on a discuté de ce qui était calculable

mais ce calcul est-il efficace ?

complexité en temps

complexité d'un run = nombre d'étapes du calcul

$c_0 \vdash c_1 \dots \vdash c_m$

complexité m

complexité d'une machine

soit M MT qui s'arrête toujours — on parle de décideur

la complexité t_M de M est une fonction sur $\mathbb{N}$ tq $t_M(n)$ est le plus petit entier tq M(x) s'arrête en moins de $t_M(n)$ étapes sur toute entrée x de taille $\leq n$

classes de complexité

soit M MT à k rubans

si $t_M = \mathcal{O}(f)$ alors on écrit

$\mathcal{L}(M) \in DTIME_k(f)$

on note $DTIME_*(f) = \bigcup_{k \geq 1} DTIME_k(f)$

classe des langages reconnaissables en temps $\mathcal{O}(f)$

c'est un majorant

fonctions constructibles en temps

$f : \mathbb{N} \longrightarrow \mathbb{N}$ est constructible en temps ssi

$\exists$ M MT à un ou plusieurs rubans telle que

. $M(1^n)$ accepte avec la sortie $1^{f(n)}$

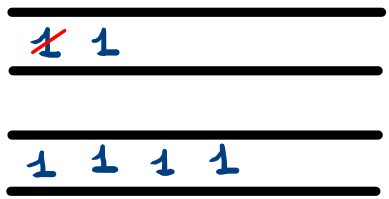
. $t_M = O(f)$ complexité bornée par la taille de la sortie

ici en unaire ; marche aussi en binaire

équivalence des deux définitions

$f(n) = n^2$ constructible en temps

idée : M à 2 rubans



copier le 1er ruban n fois en "barrant" un 1 en haut à chaque étape

$\Theta(n) \times n$ $\Theta(n)$ $= \Theta(n^2)$

$f(n) = 2^n$ constructible en temps

on utilise 3 rubans et l'algo suivant :

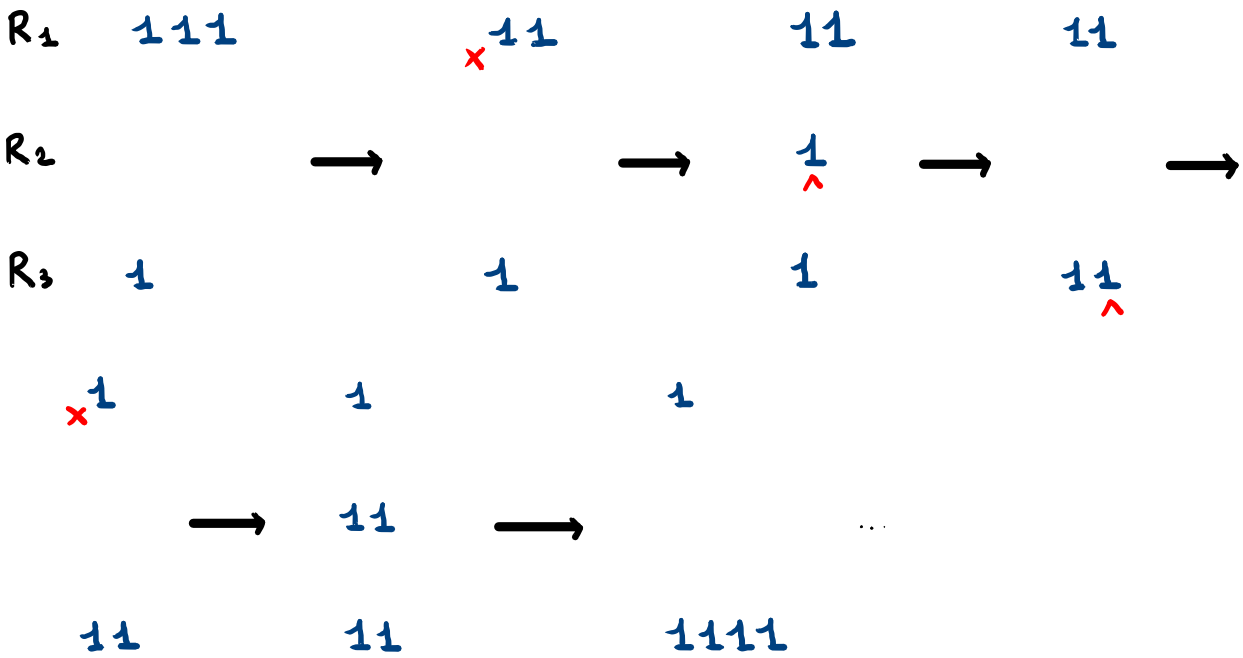
. écrire 1 sur R_3

. puis tant qu'on peut effacer un 1 de R_1

  . copier R_3 vers R_2

  . concaténer R_2 à la fin de R_3 puis effacer R_2

  . puis rembobiner la tête de R_3



R_3 double à chaque boucle et une boucle est linéaire en R_3 courant

hypothèse de récurrence : R_3 contient 2^{i-1}

au début de la i-ème étape

$\alpha \times 2^{i-1}$ opérations par étape

complexité totale de l'ordre de

$$\sum_{i=1}^{n} 2^{i-1} = 2^n - 1$$

transformation de machines

n étapes d'une machine à k rubans

$\longrightarrow$ $\mathcal{O}(n^2)$ étapes dans la MT simple équivalente

en effet, au bout de i étapes dans la MT simple, ruban de taille max. $n + i$

$\alpha \times (n + i)$ opérations par étape i

total en $\mathcal{O}(n^2)$

pour simuler une étape de M, la machine universelle $\mathcal{U}$ prend $\Theta(|\langle M \rangle|)$ étapes

en effet à chaque étape il faut retrouver la bonne transition dans $\langle M \rangle$.

et pour le passage en linéaire sur k bits

une étape $\longrightarrow$ $3k$ étapes

revoir les constructions

classes connues

temps polynomial

$$P = PTIME = \bigcup_{k \geq 1} DTIME_*(n \to n^k)$$

temps exponentiel

$$EXP = EXPTIME = \bigcup_{k \geq 1} DTIME_*(n \to 2^{n^k})$$

préservées par les transformations précédentes

exemples

RPRIME = { (x, y) | x et y premiers entre eux }

$\in P$ par l'algo d'

exercice : calculer x mod y en unaire

puis en déduire une MT qui implémente

Euclide et estimer sa complexité

CONNECTED = { G | G graphe connexe }

$\in P$ par des parcours en profondeur.

exercice : comment encoder G ?

comment déterminer si 2 sommets sont liés ?

comment faire un DFS ?

conclure.

analyser la complexité

propriétés

$P \subseteq EXP$ rappel : classes = majoration

stricte ?

P et EXP sont stables par $\cup$ $\cap$ ^{c}

voir la preuve pour R et RE

théorème de hiérarchie en temps

soient f et g constructibles en temps tq

$$(n \longrightarrow n) = o(g)$$

$$f = o\left(n \longrightarrow \frac{g(n)}{n}\right)$$

alors

$$DTIME_*(f) \subset DTIME_*(g)$$

stricte

généralisable à $f = o\left(n \longrightarrow \frac{g(n)}{\log(n)}\right)$

conséquence

$P \subset EXP$

stricte

considérer $f(n) = 2^n$

$g(n) = 2^{3n}$

on a

$DTIME_*(f) \subset DTIME_*(g)$

$P \subsetneq \subsetneq EXP$