

(b) En utilisant ce résultat, montrer que si $MAJSAT$ est calculable en temps polynomial alors $\#SAT$ est aussi calculable en temps polynomial.

**Solution:** On définit le langage $L$ par

$$L = \{(i, \phi) : 0 \leq i < 2^n, \phi \in \mathcal{F}_n, \phi \text{ est satisfait par plus de } i \text{ assignations}\}.$$

Il suffit de montrer que $L$ est décidable en temps polynomial parce que clairement le calcul de la fonction $\#SAT$ est réductible en temps polynomial à $L$ en utilisant la recherche dichotomique. Pour décider $L$, on utilisera le lemme suivant.
**Lemme:** Pour tout $n > 0$, pour tout $0 < j \leq 2^n$, il existe une formule $\psi$ à $n$ variables telle que le nombre d'assignations satisfaisantes pour $\psi$ est $j$.
**Preuve:** Si $j = 2^n$, par exemple la formule $x_1 \vee \bar{x}_1$ est bonne. Sinon, d'après l'écriture binaire de $j$, il existe $k \leq n$, et des entiers $1 \leq r_1 < \ldots < r_k$, tels que $j = 2^{n-r_1} + \ldots + 2^{n-r_k}$. Alors la formule

$$\psi(x_1, \ldots, x_n) = (x_1 \wedge \ldots \wedge x_{r_1}) \vee (x_1 \wedge \ldots \wedge x_{r_1-1} \wedge \bar{x}_{r_1} \wedge x_{r_1+1} \wedge \ldots \wedge x_{r_2}) \vee \ldots \vee (x_1 \wedge \ldots \wedge \bar{x}_{r_1} \wedge \ldots \wedge \bar{x}_{r_{k-1}} \wedge \ldots \wedge x_{r_k}) \quad (1)$$

est bonne. □
Finalement on peut réduire $L$ à $MAJSAT$ en temps polynomial de la façon suivante. Sur $(i, \phi)$, soit $\psi(x_1, \ldots, x_n)$ une formule qui a $2^n - i$ assignations satisfaisantes. On définit

$$\xi(x_0, x_1, \ldots, x_n) = (x_0 \wedge \phi) \vee (\bar{x}_0 \wedge \psi).$$

Clairement $(i, \phi) \in L$ si et seulement si $\xi \in MAJSAT$.

4. Donner un algorithme de coût linéaire qui prend en entrée un graphe biparti (non orienté) $G = (X \cup Y, E)$ décrit par listes d'adjacences, et un couplage parfait $M$ dans ce graphe et

qui décide si ce graphe possède d'autres couplages parfaits.
**Suggestion:** Orienter les arêtes du couplage de $X$ vers $Y$ et les autres arêtes de $Y$ vers $X$. Appeler $G'$ le graphe résultant.
Que peut-on dire des composantes fortement connexes de $G'$ si $G$ admet (respectivement n'admet pas) au moins deux couplages parfaits distincts?

**Solution:** Notons $G' = (X \cup Y, A)$ le graphe orienté suivant. Les sommets de $G'$ sont ceux de $G$. A toute arête $\{u, v\}$ du couplage $M$ correspond un arc de $u \in X$ vers $v \in Y$. A toute arête de $E \setminus M$ correspond un arc dirigé de $Y$ vers $X$.
*$G$ possède un second couplage parfait si et seulement si $G'$ possède une composante fortement connexe non triviale. De plus, la donnée d'un circuit de $G'$ permet de construire un second couplage parfait de $G$.*
$\Rightarrow$ Supposons que $G$ possède un second couplage parfait $M'$. Il existe une paire de sommets $(u_0, v_0) \in X \times Y$ telle que $(u_0, v_0) \in M$ et $(u_0, v_1) \in M'$, $(u_1, v_0) \in M'$ avec $v_0 \neq v_1$ et $u_0 \neq u_1$. Si $(u_1, v_1) \in M$, $(u_0, v_0, u_1, v_1, u_0)$ forme un circuit dans $G'$. Sinon il existe $(u_2, v_2) \in X \times Y$ tel que $(u_1, v_2) \in M$, $(u_2, v_1) \in M$, si $(u_2, v_2) \in M'$, on a construit un circuit de $G'$.
On rend cette construction systématique par la procédure suivante :

```
PROGRAM construction de cycle;
var chemin : liste;
  u, v : sommet;
  i : integer;
  boucle : boolean;
  begin
    boucle := FALSE;
    chemin := (u_0,v_0);
    i := 0;
    REPEAT
    if (i mod 2 = 0)
      then
          if (premier(chemin), dernier(chemin)) ∉ M'
            begin
              let(premier(chemin), v) ∈ M';
              let(u, dernier(chemin)) ∈ M';
              chemin := (v, chemin,u);
            end;
        else boucle := TRUE;  fi;
       else if (dernier(chemin), premier(chemin)) ∉ M
            then begin
                   let(u, premier(chemin)) ∈ M;
                   let(dernier(chemin),v) ∈ M;
                   chemin := (u, chemin,v);
                 end
             else boucle := TRUE;
          fi;
    fi;
    UNTIL boucle= TRUE;
  end.
```

Tant qu'on n'a pas réussi à fermer un circuit, on construit un chemin qui alterne les arêtes de $M$ et de $M'$. Cela est possible car si $(u_k, v_k)$ ont été adjoints en suivant des arêtes de $M$ (resp. $M'$), alors leurs voisins via $M'$ (resp. $M$) ne peuvent figurer parmi les sommets incorporés précédemment dans le chemin alterné. Cet invariant est vérifié initialement, il est préservé lors de chaque ajout de paire de sommets. Si on a construit $u_{n-1} \ldots, u_2, v_1, u, v, u_1, v_2, \ldots v_{n-1}$, c'est-à dire si on n'a pas réussi à construire un circuit avant d'incorporer la dernière paire de sommets et que celle-ci a été ajoutée au chemin en suivant des arêtes de $M$ (resp $M'$), alors d'après l'invariant préservé par la construction du chemin, $(u_{n-1}, v_{n-1}) \in M'$ (resp.$M$). Le circuit est bouclé.

$\Leftarrow$ Supposons dans l'autre sens que $G'$ comporte une composante fortement connexe non triviale. $G'$ comporte donc un circuit non trivial. Par définition de $G'$ ce circuit alterne des arcs correspondant à des arêtes de $M$ et des arcs correspondant à des arêtes de $E \setminus M$. En échangeant les arêtes de $M$ et celles de $E \setminus M$, on obtient un nouveau couplage parfait de $G$.

La construction de $G'$ requiert un temps linéaire en $n + m$, la construction d'un cycle éventuel dans $G'$ peut être réalisée par un parcours en profondeur en temps $O(m + n)$.

La construction d'un second couplage parfait s'il en existe un est donc apparemment plus facile que la construction du premier couplage parfait.

5. Donner un algorithme linéaire qui décide si une formule 2-SAT $\phi$ est satisfiable. On rappelle qu'une 2-clause est la disjonction de 2 littéraux. Une instance de 2-SAT est une conjonction de 2-clauses.
**Suggestion:** On pourra associer le graphe suivant $G_\phi$ à une formule 2-SAT $\phi$ : à chaque littéral $l$ défini sur l'ensemble des variables, on associe un sommet $v_l$. A la clause $l \wedge l'$, on associe les arêtes $(v_{\neg l}, v_{l'})$ et $(v_{\neg l'}, v_l)$.

   (a) Que peut-on dire sur $\phi$ si il existe une variable $p$ telle que $v_p$ et $v_{\neg p}$ figurent dans une même composante fortement connexe de $G_\phi$ ?

   (b) Si $\phi$ est satisfiable, en notant $H_\phi$ le graphe dont les sommets correspondent aux composantes fortement connexes de $G_\phi$, et où deux sommets sont reliés si les composantes connexes correspondantes le sont, que peut-on dire des sommets associés à $p$ et $\neg p$?

**Solution:** Elle repose sur le fait suivant : une formule 2-SAT $\phi$ est satisfiable si et seulement si pour toute variable $p$ figurant dans $\phi$, $v_p$ et $v_{\neg p}$ n'appartiennent pas à la même composante fortement connexe de $G_\phi$.
$\Rightarrow$ Supposons qu'il existe une assignation $\sigma$ qui satisfait l'ensemble des 2-clauses de $\phi$. Alors si $l$ est un littéral et $\sigma(l) = 1$, si $l$ est un littéral, si $v_{l'}$ est accessible à partir de $v_l$, $\sigma(l') = 1$. On peut le montrer par induction sur la longueur du chemin de $v_l$ à $v_{l'}$. Donc on ne peut avoir $v_l$ et $v_{\neg l}$ dans la même composante fortement connexe.
$\Leftarrow$ Supposons que pour tout littéral $v_l$ et $v_{\neg l}$ n'appartiennent pas à la même composante fortement connexe du graphe $G$ associé à $\phi$.
*Remarques liminaires.*

   (a) Rappelons d'abord que la décomposition de $G_\phi$ en composantes fortement connexes définit un graphe orienté $H_\phi$ dont les sommets correspondent aux composantes fortement connexes de $G_\phi$, et oú il existe un arc entre le sommet associé à la composante connexe $C$ de $G_\phi$ et le sommet associé à la composante connexe $C'$ de $G_\phi$, s'il existe un arc $(u, v)$ dans $G_\phi$ avec $u \in C$ et $v \in C'$. $H_\phi$ est sans circuit.

   (b) Toute assignation $\sigma$ qui satisfait $\phi$ doit satisfaire la propriété suivante : tous les littéraux associés à des sommets d'une même composante fortement connexe doivent être assignés à la même valeur dans $\phi$.

   (c) Si $l$ et $l'$ sont associés á la même composante fortement connexe de $G_\phi$, alors $\neg l$ et $\neg l'$ sont aussi associés à une même composante fortement connexe de $G_\phi$.

   (d) Un puits (resp. une source) de $H_\phi$ est un sommet de degré sortant (resp. entrant) nul. Si un littéral est associé à un puits de $G'$, alors sa négation est associée à une source de $G'$.

   (e) Si une assignation satisfait $\phi$, elle détermine une partition des sommets de $H_\phi$, et le

sous-ensemble des sommets associé à la valeur 1 est clos par successeur.

(f) Enfin rappelons qu'un sous-graphe induit de $H_\phi$ définit une sous-formule de $\phi$ : l'ensemble des clauses de $\phi$ où apparaissent seulement les littéraux associés à des sommets du sous-graphe induit de $G_\phi$.

La profondeur de $H_\phi$ est la longueur du plus long chemin de $H_\phi$. Pour prouver l'existence d'une assignation satisfaisant $\phi$, on peut raisonner par induction sur la profondeur du graphe quotient $G'$ de $G_\phi$.
Si le graphe quotient est de profondeur 0. On peut d'après la remarque (3) regrouper les sommets de $H_\phi$ par paires : chaque paire correspond à un groupe de littéraux, elle contient la composante associée à ces littéraux et la composante associée à leurs négations. Pour chaque paire il faut (et il suffit de) choisir la composante qui sera affectée à 1.
Supposons que l'on sache construire une assignation satisfaisante$\sigma$ lorsque $H_\phi$ est de profondeur $\leq k$.
Si $H_\phi$ est de profondeur $k+1$. On construit l'assignation de la façon suivante : tout littéral associé à un puits qui n'est pas en même temps une source de $H_\phi$ est affecté à 1 dans $\sigma$. Les sources correspondants à ces puits sont affectées à 0 dans $\sigma$. Les sommets affectés de $H_\phi$ (ce sont des puits ou des sources de $H_\phi$) sont retranchés de $H_\phi$. Le sous-graphe induit par les sommets restants définit une sous-formule de $\phi$, et ce graphe est de profondeur $\leq k$. On peut donc construire une assignation $\sigma'$ satisfaisant cette sous-formule.
En complétant $\sigma$ à l'aide de $\sigma'$, on obtient une assignation qui satisfait $\phi$.