

- Beaucoup de pbs dans NP n'ont pas d'algorithme connu en temps $O(n^k)$. (dét.)
 - ex: SAT, CLIQUE, ...
 - Mais on en connaît pour certains, PATH, $\{(a,b,c) : ab=c\}$
 - Certains pbs dans NP paraissent + difficiles que d'autres
 - Certains problèmes se réduisent à d'autres, dans le sens où si A se réduit à B, un algo $O(n^k)$ pour B donnerait un algo $O(n^{k'})$ pour A.

Exemple hyper-simple

$\text{IND-SCT} = \{ \langle G, k \rangle : G \text{ a un ens. indép. de taille } k \}$

Théorème: s'il existe un algo $O(n^k)$ pour iIND-SET,
on peut décider CLIQUE en temps $O(n^{k'})$, $k, k' \in \mathbb{N}$.

Preuve: supposons qu'on a un algo M qui décide IND-SET en temps $O(n^k)$.

Soit $\langle G, k \rangle$ une instance de CLIQUE. $k=2$

Soit \bar{G} le complément de G .

- $V(\bar{G}) = V(G)$
 - $E(\bar{G}) = \{uv : uv \notin E(G), u \neq v\}$



On crée un algo M' pour décider CLIQUE:

sur entrée $\langle \hat{G}, k \rangle$

calcular \bar{G}

donner $\langle \bar{G}, k \rangle$ à l'algorithme M pour IND-SET temps $M^k O(n^k)$
 accepter ssi M a accepté

Puisque C est une clique de $G \Leftrightarrow$

C'est un ens indep dans \tilde{G} ,

M accepte $\langle G, k \rangle \Leftrightarrow M$ accepte $\langle \bar{G}, k \rangle$. Donc M décide CLIQUE.

De plus, M' prend un temps $\mathcal{O}(\underline{|V(G)|^2 + \text{temps}(M) })$

calcular \bar{G}

$$= O(|V(G)|^2 + n^k) = O(n^k)$$

Consequence: IND-SET est au moins aussi "difficile" que CLIQUE
 si $\text{IND-SET} \in P$, CLIQUE serait dans P.

"Tout le monde" croit que CLIQUE & P.

Donc, IND-SET n'est probablement pas dans P.

Le passage de \underline{G} à $\overline{\underline{G}}$ s'appelle une transformation polynomiale.

Plus généralement, un langage A est réductible en temps polynomial à un langage B si il existe une fonction

$$f : \Sigma^* \rightarrow \Sigma^* \text{ calculable en temps polynomial}$$

telle que

$$w \in A \iff f(w) \in B$$

Dans ce cas, on écrit $A \leq_p B$
f s'appelle une transfo. polynomiale

(pourquoi? pour
montrer qu'un algo
poly pour B donnerait
un algo poly pour A
si on utilisait f)

CLIQUE \leq_p IND-SET

f: prendre le complément

$$\langle G, k \rangle \in \text{CLIQUE} \iff$$

$$\langle \bar{G}, k \rangle \in \text{IND-SET}$$

$$(\text{car } \langle \bar{G}, k \rangle = f(\langle G, k \rangle))$$

Théorème: si $A \leq_p B$ et $B \in P$, alors $A \in P$.

Preuve: si $B \in P$, $\exists M$ décidant B en $O(n^k)$.

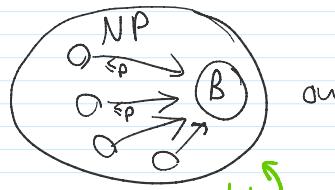
Pour $w \in \Sigma^*$, on veut décider si $w \in A$. On fait:

- calculer $f(w)$ poly
- retourner le résultat de M sur $f(w)$. poly

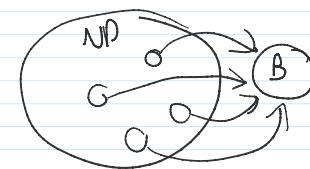
Si on "pense" que $A \notin P$ et que $A \leq_p B$, alors on peut "penser"
que $B \notin P$.

• Un langage B est NP-difficile (NP-hard) si:

$$A \leq_p B \quad \forall A \in NP$$



NP-complet



NP-difficile

• Un langage B est NP-complet si:

① $B \in NP$

② B est NP-difficile

① $B \in NP$

② B est NP-difficile

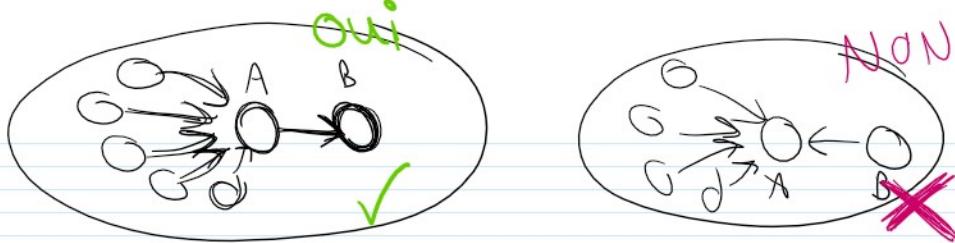
Intuition: si B est NP-difficile (ou même NP-complet) et que B se réduit à mon langage, alors mon langage est aussi difficile.

Lemme: \leq_p est transitive, i.e. $A \leq_p B \leq_p C$, alors $A \leq_p C$.

Théorème: si A est NP-difficile (ou NP-complet) et $A \leq_p B$, alors B est aussi NP-difficile.

Preuve: puisque A est NP-diff., alors $\forall C \in NP, C \leq_p A$.

Par transitivité, $\forall C \in NP, C \leq_p A \leq_p B \Rightarrow C \leq_p B$. \blacksquare



Formule (recette) générale pour montrer qu'un pb B est NP-complet.

① Montrer que $B \in NP$ (souvent facile par vérif.)

② Montrer que $A \leq_p B$, où A est NP-difficile.

└ Trouver f telle que $w \in A \Leftrightarrow f(w) \in B$

└ 2.1 Présenter ce que fait f

└ 2.2 Montrer que $w \in A \Rightarrow f(w) \in B$

└ 2.3 Montrer que $f(w) \in B \Rightarrow w \in A$

• On va réduire SAT à CLIQUE.

(montrer que CLIQUE est NP-complet sachant que SAT est NP-difficile)

• $3\text{-SAT} = \{ \varphi : \varphi \text{ est en forme } 3\text{-CNF et } \varphi \text{ est satisfaisable} \}$

└ une formule log. φ est en CNF si φ est un \wedge de plusieurs clauses. Clause = ne contient que des V.

ex: CNF: $\varphi = (x_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4)$

soit une première clause. On va la colorier que nous.

$$\text{ex: CNF: } \varphi = (\underline{x_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x}_4}) \wedge (\underline{\bar{x}_1 \vee x_2}) \wedge (\underline{\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4})$$

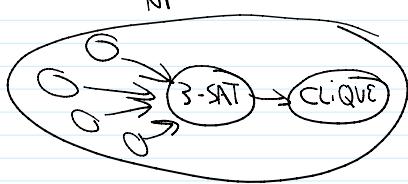
clause clause clause

Une formule φ est en 3-CNF si elle est CNF et chaque clause contient 3 variables (positive/négative).

$$\text{ex: 3-CNF } \varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee x_5)$$

Théorème: 3-SAT est NP-complet. (+Tard)

Cook-Levin NP



But: simplifie nos réductions

Théorème: CLIQUE est NP-complet.

Preuve: ① CLIQUE ∈ NP, car une clique du graphe est de certificat vérifiable en temps poly.

② 3-SAT \leq_p CLIQUE

Soit φ une instance de 3-SAT.

Soient x_1, \dots, x_n ses variables et soient C_1, \dots, C_m ses clauses.

$$\text{ex: } C_1 = (x_1 \vee x_2 \vee \bar{x}_3) \quad C_2 = (\bar{x}_2 \vee x_3 \vee \bar{x}_4) \quad C_3 = (x_2 \vee x_3 \vee \bar{x}_5)$$

$$\varphi = C_1 \wedge C_2 \wedge C_3 \quad (\text{en général } \varphi = C_1 \wedge C_2 \wedge \dots \wedge C_m)$$

2.1
décrire
f

On veut transformer φ en une instance $\langle G, k \rangle$ du CLIQUE telle que $\varphi \in 3\text{-SAT} \iff \langle G, k \rangle \in \text{CLIQUE}$.

On pose $k = m$. (# de clauses)

Pour construire G , pour chaque clause $C_i = (a_i \vee b_i \vee c_i)$

on ajoute à G 3 sommets a_i^0, b_i^0, c_i^0 .

\Rightarrow il y a m clauses, chacune avec 3 vars $\Rightarrow G$ a $3m$ sommets

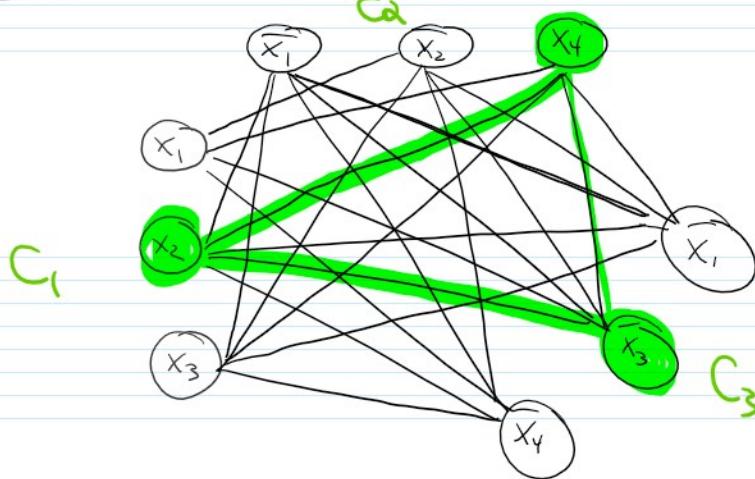
[Pour les arêtes, on ajoute une arête entre y_i et z_j de G si y_i et z_j proviennent de clauses différentes]

Pour les arcées, on ajoute une arête entre y_i et z_j de \Leftrightarrow
 y_i et z_j proviennent de clauses différentes
et ne représentent pas des variables dont l'une est
la négation de l'autre.



choix des variables y_i et z_j = compatibles

ex: $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4)$



G

$k=3$ ($k=m$)

Pour former une clique de taille 3, choisir un sommet par triplet de clause sans choisir de variables contradictoires.

En vert: clique de taille 3, correspond à l'assignation $x_2=T, x_4=T, x_3=F$ satisfait φ

2.2 $\varphi \in 3\text{-SAT} \Rightarrow \langle G, k \rangle \in \text{CLIQUE}$ (i.e. G a une clique de taille $k=m$)

Si $\varphi \in 3\text{-SAT}$, alors l'assignation A des x_i t.g. chaque clause C_i est satisfaite par une valeur d'une de ses variables.

Dans G, on construit une clique K de taille m en choisissant, pour clause C_i , le sommet correspondant à la variable qui a servi à satisfaire C_i dans notre assignation A.

Puisque A n'affecte pas T et F à une variable x_i , les communs choix n'ont pas tous une arête et

Puisque A n'affecte pas 'l et t à une variable x_i ,
les sommets choisis partagent tous une arête et
dans forment une clique, de taille m.

2.3

semaine prochaine

$$\langle G, k \rangle \in \text{CLIQUE} \Rightarrow \varphi \in 3\text{-SAT}$$