Définitions

Algorithme :

- Méthode pour résoudre un **problème**, qui donne une solution **exacte** en un **temps fini** pour toutes les **instances** du problème.
- Séquence **finie** d'étapes **bien définies** qui transforment l'**entrée** en la **sortie.**

Entrée(s) : Donnée(s) à fournir à l'algorithme.
Sortie(s) : Résultat(s).

Instance (ou exemplaire) d'un problème : Une suite de valeurs données en entrée.

Programme : Traduction de l'algorithme en un langage compréhensible par l'ordinateur.

Remarques

- Certains auteurs admettent des boucles infinies dans la définition d'un algorithme.
- La caractéristique d'exactitude est parfois levée par exemple pour certains tests de primalité. Le résultat est alors seulement « probablement » exact.
- Un même problème peut être résolu par différents algorithmes.

Origines :

- Mathématicien perse du IXe siècle Al-Khwârizmî.
- Les premiers algorithmes ont été développés essentiellement en arithmétique (algorithme d'Euclide par exemple).

Caractéristiques d'un algorithme

- Il faut déterminer :
  - L'ensemble des valeurs possibles en entrée
  - La taille du problème (le nombre de « données élémentaires » en entrée)
- On évaluera les algorithmes en fonction de la taille du problème qu'ils résolvent (potentiellement plusieurs paramètres).
- Un algorithme **efficace** consomme peu de ressources (temps, espace).
- Un algorithme **optimal** est aussi efficace, sinon plus, que tous les autres algorithmes résolvant le même problème. Attention, ce concept ne dépend pas seulement du problème mais aussi des données qu'on aura à traiter.

Types de problèmes

- Problèmes de décision : La sortie est un booléen.
  - Fouille dans un tableau
  - Satisfaisabilité (SAT) d'une expression booléenne
  - Primalité d'un nombre entier
- Problèmes de recherche :
  - Recherche des occurrences d'un élément dans un tableau
  - Recherche d'un nombre premier qui divise un entier donné
  - Problème de la sélection

Tout problème de décision est la restriction d'un problème de recherche.

Types de problèmes

- Problèmes d'optimisation : on cherche à minimiser ou maximiser une fonction.
  - Problème du flot maximal
  - Problème du commis voyageur
  - Recherche du plus grand facteur premier d'un entier donné
- Problèmes de dénombrement :
  - Nombre de facteurs premiers d'un entier donné
  - Nombre d'inversions ($i < j$ et $\pi[i] > \pi[j]$) d'une permutation
- Problèmes d'évaluation d'une fonction :
  - Produit de deux matrices
  - Plus grand commun diviseur (pgcd) de deux entiers

Problème de la fouille

Soit x une valeur entière et Tab un tableau d'entiers dont les indices sont compris entre 1 et n. Existe-t-il un indice i compris entre 1 et n tel que $x = Tab[i]$?

- Entrées :
  - x une valeur entière
  - Tab un tableau d'entiers indicé de 1 à n
- Sortie :
  - Booléen
- Taille du problème : n (taille du tableau)
- Exemplaire du problème : $(7, [3, 7, 10, 67, 25, 100])$

$$P(7, [3, 7, 10, 67, 25, 100]) = vrai$$

Problème de l'accessibilité

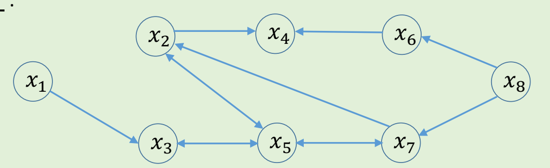
Soit $G = (S, A)$ un graphe orienté. Soit $s, t \in S$, le sommet t est-il accessible depuis le sommet s ?

- Entrées :
  - G un graphe
  - s un sommet
  - t un sommet
- Sortie :
  - Booléen
- Taille du problème : $|S|$ et $|A|$

Exemplaire du problème de l'accessibilité

Graphe G :

$|S| = 8$
$|A| = 12$



$P(G, x_1, x_4) = vrai$

$P(G, x_6, x_2) = faux$

Problèmes de la satisfaisabilité (SAT)

Définition : Un littéral est une expression booléenne qui est soit une formule atomique (V, F ou une variable booléenne) soit la négation d'une formule atomique.

Exemples : $V, F, x, \neg y$

Rappels :

Conjonction : $x \wedge y$

x \ y	F	V
F	F	F
V	F	V

Négation : $\neg x$

x	F	V
$\neg x$	V	F

Disjonction : $x \vee y$

x \ y	F	V
F	F	V
V	V	V

Forme normale conjonctive

- Définition : Une expression booléenne ϕ est en **forme normale conjonctive** (FNC) si ϕ est de la forme $\bigwedge_{j=1}^{m} C_j$, où chaque **clause** C_j est une disjonction de littéraux.
- Exemples :
  - $(\neg x \vee y \vee z) \wedge (x \vee w) \wedge (x \vee y \vee v \vee \neg w)$
  - $(\neg x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6) \wedge (x_4 \vee \neg x_5)$
- Contre-exemples :
  - $\neg (x \vee y)$
  - $\big((x \wedge v) \vee (y \wedge z)\big) \wedge w$
- Toute expression booléenne est logiquement équivalente à une forme normale conjonctive :
  - $\neg (x \vee y) \equiv \neg x \wedge \neg y$
  - $\big((x \wedge v) \vee (y \wedge z)\big) \wedge w \equiv (x \vee y) \wedge (x \vee z) \wedge (v \vee y) \wedge (v \vee z) \wedge w$

Problème FNC-SAT

Soit ϕ une expression booléenne en FNC, existe-t-il une affectation des variables qui rende ϕ vraie ?

- Entrée : FNC ϕ
- Sortie : Booléen
- Taille du problème : n (le nombre de variables)
- Exemplaires du problème :
  - $P\big((\neg x \vee y) \wedge w \wedge (z \vee \neg w)\big) = vrai$, car, par exemple, l'affectation $w = V$, $z = V$, $y = V$ et $x = V$ rend ϕ vraie.
  - $P\big((\neg x \vee \neg z) \wedge w \wedge (z \vee \neg w) \wedge (\neg z \vee \neg w \vee x)\big) = faux$, car aucune affectation des variables ne rend ϕ vraie.

Problèmes k-SAT, $k \geq 2$

- Soit ϕ une expression booléenne en FNC, où chaque clause contient $k \geq 2$ littéraux (avec un nombre indéterminé de variables), existe-t-il une affectation des variables qui rende ϕ vraie ?
- Exemplaires du problème :
  - $\boldsymbol{k = 2}$: $P\big((x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)\big) = faux$, car aucune affectation des variables rend ϕ vraie.
  - $\boldsymbol{k = 3}$: $P\big((\neg z \vee \neg y \vee x) \wedge (\neg z \vee y \vee x) \wedge (\neg z \vee y \vee \neg x)\big) = vrai$, car, par exemple, l'affectation $x = V$, $y = V$, $z = V$ rend ϕ vraie.
- Remarques :
  - Il existe des algorithmes polynomiaux pour résoudre le problème 2-SAT.
  - Les problèmes k-SAT, $k \geq 3$ sont NP-complets.

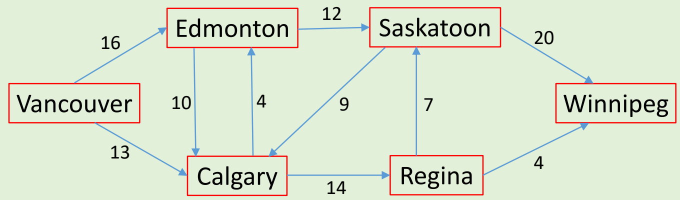
Problème du flot maximal

- <u>Définition</u> : Un réseau $R = (S, A, s, t, c)$ est un graphe orienté avec :
  - S : ensemble de sommets
  - A : ensemble d'arcs
  - $s \in S$: source
  - $t \in S$: puits
  - $c : A \longrightarrow \mathbb{N}$: fonction capacité

On suppose que pour tout sommet $v \in S$, il existe un chemin de s vers t passant par v.

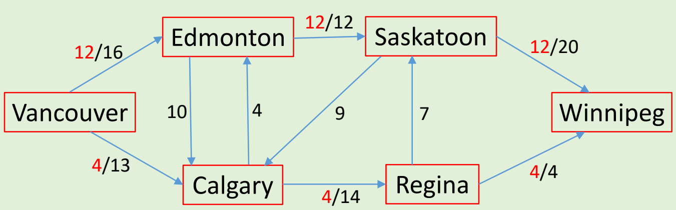
- On définit alors le flot f comme une fonction $f : A \longrightarrow \mathbb{N}$ qui vérifie :
  - $\forall a \in A, f(a) \leq c(a)$
  - $\forall i \in S$, tel que $i \neq s$ et $i \neq t$, on a $\sum_{(k,i)\in A} f(k,i) = \sum_{(i,j)\in A} f(i,j)$
- <u>Valeur du flot</u> : $|f| = \sum_{(s,j)\in A} f(s,j) = \sum_{(i,t)\in A} f(i,t)$
- <u>Problème</u> : Étant donné un réseau R, quelle est la valeur maximale d'un flot dans R ?

Exemple : nombre maximal de boîtes pouvant être livrées par jour

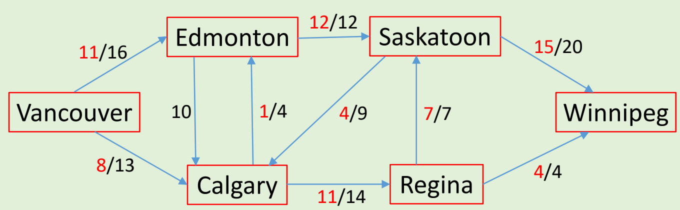


<u>Problème</u> : Quel est le nombre maximal de boîtes qui peuvent être livrées de Vancouver à Winnipeg par jour ?

Exemple 1 de flot : $|f| = 16$



Exemple 2 de flot : $|f| = 19$



Flot maximal : $|f| = 23$

Problème du commis voyageur

- <u>Définition</u> : Une permutation π de l'ensemble $[n] = \{1; 2; \dots; n\}$ est une fonction bijective de $[n]$ vers $[n]$.
- <u>Exemple</u> :

i	1	2	3	4	5	6	7	8	9
$\pi(i)$	2	7	3	8	5	6	4	9	1

- <u>Entrées</u> :
  - $V = \{1; 2; \dots; n\}$ un ensemble de villes
  - $d : V \times V \to \mathbb{N}$, où $d_{i,j}$ est la distance entre les villes i et j ($d_{i,j} = d_{j,i}$ et $d_{i,i} = 0$)
- <u>Sortie</u> :
  - Une permutation π de $[n]$ qui minimise $d = d_{\pi(n),\pi(1)} + \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)}$
- Le problème revient donc à minimiser la distance totale d'une « tournée », c'est-à-dire d'un itinéraire qui passe une et une seule fois par chaque ville et qui revient à la ville de départ à la fin.

Problème de la sélection

- <u>Entrées</u> :
  - Tab un tableau d'entiers indicé de 1 à n
  - i un entier entre 1 et n
- <u>Sortie</u> : L'entier x qui vérifie toutes les propriétés suivantes :
  - x est un élément de Tab
  - $i - 1$ éléments de Tab sont inférieurs ou égaux à x
  - $n - i$ éléments de Tab sont supérieurs ou égaux à x
- <u>Taille du problème</u> : n (taille du tableau)
- <u>Exemplaire du problème</u> : $([3, 7, 10, 12, 25, 11, 45], 4)$

  $P([3, 7, 10, 12, 25, 11, 45], 4) = 11$

Problème du tri

- Entrée : Tab un tableau d'entiers indicé de 1 à n
- Sortie : Tab' un tableau d'entiers indicé de 1 à n tel que :
  - Il existe une permutation π de $[n]$ telle que $Tab'[i] = Tab[\pi(i)]$ pour tout entier $1 \leq i \leq n$
  - $Tab'[i] \leq Tab'[i+1]$ pour tout entier $1 \leq i \leq n-1$

- Exemplaire du problème :

i	1	2	3	4	5	6	7	8	9
$Tab[i]$	91	5	13	74	65	55	11	28	89
$Tab'[i]$	5	11	13	28	55	65	74	89	91
$\pi(i)$	2	7	3	8	6	5	4	9	1

Problème de l'arrêt

- Entrées :
  - M : un programme
  - ω : une chaîne de caractères (une entrée pour le programme M)

- Sortie : un booléen :
  - *vrai* si l'exécution de M avec l'entrée ω provoque une boucle infinie
  - *faux* dans le cas contraire

- Ce problème n'est pas résoluble par un algorithme : il est **indécidable**.
  - Machines de Turing
  - Théorème d'incomplétude de Gödel