IFT2105 Devoir3

Samy Rasmy 20214818 Yuchen Hui 20150470

March 25, 2022

Question1

Question2 a) 3-ClIQUE

3-CLIQUE est un langage dans la classe P.

Proof. On donne un algorithme en temps polynômial.

Algorithm 1 fonction 3-CLIQUE? $(G[1 \ldots n, 1 \ldots n])$: bool

---

{ici on représnete le graphe par une matrice adjacente de $n * n$, où n est le nombre de noeud. S'il exsite une arête entre noeud i et noeud j, alors $Gi[i,j] = true$; sinon $G[i,j] = false.$}
for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    if $G[i,j] == true$ **then**
      for $k = 1$ **to** n **do**
        if $G[i,k] == true$ AND $G[k,j] == true$ **then**
          return true
return false

On voit bien que la complexité en temps est de $\mathcal{O}\left(n^3\right)$ (n est le nombre des noeuds), d'où le langage 3-CLIQUE $\in P$

□

Question2 b) 3-DOUBLE-SAT

Le langage 3-DOUBLE-SAT est NP-Complet.

Proof. On prouve d'abord qu'il $\in NP$

0.1 3-DOUBLE-SAT $\in NP$

Un certificat est deux affectation des varibles pour l'expression booléenne. Un vérificateur accepte si tous les deux affectations rendent l'expression vraie, sinon il rejette. Évidement, ce vérificateur travaille dans $\mathcal{O}\left(n\right)$, où n est le nombre de clauses dans l'expression booléenne.

0.2 3-SAT $\leq_p$ 3-DOUBLE-SAT

Soit la fonction de réduction:

$$\begin{aligned} f : \Sigma^* &\longrightarrow \Sigma^* \\ <E> &\longmapsto f(<E>). \end{aligned}$$

telle que
si e est l'expression correspondant au <E>, alors $e \wedge (x_i \vee \overline{x_i} \vee \overline{x_i})$ est l'expression représenté par $f\left(<E>\right)$, où on introduit une nouvelle variable x_i qui avec $\overline{x_i}$ n'apparaissent jamais dans e.

0.2.1 <E> $\in$ 3-SAT $\implies f\left(<\mathbf{E}>\right) \in$ 3-DOUBLE-SAT

soit e est l'expression correspondant au <E>.

$$\begin{aligned} < E > \in \text{3-SAT} &\implies \text{il existe au moin une affectation (dénotons par A) qui satisfait } e \\ &\implies \text{il existe au moin deux affectation } (\text{A} \cup [x_i = true]) \text{ et } (\text{A} \cup [x_i = false]) \\ &\qquad \text{qui satisfont } e \wedge (x_i \vee \overline{x_i} \vee \overline{x_i}) \\ &\implies f\left(<\text{E}>\right) \in \text{3-DOUBLE-SAT.} \end{aligned}$$

0.2.2 <E> $\notin$ 3-SAT $\implies f\left(<\mathbf{E}>\right) \notin$ 3-DOUBLE-SAT

soit e est l'expression correspondant au <E>.

$$\begin{aligned} < E > \notin \text{3-SAT} &\implies \text{il n'existe pas d'affectation qui satisfait } e \\ &\implies \text{il n'existe pas d'affectation qui satisfait } e \wedge (x_i \vee \overline{x_i} \vee \overline{x_i}) \\ &\quad \text{, car } e \text{ prend toujour la valeur fause} \\ &\implies f\left(<\text{E}>\right) \notin \text{3-DOUBLE-SAT.} \end{aligned}$$

0.2.3 Ça se voit que f est de $\mathcal{O}\left(1\right) \subseteq \mathcal{O}\left(n\right)$

Finalement, on peut conclure que 3-DOUBLE-SAT est NP-Complet, étant donné qu'il est à la fois $\in$ NP et NP-Difficile (un langage NP-Complet se réduit à lui). $\square$

Question2 c) 4-COL

Le langage 4-COL est NP-Complet.

Proof. On prouve d'abord qu'il $\in$ NP.

0.3 4-COL $\in NP$

Un certificat est un coloriage contenant 3 couleurs. Un vérificateur scanne toutes les arêtes et vérifie si les deux noeud adjacents de chaque arêtes portent des couleurs différentes. Si oui il l'acceptera, sinon il le rejette. Ce vérificateur travaille dans $\mathcal{O}(|E|) \subseteq \mathcal{O}\left(|V|^2\right)$, donc 4-COL $\in$ NP.

0.4 3-COL $\leq_p$ 4-COL

Soit la fonction de réduction:

$$\begin{aligned} f : \Sigma^* &\longrightarrow \Sigma^* \\ <G> &\longmapsto f(<G>). \end{aligned}$$

telle que
Dénotons g le graphe correspondant au <G>, g' le graphe correspondant au $f(<G>$. Nous construisons g' en ajoutant un noeud supplémentaire au g et en reliant ce noeud aux tous les sommets de g.

0.4.1 **<G>** $\in$ **3-COL** $\implies$ $f\,($**<G>**$) \in$ **4-COL**

Soit g est le graphe correspondant au <G>. Soit g' est le graphe correspondant au $f < G >$.

$$\begin{aligned} < G > \in \text{3-COL} &\implies \text{Graphe g est 3-coloriable} \\ &\implies \text{Dans } g'\text{, si on assigne toujours au sommet ajouté une couleur} \\ &\quad\ \text{autres que les couleurs utilisé par les sommets de } g\text{,} \\ &\quad\ \text{il est possible d'obtenir un coloriage qui} \\ &\quad\ \text{n'utilise que 4 couleurs: Par exemple,} \\ &\quad\ \text{On colorie le sougraphe } g \text{ de } g' \text{ en utilisant} \\ &\quad\ \text{rouge, vert et bleu en formant un coloriage valide de g,} \\ &\quad\ \text{ensuite on colorie le sommnet supplémentaire en orange.} \\ &\quad\ \text{Ainsi on obtient un 4-coloriage valide de } g'. \\ &\implies f\,(\text{<G>}) \in \text{4-COL.} \end{aligned}$$

0.4.2 **<G>** $\notin$ **3-COL** $\implies$ $f\,($**<G>**$) \notin$ **4-COL**

Soit g est le graphe correspondant au <G>. Soit g' est le graphe correspondant au $f(< G >)$. <G> $\notin$ 3-COL signifie qu'il faut au mois utiliser 4 couleurs afin que le graph g soit coloriable. Ainsi, dans graphe g', puisque le sommet qu'on ajoute est relié aux tous les sommets de g, il nous faut lui assigner une couleur autres que les couleurs utilisées par ces sommets pour que le praphe g' soit coloriable. Alors il faut maintenant au moins 5 couleurs (4 aux g et 1 au sommet ajouté) si on voudrait colorier g' correctement. Donc g' n'est pas 4-coloriable et on peut conclure que $f\,(<G>) \notin$ 4-COL.

0.4.3 Ça se voit que f est de $\mathcal{O}(|V|)$

Finalement, on peut conclure que 4-COL est NP-Complet, étant donné qu'il est à la fois $\in$ NP et NP-Difficile (un langage NP-Complet 3-COL se réduit à lui). $\square$

Question3

Soit e l'expression booléenne sous considération, $x_1, x_2, \ldots, x_n$ les n variables qui apparaissent dans e (sous forme de x_i ou $\overline{x_i}$). Maintenant étudions deux expression: $e_1 = e \wedge (x_i \vee x_i \vee x_i)$ et$e_2 = e \wedge (\overline{x_i} \vee \overline{x_i} \vee \overline{x_i})$. Supposons que Oracel($e$) accepte, i.e. e est satisfaisable(typo?). Alors on a:

$Oracle(e_1) == true \implies$ il existe une affectation satisfaisable où x_i prend la valeur true.

$Oracle(e_2) == true \implies$ il existe une affectation satisfaisable où x_i prend la valeur false..

Avec ces deux observations, on peut trouver un algorithme:
Puisqu'un appel au Oracle prend un temps constant, cet algorithme s'execute

Algorithm 2 fonction Affectation(e) : **array** $X[1 \ldots n]$

```
{e est l'expression booléenne, X est un tableau qui va contenir une affectation
satisfaisable à la fin.}
if Oracle(e) == false then
return e n'est pas satisfaisable.
for i = 1 to n do
    if Oracle(e ∧ (x_i ∨ x_i ∨ x_i)) == true then
        e ← e ∧ (x_i ∨ x_i ∨ x_i)
        X[i] == true
    else if Oracle(e ∧ (x̄_i ∨ x̄_i ∨ x̄_i)) == true then
        e ← e ∧ (x̄_i ∨ x̄_i ∨ x̄_i)
        X[i] = false
return X
```

en $\mathcal{O}(n)$.