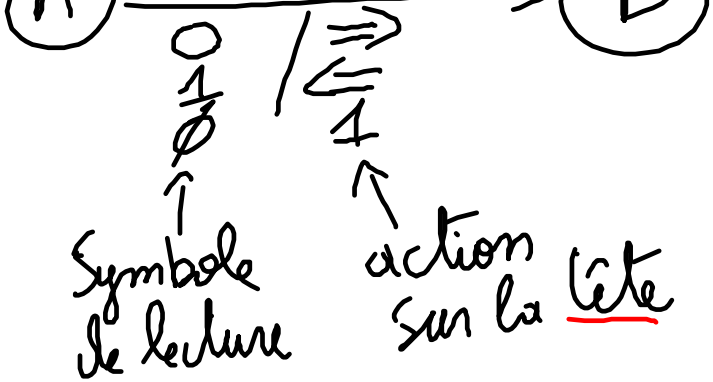
Machine de Turing

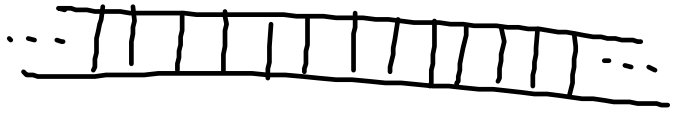
Graphe orienté
Bande

→ nœud = état
arcs = transition

○ : état initial ◎ : état final ⓞ : état final acceptant



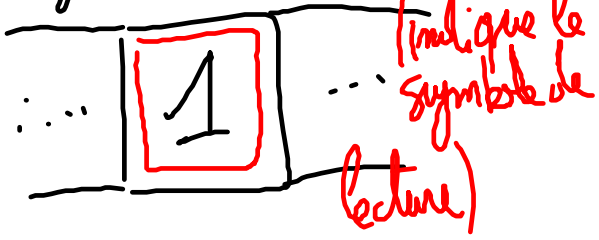
Suite infinie de cases



(éventuellement 2D ou semi-infinie)

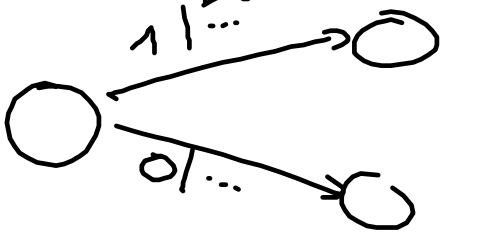
Sur chaque case : 1 symbole 0, 1 ou ∅

éventuellement une tête

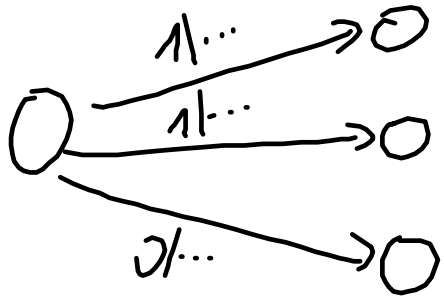


2 Types de Machines

Déterministe



Non-Déterministe

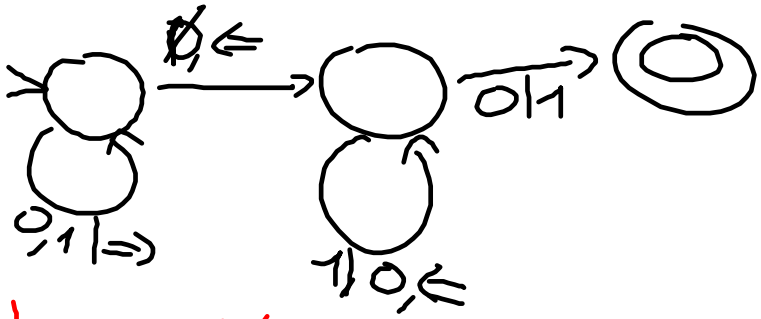


Toute machine non déterministe peut être simulée par une machine déterministe

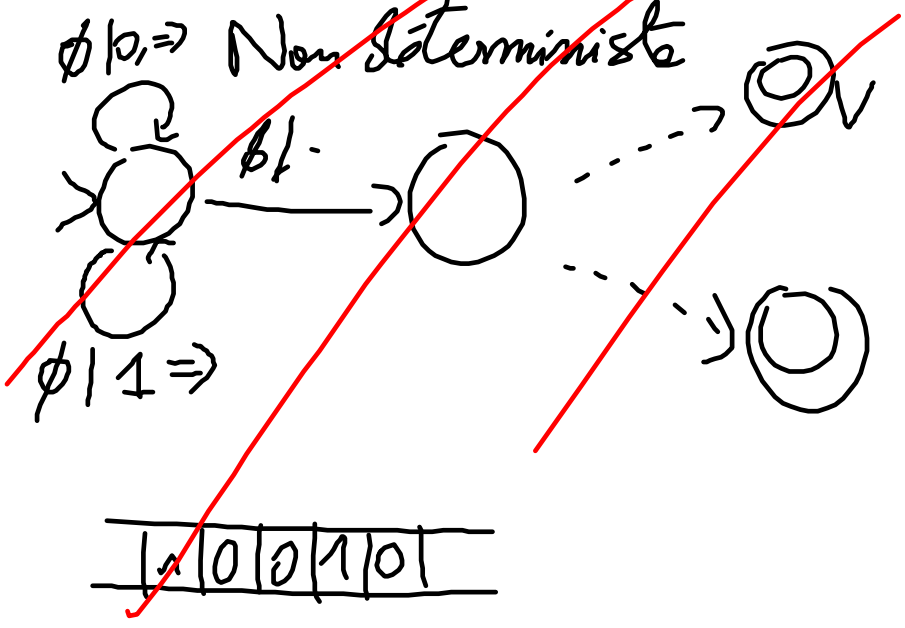
Pb de successeur

Soit $x \in \mathbb{N}$, que vaut $x+1$?

Machine déterministe



∅|0,⇒ Non déterministe



Le non déterminisme n'a de sens que pour les pb de Décision, dont la réponse est OUI ou NON.

Machine déterministe $\Longleftrightarrow$ langage de prog classique (C, java, python, ruby, ...)

Stratégie pour les preuves

- existe-t-il ~~une machine~~ / un algo pour résoudre tel problème ?
  $\exists$ algo
- n'existe-t-il pas une machine / ~~un algo~~ pour résoudre tel problème ?
  $\forall$ machine

Chapitre 3 : Les classes de complexité

ENSIIE - Théorie de la complexité

Dimitri Watel (dimitri.watel@ensiie.fr)

2018

Complexité du calcul d'une machine de Turing

machine + 1 entrée x

Définition

- La complexité en temps d'un calcul d'une machine de Turing est le nombre d'itérations nécessaires à la machine avant qu'elle ne s'arrête.
- La complexité en espace d'un calcul d'une machine de Turing est le nombre de cases mémoires sur lesquelles la machine a écrit avant qu'elle ne s'arrête.

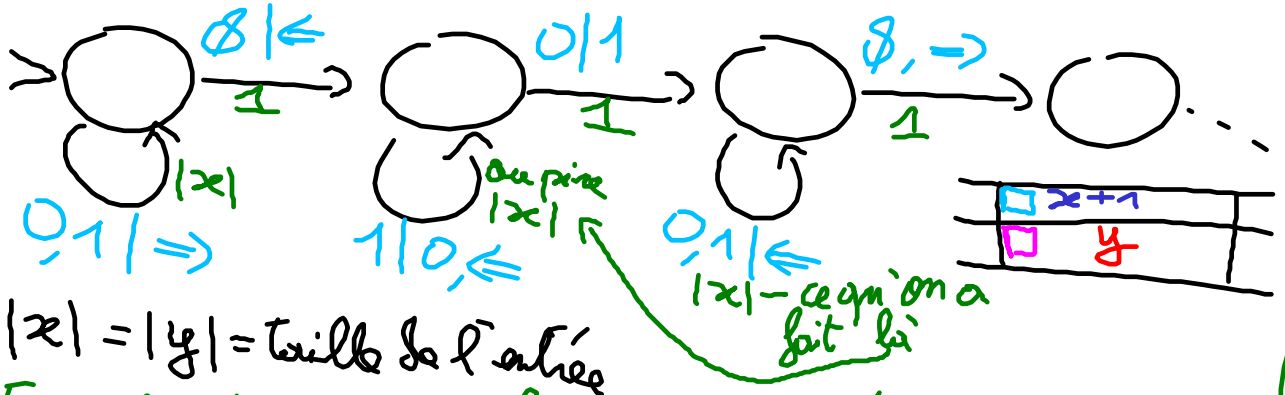
distinctes

(⚠ les cases non réécrites de x ne comptent pas)

La complexité est souvent calculée en fonction de la taille de x.

Exemple

Soit x et y, est ce que $y = x+1$?

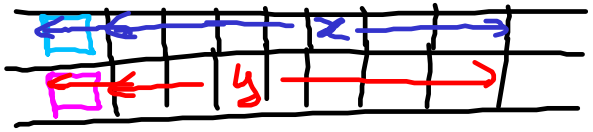


$|x| = |y|$ = taille de l'entrée

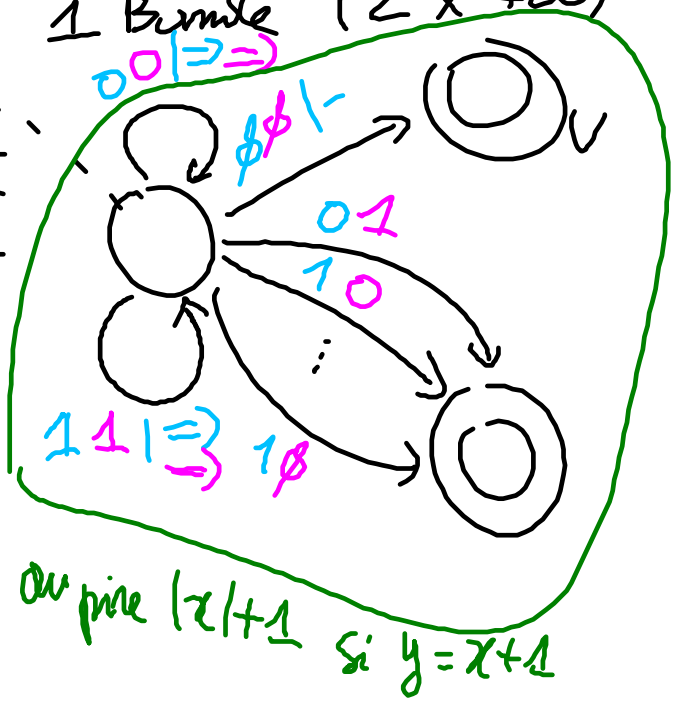
En vert : nb de fois que la transition est tirée.

Ex : $x = 10010$ Temps = $\underbrace{5+1+5+2}_{x \leftarrow x+1} + \underbrace{3}_{x == y ?}$

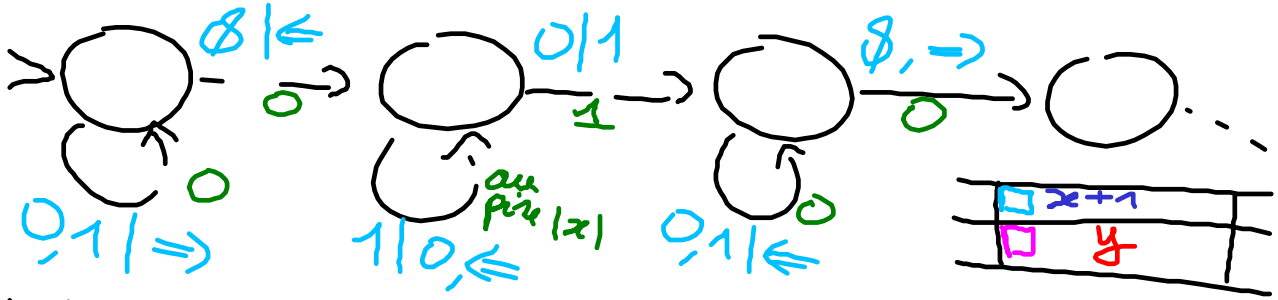
$y = 10110$

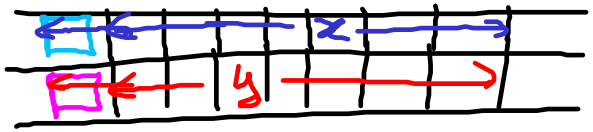


2 bits,
1 Bande ($2 \times +\infty$)

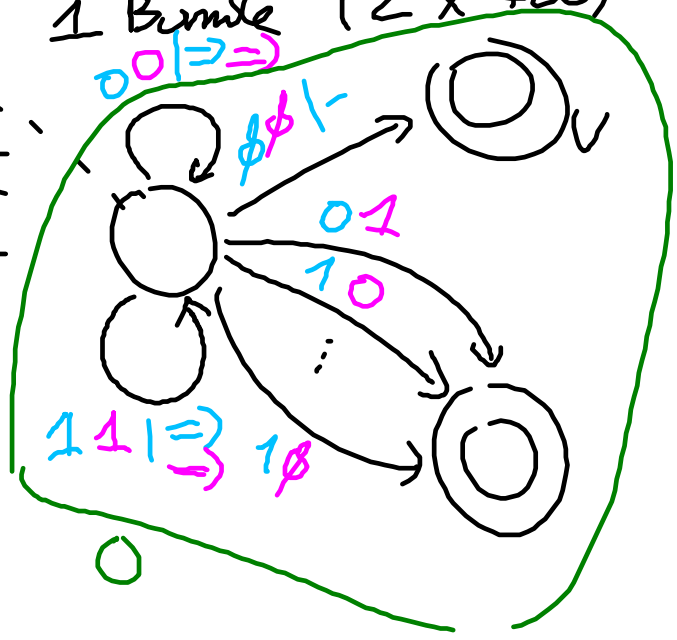


Exemple
Soit x et y, est ce que $y = x+1$?





2 bits,
1 Bande (2 x +∞)



$|x| = |y|$ = taille de l'entrée

En vert : nb de cases distinctes écrites par la transition

Ex : $x = 10010$ Espace = 1
$y = 10100$ (un 0 devient un 1)

Complexité du calcul d'une machine de Turing

Définition

Soit une machine $\mathcal{M}$ **déterministe** et un mot x, on pose $t(\mathcal{M}, x)$ la complexité en temps du calcul de $\mathcal{M}$ quand l'entrée est x. De même, on pose $s(\mathcal{M}, x)$ sa complexité en espace.

Définition

Soit une machine $\mathcal{M}$ **non-déterministe**, un mot x et une suite de choix C, on pose $t(\mathcal{M}, x, C)$ la complexité en temps du calcul de $\mathcal{M}$ quand l'entrée est x et que la machine fait les choix C. De même, on pose $s(\mathcal{M}, x, C)$ sa complexité en espace.

Complexité dans le pire cas

Définition

Soit une machine $\mathcal{M}$ **déterministe**, la complexité (en temps) dans le pire cas de $\mathcal{M}$ est une fonction f telle que

$$f(n) = \max_{x \in \{0,1,\#\}^n} (t(\mathcal{M}, x))$$

Définition

Soit une machine $\mathcal{M}$ **non-déterministe**, la complexité (en temps) dans le pire cas de $\mathcal{M}$ est une fonction f telle que

$$f(n) = \max_{x \in \{0,1,\#\}^n} \min_{\text{Choix } C} (t(\mathcal{M}, x, C))$$

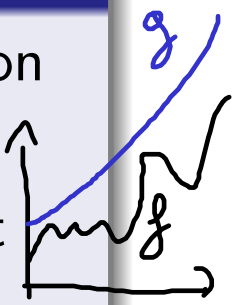
On définit de même la complexité en espace dans le pire cas.

Complexité asymptotique dans le pire cas

Définition

Soit une machine $\mathcal{M}$ dont la complexité dans le pire cas est f, on dit que $\mathcal{M}$

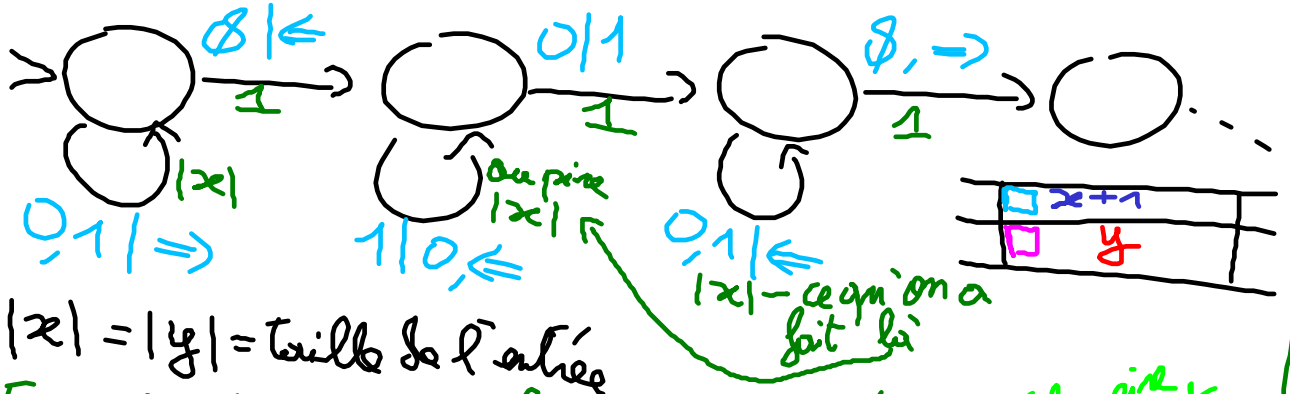
- a une complexité dans le pire cas bornée asymptotiquement par g si $f(n) = O(g(n))$ quand $n \to +\infty$.
- a une complexité dans le pire cas minorée asymptotiquement par g si $f(n) = \Omega(g(n))$ quand $n \to +\infty$.
- a une complexité dans le pire cas asymptotiquement de l'ordre de g si $f(n) = \Theta(g(n))$ quand $n \to +\infty$.



Par abus de langage, on dira que $\mathcal{M}$ a une complexité $O(g(n))$ si la **complexité en temps dans le pire cas** de $\mathcal{M}$ est bornée asymptotiquement par g.

Exemple

Soit x et y, est ce que $y = x+1$?



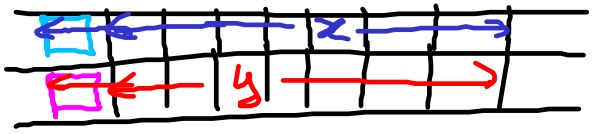
$|x| = |y|$ = taille de l'entrée

En vert : nb de fois que la transition est tirée.

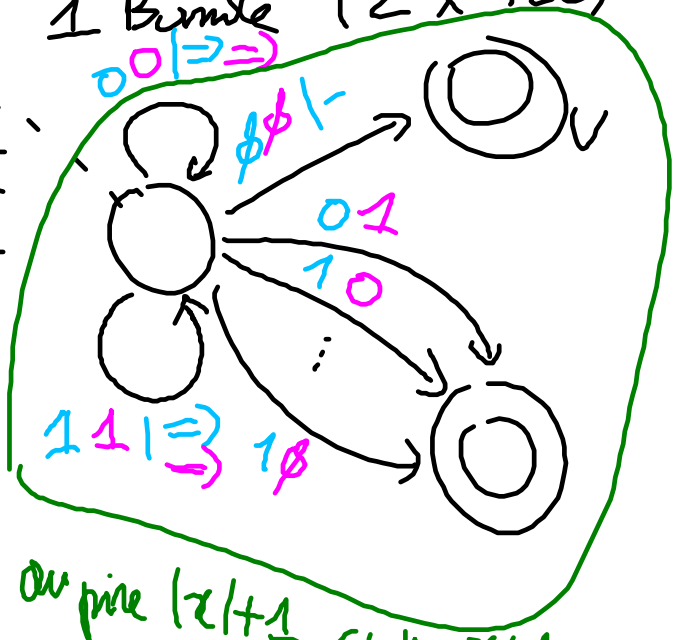
Complexité en temps : $f(|x|,|y|) = 3|x| + 4 = O(|x|)$

(au pire on entre dans $011\ldots1$ ou $100\ldots0$)

Existe en espace $= O(|x|)$



2 bits,
1 Bande $(2 \times +\infty)$



au pire $|x|+1$ si $y = x+1$

Problème de décision

Définition (formelle)

Un problème de décision Π est constitué d'un ensemble $\mathcal{L}$, dont les éléments sont appelés *instances* ou *entrées*, et d'un ensemble $\mathcal{L}_Y \subset \mathcal{L}$ d'*instances positives*. On nomme $\mathcal{L} \backslash \mathcal{L}_Y = \mathcal{L}_N$ les *instances négatives*.

Résoudre un problème avec une machine de Turing déterministe

Une machine de Turing **déterministe** $\mathcal{M}$ résoud Π si elle s'arrête pour toute entrée et

- quand $x \in \mathcal{L}_Y$, $\mathcal{M}$ accepte x ;
- quand $x \in \mathcal{L}_N$ ou $x \notin \mathcal{L}$, $\mathcal{M}$ refuse x.

Complexité d'un problème de décision

Définition : DTIME

Soit un problème de décision Π, alors Π a une complexité (au pire) $O(f(n))$ s'il existe une machine de Turing **déterministe** de complexité $O(f(n))$ résolvant Π. On écrit $\Pi \in \text{DTIME}(f(n))$.

Pb Le Plus court chemin : $\text{DTIME}(n^2)$ (mais aussi $\text{DTIME}(m + n\log(n))$ où m = nb arêtes.)

Sac à dos : $\text{DTIME}(2^n)$ (n = nb d'objets) ou $\text{DTIME}(n \times B)$ (B = taille du sac)

Soient 2 grammaires, génèrent-elle les même mots ?

$\text{DTIME}(2^{2^n})$ (?)

La classe de complexité P

Définition de la classe P

Un problème de décision Π est dit polynomial ou appartenant à la classe P si sa complexité est polynomiale. Autrement dit, il existe une machine de Turing **déterministe** qui résoud Π en temps polynomial.

$$\mathrm{P} = \bigcup_{c \in \mathbb{N}} \mathrm{DTIME}(n^c)$$

Définition de la classe EXPTIME

Un problème de décision Π est dit exponentiel ou appartenant à la classe EXPTIME si sa complexité est exponentielle. Autrement dit, il existe une machine de Turing **déterministe** qui résoud Π en temps exponentiel :

$$\text{EXPTIME} = \bigcup_{c \in \mathbb{N}} \text{DTIME}(2^{(n^c)})$$

inclu 3^n

$n!$ n^n

n'inclue pas (je crois)

$$\text{P} \subsetneq \text{EXPTIME}$$

Complexité en espace d'un problème de décision

Définition : DSPACE

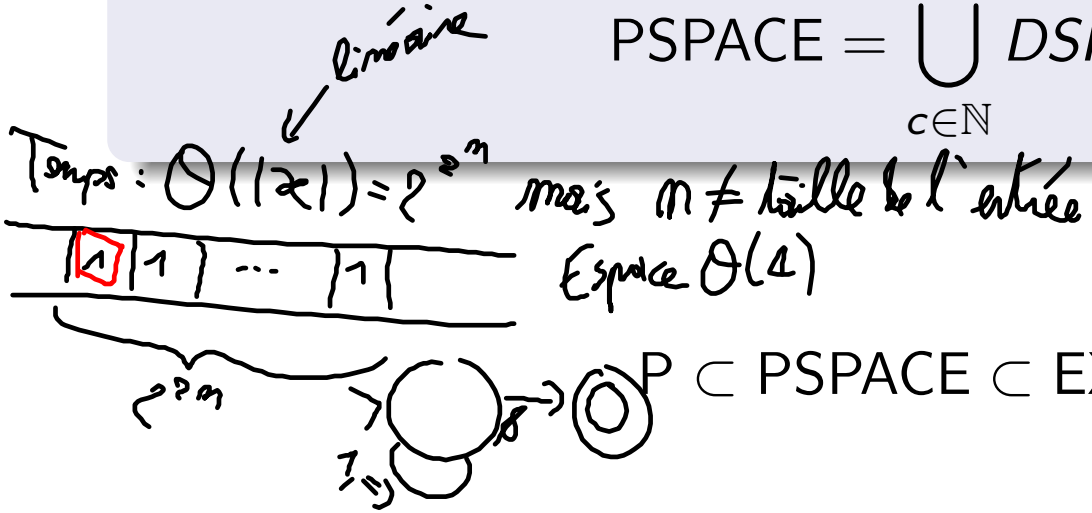
Soit un problème de décision Π, alors Π a une complexité en espace (au pire) $O(f(n))$ s'il existe une machine de Turing **déterministe** de complexité en espace $O(f(n))$ résolvant Π. On écrit $\Pi \in \text{DSPACE}(f(n))$.

La classe de complexité PSPACE

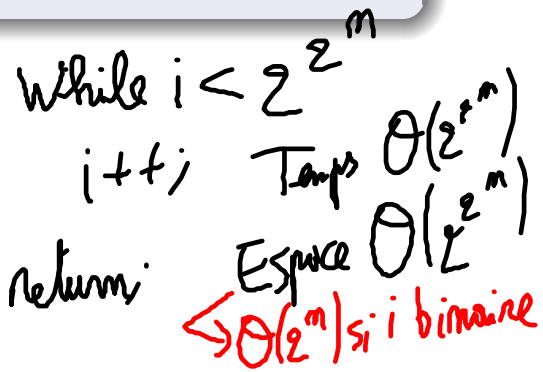
Définition de la classe PSPACE

Un problème de décision Π appartient à la classe PSPACE si sa complexité en espace est polynomiale. Autrement dit, il existe une machine de Turing **déterministe** qui résoud Π en espace polynomial.

$$\text{PSPACE} = \bigcup_{c \in \mathbb{N}} DSPACE(n^c)$$



$$\text{P} \subset \text{PSPACE} \subset \text{EXPTIME}$$



La classe de complexité EXPSPACE

Définition de la classe EXPSPACE

Un problème de décision Π appartient à la classe EXPSPACE si sa complexité en espace est exponentielle. Autrement dit, il existe une machine de Turing **déterministe** qui résoud Π en espace exponentiel :

$$\text{EXPSPACE} = \bigcup_{c \in \mathbb{N}} \text{DSPACE}(2^{n^c})$$

$$\text{PSPACE} \subsetneq \text{EXPSPACE}$$

$$\text{P} \subset \text{PSPACE} \subset \text{EXPTIME} \subset \text{EXPSPACE}$$

Problème de décision

Définition (formelle)

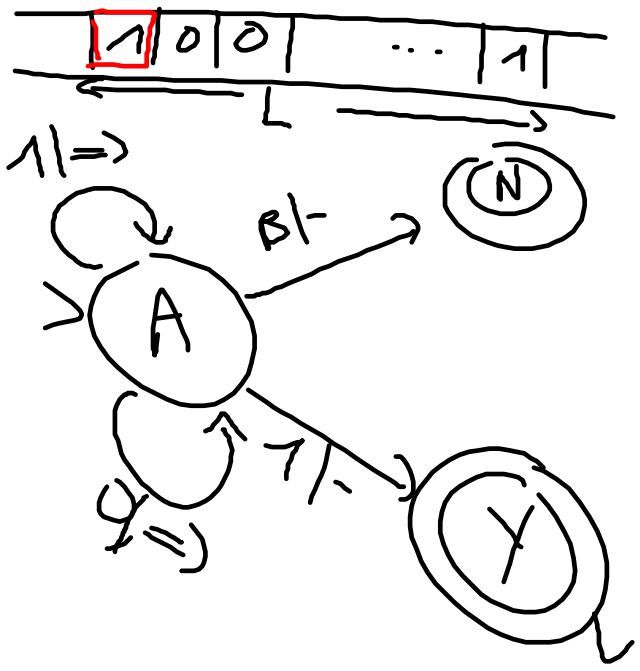
Un problème de décision Π est constitué d'un ensemble $\mathcal{L}$, dont les éléments sont appelés *instances* ou *entrées*, et d'un ensemble $\mathcal{L}_Y \subset \mathcal{L}$ d'*instances positives*. On nomme $\mathcal{L} \backslash \mathcal{L}_Y = \mathcal{L}_N$ les *instances négatives*.

Résoudre un problème avec une machine de Turing non déterministe

Une machine de Turing **non déterministe** $\mathcal{M}$ résoud Π si

- quand $x \in \mathcal{L}_Y$, $\mathcal{M}$ accepte x ; $\Leftrightarrow \exists$ choix / $\mathcal{M}$ accepte x.
- quand $x \in \mathcal{L}_N$ ou $x \notin \mathcal{L}$, $\mathcal{M}$ refuse fortement x $\Leftrightarrow \forall$ choix $\mathcal{M}$ refuse x

Soit $L \in \{0;1\}^n$, est-ce que $1 \in L$?



Si $L = 0^n$, $\cap$ refuse L. → ∀ choix $\cap$ refuse L

Si : $1 \in L$, et que la machine utilise (A, Y) quand la tête est dessus, $\cap$ accepte L.

↘ ∃ choix / $\cap$ accepte L

$\Rightarrow$ $\cap$ résout le problème.

SAT : Soit φ une formule avec n variables $x_1 x_2 \dots x_n$
peut-on rendre φ vrai ?

Déterministe

$\forall a_1 a_2 \dots a_n \in \{\top, \bot\}^n$ $\quad \Theta(2^n)$

φ est-elle vraie si on affecte x_i avec a_i ?

Si oui → return OUI

return NON.

Non Déterministe

$\forall i \in [\![1, n]\!]$ $\quad \Theta(n)$

$a_i \leftarrow \top$ ou $\bot$ (Non Déterministe)

return φ vraie si on affecte x_i avec a_i ?

Si φ est une contradiction quel que soient les choix de a_i $\Rightarrow$ Π refuse φ fortement

Sinon, $\exists$ au moins une affectation a_i de x_i qui satisfait φ et qui peut être choisie par Π : Π accepte φ

Complexité non déterministe d'un problème de décision

Définition : NTIME

Soit un problème de décision Π, alors $\Pi \in \text{NTIME}(f(n))$ s'il existe une machine de Turing **non déterministe** $\mathcal{M}$ de complexité $O(f(n))$ qui résoud Π.

La classe de complexité NP

Définition de la classe NP (non-deterministic Polynomial)

Un problème de décision Π appartient à la classe NP s'il existe une machine de Turing **non déterministe** qui résoud Π en temps polynomial.

$$\text{NP} = \bigcup_{c \in \mathbb{N}} \text{NTIME}(n^c)$$

SAT $\in$ NP

Sac à dos $\in$ NP

Plus court chemin $\in$ P $\subset$ NP

$$\text{P} \subset \text{NP} \subset \text{PSPACE}$$

La classe de complexité NEXPTIME

Définition de la classe NEXPTIME

Un problème de décision Π appartient à la classe NEXPTIME s'il existe une machine de Turing **non déterministe** qui résoud Π en temps exponentiel :

$$\text{NEXPTIME} = \bigcup_{c \in \mathbb{N}} \text{NTIME}(2^{n^c})$$

$$\text{EXPTIME} \subset \text{NEXPTIME} \subset \text{EXPSPACE}$$

$$\text{NP} \subsetneq \text{NEXPTIME}$$

Complexité en espace non déterministe d'un problème de décision

Définition : NSPACE

Soit un problème de décision Π, alors $\Pi \in \text{NSPACE}(f(n))$ s'il existe une machine de Turing **non déterministe** $\mathcal{M}$ de complexité en espace $O(f(n))$ qui résoud Π.

La classe de complexité NPSPACE

Définition de la classe NPSPACE

Un problème de décision Π appartient à la classe NPSPACE s'il existe une machine de Turing **non déterministe** qui résoud Π en espace polynomial.

$$\text{NPSPACE} = \bigcup_{c \in \mathbb{N}} \text{NSPACE}(n^c)$$

!!!

$$\text{PSPACE} = \text{NPSPACE}$$

$\subseteq$: Trivial

$\supseteq$: parce que $\text{NSPACE}(g(n)) \subseteq \text{DSPACE}(g^2(n))$

La classe de complexité NEXPSPACE

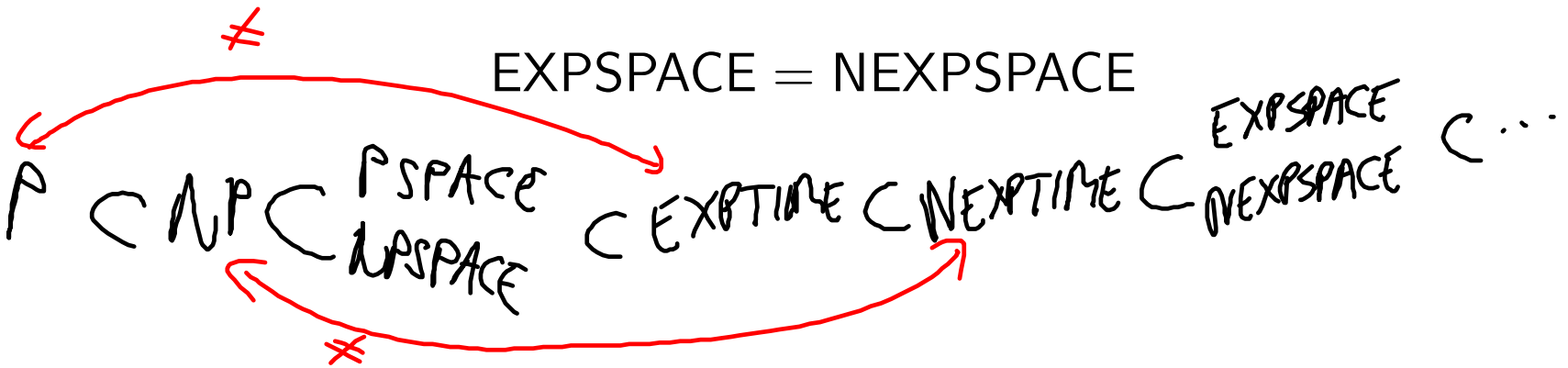
Définition de la classe NEXPSPACE

Un problème de décision Π est appartient à la classe NEXPSPACE s'il existe une machine de Turing **non déterministe** qui résoud Π en espace exponentiel :

$$\text{NEXPSPACE} = \bigcup_{c \in \mathbb{N}} \text{NSPACE}(2^{n^c})$$

!!!

$$\text{EXPSPACE} = \text{NEXPSPACE}$$



La classe de complexité Co-NP

Définition de la classe Co-NP

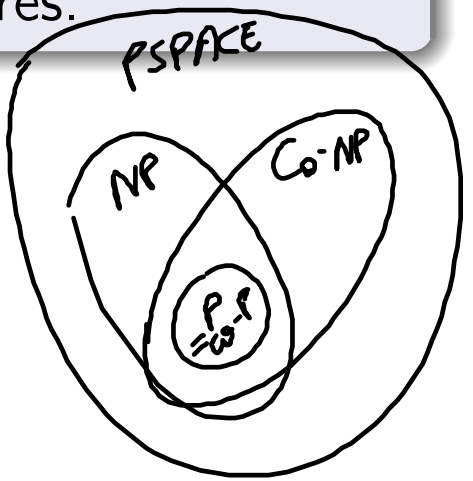
Un problème de décision $\Pi = (\mathcal{L}, \mathcal{L}_Y, \mathcal{L}_N)$ appartient à la classe Co-NP si $\Pi^c = (\mathcal{L}, \mathcal{L}_N, \mathcal{L}_Y)$ appartient à NP.

Autre définition de la classe Co-NP

Un problème de décision Π appartient à la classe Co-NP s'il existe une machine $\mathcal{M}$ **non déterministe** de complexité polynomiale qui accepte fortement les mots de $\mathcal{L}_Y$ et refuse les autres.

$$P \subset NP \cap Co\text{-}NP$$

$$Co\text{-}NP \subset PSPACE$$



Et tant d'autres

- Hiérarchie polynomiale : $\Sigma_2, \Pi_2, \Sigma_k, \Pi_k, PH$
- Classes probabiliste : BPP, ZPP, RP
- Classes quantiques : BQP, EQP
- Classes non décidables : RE, ALL

`https://complexityzoo.uwaterloo.ca/Complexity_Zoo`

Π : Soit $j_1 j_2 \dots j_n$ les tâches
j_i prend un temps
j_2 ——————
t_{i1} sur la machine 1
t_{i2} sur la machine 2

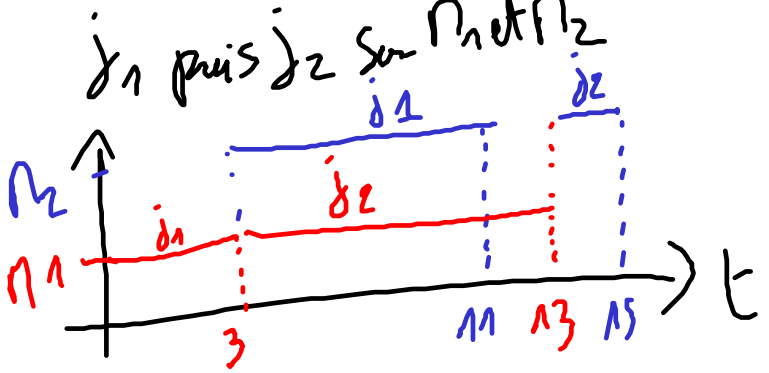
Les 2 machines ne peuvent pas travailler en même temps sur le même job.

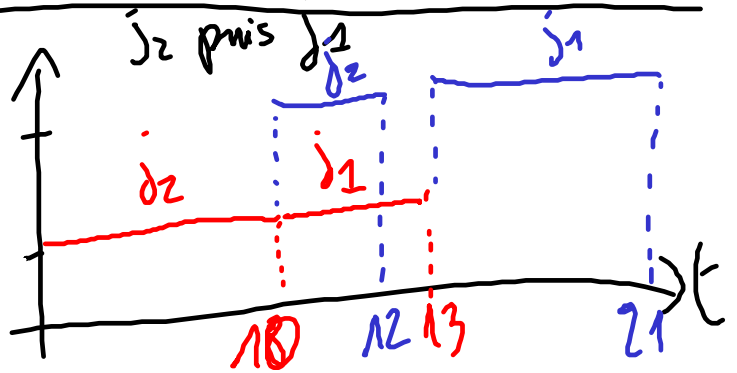
Ordonner $j_1 j_2 \dots j_n$ sur Π_1 et Π_2 de sorte à ce que le traitement prenne un temps minimum.

Chaque job doit passer sur les 2 machines, Π_1 puis Π_2.

$n=2$

t	j_1	j_2
Π_1	3	6
Π_2	8	2





Pb de décision : soit $K \in \mathbb{N}$, $\exists$? un ordre des jobs sur Π_1 et Π_2 tq le temps total de l'ordonnancement soit $\leq$ à K ? ← Pb (ORDO)

(ORDO) $\in$ NP ? OUI

Algo de Johnson : (ORDO) $\in$ P

NON DET
Calculer un ordre des jobs sur Π_1 et Π_2
certificat $O(n^2)$

DET
Vérifier si le temps de l'ordre est $\leq$ à K.
→ OUI (facile)
→ NON (dur)
Vérifier $O(n)$ (faire le dessin)

Soit A une matrice
b un vecteur

$\exists x \; / \; Ax = b$?