Complexité: Motivation

Motivation: Premier exemple

- Problème **REACH:**
  - **Instance:** un graphe orienté $G = (V, E)$,deux sommets $u, v \in V$
  - **Question:** Existe-t'-il un chemin entre u et v dans G?
- C'est un problème de décision.
- Il existe un algorithme "raisonnable" (polynômial): parcours de graphe en profondeur, largeur,. . . en temps $O(n^2)$ où n est le nombre de sommets.

Motivation: Algorithme pour REACH

- **Parcours de graphe:**
  1. $S = \{u\}$.
  2. Pour $i = 1$ à $|V|$ faire Marque[i]:=0;
  3. Tant que $S \neq \emptyset$
     (a) Choisir $i \in S$; $S := S - \{i\}$; $Marque[i] := 1$.
     (b) Pour tous les j avec $(i, j) \in E$
        i. Si $Marque[j] = 0$ alors Ajouter j à S.
  4. Accepter si $Marque[v] = 1$, rejeter sinon.

- **Cet Algorithme en temps et espace mémoire polynômial $O(n^2)$ avec $n = |V|$ le nombre de sommets.**

- **Un résultat de la complexité:** Il existe un algorithme en espace mémoire $O(\log^2 n)$.

Motivation: Circuit Hamiltonien

- Problème **HAMILTON:**
  - **Instance:** un graphe $G = (V, E)$
  - **Question:** existe t'il un cycle hamiltonien (un cycle qui passe par tous les sommets du graphe une seule fois)?
- C'est un problème de décision.
- Il existe un algorithme en $O(n!)$, ou en $O(2^n)$: non raisonnable.
- Pas d'algorithme polynômial connu.

Motivation: Voyageur de Commerce

- Problème **MINCYCLE:**
  - **Instance:** un graphe $G = (V, E)$, les distances $d(i, j) \in \mathbb{N}$ pour $(i, j) \in E$
  - **Question:** Quel est le cycle de longueur minimum ?
- Ce problème **n'est pas** un problème de décision.
- Problème de décision considéré en complexité: Problème **TSP:**
  - **Instance:** un graphe $G = (V, E)$, les distances $d(i, j) \in \mathbb{N}$ pour $(i, j) \in E$, un entier k.
  - **Question:** Existe-t'-il un cycle de longueur inférieure à k ?

Problème de décision

- **Définition formelle:** un problème de décision $\Pi = (D_\Pi, Y_\Pi)$ est la donnée d'un ensemble d'instances D_Π et d'un sous-ensemble $Y_\Pi \subset D_\Pi$ d'instances positives.
- Problème **SUBGRAPHISOM.:**
  - **Instance:** Deux graphes $G_1 = (V_1, E_1)$, $G_2 = (V_2,\ E_2)$.
  - **Question:** G_1, contient-il un sous-graphe isomorphe à G_2 ?

  $D_\Pi = \{(G_1, G_2)\}$ et $Y_\Pi =$ les instances positives.

Comment coder les instances ?

- **Entiers:** n

  1. Binaire: $|n| = \log n$.
  2. Unaire: $|n| = n$.

- **Graphes:** $G = (V, E)$

  1. Matrice d'adjacence: $|G| = |V|^2$.
  2. Liste d'adjacence: $|G| = |V| \times |E| \times log(|V|)$.
  3. Liste d'arêtes: $|G| = 2 \times |E| \times log(|V|)$.

- **Applications sur les exemples:**
  - Sous-graphe isomorphe: $|(G_1, G_2)| = |G_1| + |G_2|$.
  - Voyageur de commerce:
    $|(C, d, k)| = |C| + log(d_{max})|C|^2 + log(k)$.

Algorithme

- **Algorithme:** Programme en $basic$, C, $pascal$, $logo$, $Java$, $machine\ de\ Turing$ ...
- **Résultat fondamental:** Un programme A_1 en langage 1 se traduit en un programme A_2 en langage 2 avec

$$TempsExecution(A_2) = p_{1\to 2}(TempsExecution(A_1))$$

pour un polynôme $p_{1\to 2}$.

- **Conséquences:** Raisonnable $=$ Polynômial.

Choix du codage

- **Conséquence:** le choix du codage n'est pas très important **si** ils sont équivalents à temps polynômial près.

- **Exemple:** G graphe quelconque (V, E).

  On peut transformer le codage du graphe en liste d'adjacence
  ($|G| = 2|E| * \log(|V|)$) en matrice d'adjacence
  ($|G| = |V|^2$)
  en temps polynômial (car $|E| = O(|V|^2)$) .

- **Contre Exemple:**

  coder en unaire $\neq$ coder en binaire.

Notion de réduction entre problèmes

- Problème **EQUATION1:**
  - **Instance:** Entiers α, β.
  - **Question:** Existe-t'-il un entier x tel que $\alpha x + \beta = 0$?
- Problème **EQUATION2:**
  - **Instance:** Entiers a,b, c.
  - **Question:** Existe-t'-il un entier x tel que $ax^2 + bx + c = 0$?
- Le problème EQUATION1 est plus simple que le problème EQUATION2.

Notion de réduction entre problèmes

- **Définition:** un problème $Pb1$ est plus simple qu'un problème $Pb2$ (noté $Pb1 \leq_P Pb2$) : s'il existe une fonction $f : D_{Pb1} \rightarrow D_{Pb2}$ qui se calcule en temps polynômial telle que

$$x \in Y_{Pb1} \;\; ssi \; f(x) \in \; Y_{Pb2}$$

La transformation polynômiale f est définie de la façon suivante :

- $f : D_{\text{Problème 1}} \rightarrow D_{\text{Problème 2}}$
- $f(\alpha, \beta) = (a, b, c)$ avec $\begin{cases} a := 0 \\ b := \alpha \\ c := \beta \end{cases}$

Transformation polynômiale

- **Définition:** un problème $Pb1$ est plus simple qu'un problème $Pb2$ (noté $Pb1 \leq_P Pb2$) : s'il existe une fonction $f : D_{Pb1} \rightarrow D_{Pb2}$ qui se calcule en temps polynômial telle que

$$x \in Y_{Pb1} \quad ssi \; f(x) \in \; Y_{Pb2}$$

Problème **Problème 1:**

- **Instance:** Entiers α, β.
- **Question:** Existe-t'-il x tel que $\alpha x + \beta = 0$?

Problème **Problème 2:**

- **Instance:** Entiers a, b, et c.
- **Question:** Existe-t'-il x tel que $ax^2 + bx + c = 0$?

$\Longrightarrow$

$$Pb1 \leq_P Pb2$$

Problème SAT

- **Définition:**
  - Une variable booléenne est un symbole à qui on peut associer la valeur "true" ou "false".
  - Formule booléenne = formule obtenue par la grammaire $x|true|false|\phi \wedge \psi|\phi \vee \psi|\neg\phi$, où x est une variable booléenne.
- Problème **SAT:**
  - **Instance:** Une formule booléenne Φ
  - **Question:** ϕ est elle satisfiable?
- Existe-t'-il un algorithme polynômial?

Problème SAT

- **Un algorithme exponentiel:**

1. Vérifier que ϕ encode une formule correcte.
2. Soit n le nombre de variables.
3. Pour chaque w dans $\{0,1\}^n$.
   (a) Pour $i = 1$ à n, substituer variable numéro i par $i$ème bit de w.
   (b) Simplifier formule résultante.
   (c) Si obtient true, accepter.
4. Rejeter.

Algorithme non déterministe

Algorithme non déterministe: algorithme avec les instructions classiques + instruction “choisir entre instruction $x := 0$ et $x := 1$ de façon non-déterministe”.

Classe P: problèmes de décision qui peuvent être décidés par un algorithme *déterministe* qui répond en un temps polynômial en la taille de l’instance.

Classe NP: problèmes de décision qui peuvent êtres décidés par un algorithme *non déterministe* qui répond en un temps polynômial en la taille de l’instance.

Question: $P = NP$?

Algorithme non déterministe

- **Proposition:** SAT est dans NP.
- **Preuve:** Algorithme Non-déterministe.

1. Vérifier que ϕ encode une formule correcte.
2. Soit n le nombre de variables.
3. Mettre w au mot vide.
4. Pour $i = 1$ à n,
   (a) choisir entre $x := 0$ ou $x := 1$ de façon non-déterministe.
   (b) $w := xw$.
5. Pour $i = 1$ à n, substituer variable numéro i par $i$ème bit de w.
6. Simplifier formule résultante.
7. Si obtient true, accepter, sinon rejeter.

Algorithme non déterministe

- **Proposition:** TSP est dans NP.
- Preuve: exercice.
- **Proposition:** HAMILTON est dans NP.
- Preuve: exercice.

Problème NP-complet

- **Définition:** Un problème Π est NP-complet si
  - $\Pi \in NP$,
  - Π est NP-difficile, i.e, ($\forall A \in NP$, $A \leq_P \Pi$).
- Remarque: un problème Π est NP-difficile ssi il est plus difficile qu'un problème NP-complet.

Problème SAT

Théorème de Cook. SAT est $NP-$complet.

Quelques résultats

- **Classification des problèmes**
  1. P:
     - REACH est dans P.
  2. NP:
  3. NP-complet:
     - HAMILTON est NP-complet.
     - TSP est NP-complet.
- **Corollaire:** $HAMILTON \equiv TSP \equiv 3SAT.$
- **Question:** $P = NP$?
- **Théorème:** Si $P \neq NP$, il existe des problèmes dans NP et non complet.

Aller plus loin

- **Classes de complexité en espace:** $PSPACE$, $NPSPACE$, ...
- **Caractérisations logiques**
- **Caractérisations algébriques**
- ...
- **Algorithmes parallèles**
- **Algorithmes d'approximation**
- **Algorithmes probabilistes**
- ...