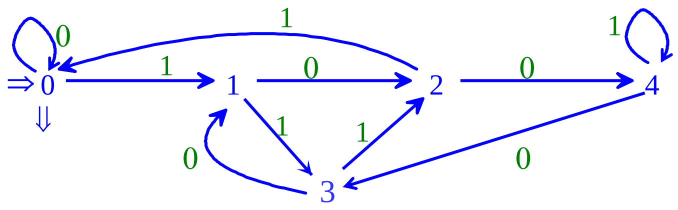
UMLV ©

Automates déterministes

$a = (Q, A, i, T, \delta)$

Q	états	ensemble fini
A	alphabet	ensemble fini
i	état initial	$i \in Q$
T	états terminaux	$T \subseteq Q$
δ	fonction de transition	$\delta : Q \times A \to Q$



$L(a) = \{ x \in A^* / x$ étiquette d'un chemin de i à $t \in T \}$

L = { écritures en base 2 des multiples de 5 }

UMLV ©

Algorithmes sur automates

- émondage
- calcul de $L(a)$
- expression rationnelle $\to$ automate
- déterminisation
- minimisation
- tests « $L(a) = \varnothing$? » ou « $L(a) = A^*$? »
- Construction d 'un automate a tel que
  - $L(a) = A^* - L(B)$
  - $L(a) = L(B) \cup L(C)$
  - $L(a) = L(B) \cap L(C)$
  - $L(a) = L(B)^*$
  - ...

tester $a = B$

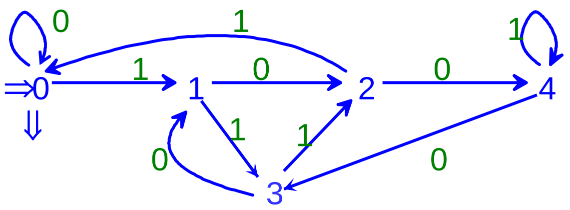
- test s 'équivalence : $L(a) = L(B)$?
- ...

Représentation des automates

Représentation de la fonction de transition δ par

- listes des flèches
- matrice d 'adjacence
- listes des successeurs
- table de transition

δ	0	1
0	0	1
1	2	3
2	4	0
3	1	2
4	3	1



Utilisation d 'un automate

Test d 'appartenance de x à $L(a)$, $a = (Q, A, i, T, \delta)$

fonction reconnaît (mot x, automate (Q, A, i, T, δ)) : booléen;
début $p \leftarrow i$;
  tant que non fin de x **faire {**
    $a \leftarrow$ lettre suivant de x ;
    $p \leftarrow \delta$ (p, a) ;
  }
  si $p \in T$ **retour** vrai **sinon** faux ;
fin

reconnaissance de **0 0 0 1 0 1 0 0** ($10100_2 = 20_{10}$)

a		0	0	0	1	0	1	0	0
p	0	0	0	0	1	2	0	0	0

δ	0	1
0	0	1
1	2	3
2	4	0
3	1	2
4	3	1

Automate minimal

UMLV ©

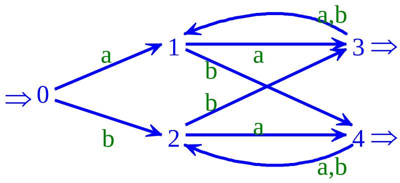
$a = (Q, A, i, T, \delta)$ automate déterministe

$p \in Q$ $L(p) = L(\,(Q, A, p, T, \delta)\,)$

Congruence syntaxique Soit E équivalence sur Q définie par

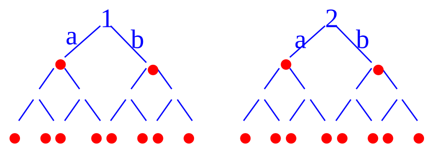
$(p,q) \in E \leftrightarrow L(p) = L(q)$

a/E automate minimal de $L(a)$.



$L(3) = L(4) = (A^2)^*$

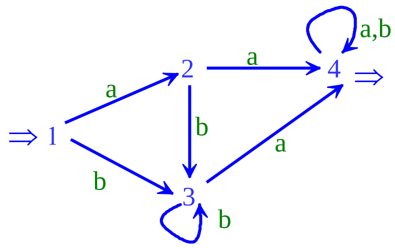
$L(1) = L(2) = A(A^2)^*$



Test de minimalité ?

UMLV ©

S 'il existe p,q tels que : $\forall\, a \in A \;\; \delta(p,a) = \delta(q,a)$
alors a non minimal



2 et 3 équivalents

Réciproque fausse !

Est-il plus facile de

- tester si a est minimal

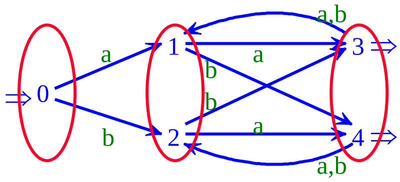
que de

- minimiser a ?

Minimisation par partitionnement

Lemme E est l'équivalence la plus grossière telle que :

- $(p,q) \in E$ et $p \in T \Rightarrow q \in T$ compatibilité avec T
- $(p,q) \in E \Rightarrow (\delta(p,a), \delta(q,a)) \in E$ compatibilité avec δ



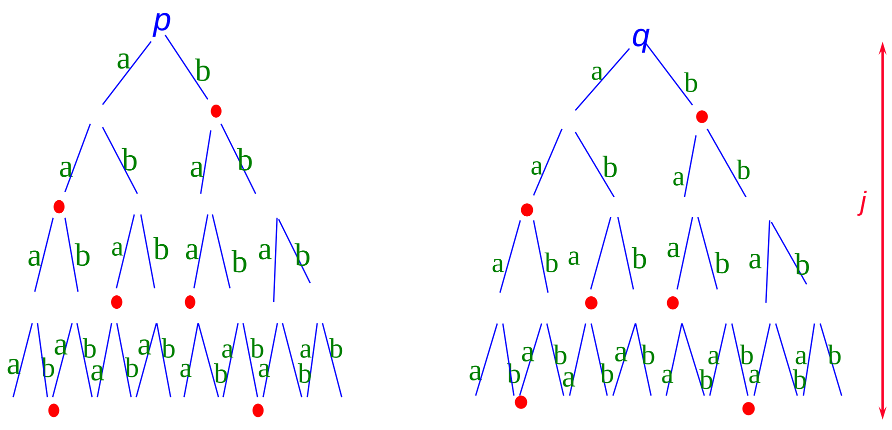
F équivalence initiale = $(T \times T) \cup ((Q-T) \times (Q-T))$

E_j équivalence : $(p,q) \in E_j \Leftrightarrow$ pour tout $u \in A^{\leq j}$ $(\delta(p,u), \delta(q,u)) \in F$

Lemme il existe $k \geq 0$

$E_0 = F > E_1 > \dots > E_k = E_{k+1} = E$

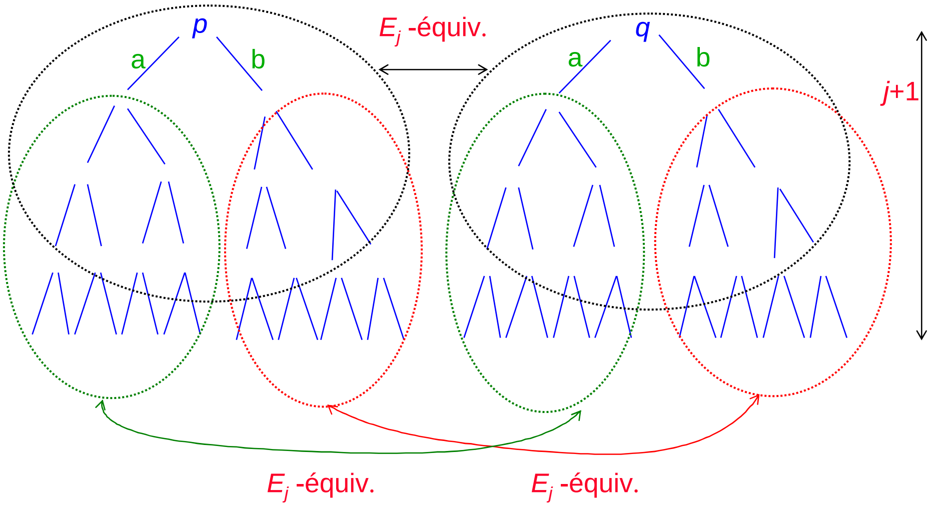
p E_j équivalent à q



UMLV ©

Lemme de récurrence

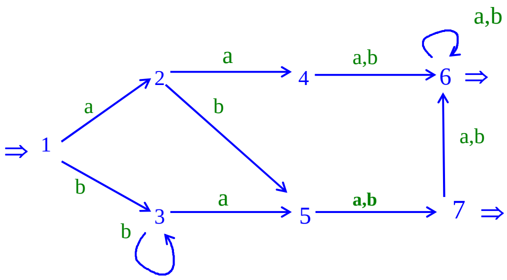
$(p,q) \in E_{j+1}$ ssi $\forall\, a \in A\ (\delta(p,a), \delta(q,a)) \in E_j$ et $(p,q) \in E_j$



Algorithme de Moore

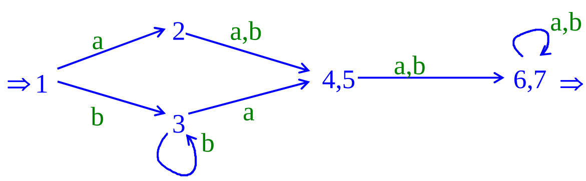
UMLV ©

Calcul de la congruence syntaxique :
raffiner F en utilisant la récurrence jusqu'à stabilisation



$F = E_0$:	1	2	3	4	5 |	6	7
E_1 :	1	2	3 |	4	5 |	6	7
E_2 :	1 |	2 |	3 |	4	5 |	6	7

Automate minimal



Mise en œuvre

UMLV ©

Automate donné par

δ	a ... z		F
1			
2			
.			
.			
n			

Calcul de E_{j+1} à partir de E_j

	E_j	$E_j(\delta(.,a))$ ... $E_j(\delta(.,z))$		E_{j+1}
1				
2			Calcul des lignes égales $\longrightarrow$ par tri lexicographique	
.				
n				

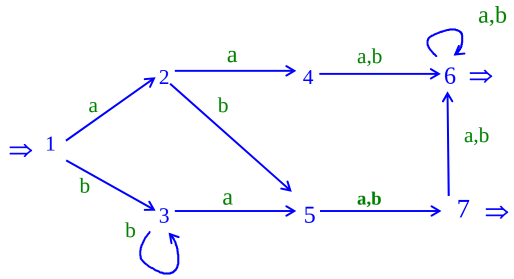
Temps de calcul

pour chaque E_{j+1} $O(n.\text{card } A)$

moins de n étapes

$\Rightarrow O(n^2.\text{card } A)$

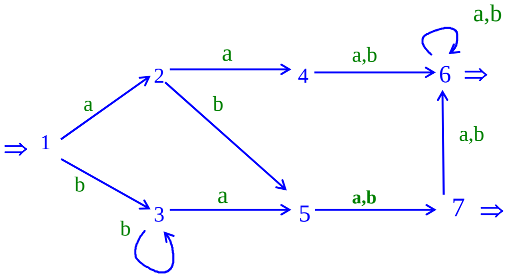
UMLV ©



	$\delta(.,a)$	$\delta(.,b)$	$F=E_0$
1	2	3	0
2	4	5	0
3	5	3	0
4	6	6	0
5	7	7	0
6	6	6	1
7	6	6	1

Calcul de E_1

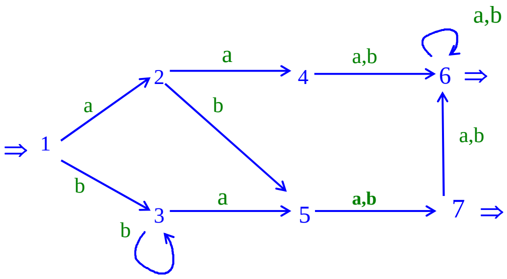
	E_0	$E_0(\delta(.,a))$	$E_0(\delta(.,b))$		E_1
1	0	0	0	1	0
2	0	0	0	2	0
3	0	0	0	3	0
4	0	1	1	4	1
5	0	1	1	5	1
6	1	1	1	6	2
7	1	1	1	7	2



	E_0	E_1
1	0	0
2	0	0
3	0	0
4	0	1
5	0	1
6	1	2
7	1	2

Calcul de E_2

	E_1	$E_1(\delta(.,a))$	$E_1(\delta(.,b))$		E_2
1	0	0	0	1	0
2	0	1	1	2	1
3	0	1	0	3	2
4	1	2	2	4	3
5	1	2	2	5	3
6	2	2	2	6	4
7	2	2	2	7	4



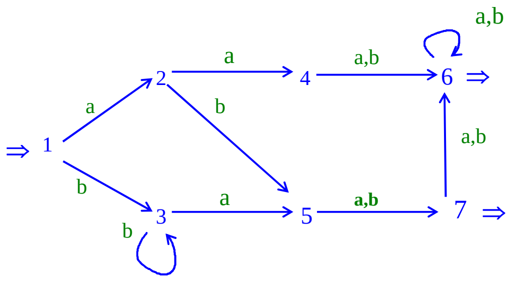
	E_0	E_1	E_2
1	0	0	0
2	0	0	1
3	0	0	2
4	0	1	3
5	0	1	3
6	1	2	4
7	1	2	4

Calcul de E_3

	E_2	$E_2(\delta(.,a))$	$E_2(\delta(.,b))$
1	0	1	2
2	1	3	3
3	2	3	2
4	3	4	4
5	3	4	4
6	4	4	4
7	4	4	4

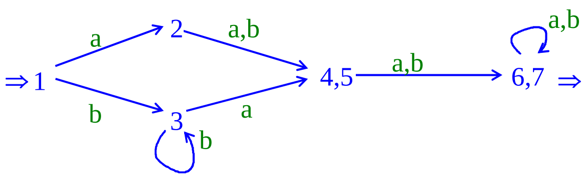
autant de classes qu'avec E_2

$\Rightarrow E_2 = E_3 = E$ congruence syntaxique



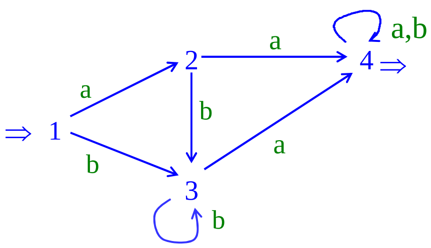
	E
1	0
2	1
3	2
4	3
5	3
6	4
7	4

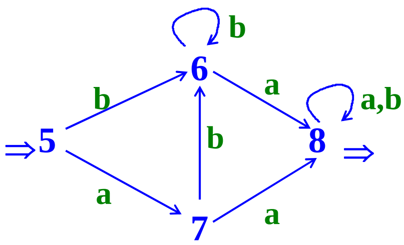
automate minimal : état = classe d'équivalence



Égalité d ’automates

Tester l'égalité des structures





en construisant la bijection

1 ⟶ 5

2 ⟶ 7

3 ⟶ 6

4 ⟶ 8

Calcul de la bijection

$a_1 = (Q_1, A, i_1, T_1, \delta_1)$ $a_2 = (Q_2, A, i_2, T_2, \delta_2)$

Calcul de la bijection f entre Q_1 et Q_2 par la procédure bij

bij(p, q) met en bijection p et q

début
  bij(i_1, i_2) ;
  si les états sont en bijection par f **et**
    pour tout q_1, $q_1 \in T_1$ ssi $f(q_1) \in T_2$ **alors**
    « a_1 et a_2 sont égaux » ;
fin

Calcul de la bijection

$a_1 = (Q_1, A, i_1, T_1, \delta_1)$ $a_2 = (Q_2, A, i_2, T_2, \delta_2)$

Test de bijection par calcul simultané de la fonction inverse : $g = f^{-1}$

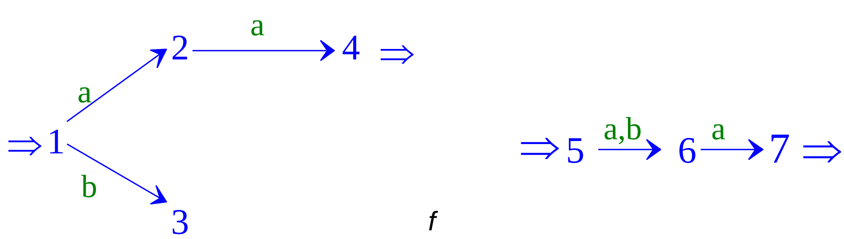
procédure bij (q_1, q_2) ;
début
  si (un seul de $f(q_1)$ et $g(q_2)$ défini)
  ou ($f(q_1)$ et $g(q_2)$ définis **et** $f(q_1) \neq q_2$) **alors**
    « a_1 et a_2 non égaux » ;
  sinon si $f(q_1)$ et $g(q_2)$ non définis **alors {**
    $f(q_1) \leftarrow q_2$; $g(q_2) \leftarrow q_1$;
    pour chaque $a \in A$ **faire**
      bij($\delta_1(q_1, a)$, $\delta_2(q_2, a)$) ;
  }
fin

UMLV ©

Temps d 'exécution : $O(\mathrm{card}A \,.\, \min\{\mathrm{card}Q_1, \mathrm{card}Q_2\})$

s'étend à des automates non complets

l 'algorithme précédent ne peut se passer de *g* !



f

1 → 5
2 → 6
3 → 6
4 → 7

Équivalence d 'automates

UMLV ©

Tester $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ $\qquad$ $n = \mathrm{card}\ Q_1 + \mathrm{card}\ Q_2$

- si $\mathcal{A}_1$ et $\mathcal{A}_2$ minimaux
  $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ ssi $\mathcal{A}_1 = \mathcal{A}_2$
- par minimisation
  -- minimiser $\mathcal{A}_1$ et $\mathcal{A}_2$
  -- puis tester $\mathcal{A}_1 = \mathcal{A}_2$

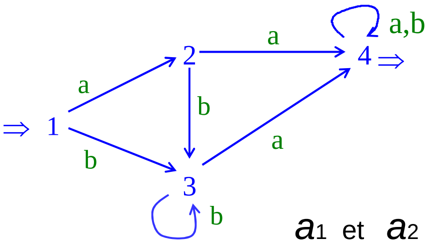
Temps (avec algorithme de Hopcroft) : $O(\mathrm{card}A \,.\, n \,.\, \log n)$

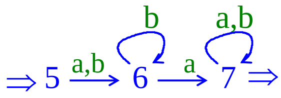
- test direct
  utilise la représentation d'ensembles
  avec opérations UNION / CLASSE

Temps : $O(\mathrm{card}A \,.\, n \,.\, \alpha(n))$ $\qquad$ [α = “logstar”]

Exemple

UMLV ©





$\mathcal{A}_1$ et $\mathcal{A}_2$ équivalents $\leftrightarrow$ i_1 et i_2 équivalents
(dans union disjointe)

Construction de l'équivalence :

1 | 2 | 3 | 4 | 5 | 6 | 7
1 5 | 2 | 3 | 4 | 6 | 7
1 5 | 2 6 | 3 | 4 | 7
1 5 | 2 3 6 | 4 | 7
1 5 | 2 3 6 | 4 7

$\mathcal{A}_1$ et $\mathcal{A}_2$ équivalents

Algorithme

UMLV ©

procédure Equiv(q_1, q_2) ;
début
  si un seul de q_1 et q_2 terminal **alors**
    « $\mathcal{A}_1$ et $\mathcal{A}_2$ non équivalents »
  sinon {
    $C_1 \leftarrow$ classe(q_1) ; $C_2 \leftarrow$ classe(q_2) ;
    si $C_1 \neq C_2$ **alors {**
      remplacer C_1 et C_2 par $C_1 \cup C_2$; /* union disjointe */
      pour chaque $a \in A$ **faire**
        Equiv($\delta_1(q_1, a)$, $\delta_2(q_2, a)$) ;
    }
  }
Fin

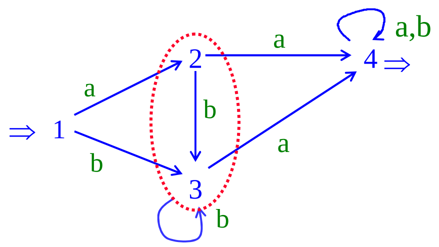
Temps $O(\text{card}\,A \cdot n \cdot \alpha(n))$ ($n = \text{card}\,Q_1 + \text{card}\,Q_2$)
avec représentation des classes par type « UNION / CLASSE »

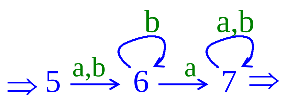
Minimisation par équivalence

UMLV ©

Note

Si l'un des automates est minimal, Equiv calcule la congrence syntaxique de l'autre.





a_2 est minimal

Équivalence finale : 1 5 | 2 3 6 | 4 7

Congruence de a_1 : 1 | 2 3 | 4

CLASSE / UNION

UMLV ©
UMLV ©

Gestion d'une partition de {1, 2, …, *n*} avec les **opérations principales**

CLASSE : numéro de classe d'un élément

UNION : union de classes disjointes

Implémentations possibles

1. table simple
2. arbres
3. idem + compression de chemins

Exemple n = 7

soit 1 ≡ 2 ; {1, 2} {3} {4} {5} {6} {7}

soit 5 ≡ 6 ; {1, 2} {3} {4} {5, 6} {7}

soit 3 ≡ 4 ; {1, 2} {3, 4} {5, 6} {7}

soit 1 ≡ 4 ; {1, 2, 3, 4} {5, 6} {7}

est-ce que 2 ≡ 3 ? oui car 2 et 3 dans la même classe

688

Par table simple

UMLV ©

	1	2	3	4	5	6	7
CLASSE	1	1	3	3	5	5	7

représente {1,2} {3,4} {5,6} {7}

UNION des classes (disjointes) de *p* et *q*

```
{     x ← CLASSE [p] ;  y ← CLASSE [q] ;
      pour k ← 1 à n faire
            si CLASSE [k] = y alors
                  CLASSE [k] ← x ;
}
```

Temps
CLASSE : constant
UNION : O(*n*)

	1	2	3	4	5	6	7
CLASSE	1	1	1	1	5	5	7

représente {1,2,3,4} {5,6} {7}

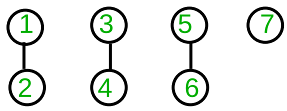
689

Par arbres

UMLV ©

partition	{1,2}	{3,4}	{5,6}	{7}
CLASSE,TAILLE	1,2	3,2	5,2	7,1

arbres



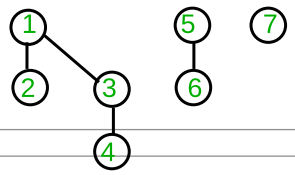
	1	2	3	4	5	6	7
P	-	1	-	3	-	5	-

```
CLASSE(i ){
      k ←  i ;
      tant que  P[k] défini faire k ← P[k] ;
      retour ( CLASSE[k] ) ;
}
```

Temps
CLASSE : O(*n*)
UNION : constant

partition	{1,2,3,4}	{5,6}	{7}
CLASSE,TAILLE	1,4	5,2	7,1

arbres



	1	2	3	4	5	6	7
P	-	1	1	3	-	5	-

690

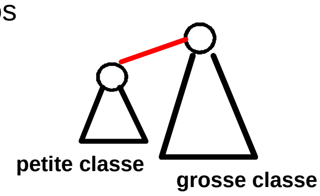
UMLV ©

Union des arbres

Éviter des arbres filiformes pour réduire
le temps de calcul de **`CLASSE(`*i* `)`**

Stratégie pour `UNION` : toujours mettre
le petit arbre enfant de la racine du gros

Temps
`CLASSE` : O(log*n*)
`UNION` : constant

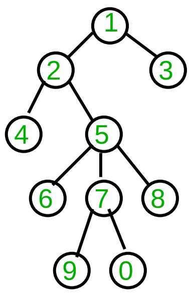


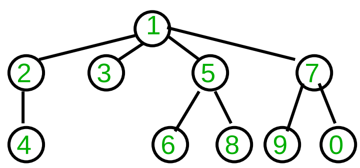
Preuve
niveau(*i*) augmente de 1 quand
union de *P* et *Q*, card *P* ≤ card *Q* et *i* ∈ *P*
i.e., quand la taille de la classe de *i* double au moins
Ceci ne peut arriver que $\lfloor \log_2 n \rfloor$ fois au plus

UMLV ©

Compression de chemins

Idée : réduire le temps de calcul de **`CLASSE(`*i* `)`**
en « aplatissant » l'arbre à chaque calcul





après calcul
de **`CLASSE(7)`**

Temps de *n* calculs de **`CLASSE`** : O(*n* α (*n*))

où α(*n*) est le plus petit entier *k* tel que $n \leq \left. 2^{2^{\cdot^{\cdot^{2}}}} \right\} k$ fois

Preuve [Aho, Hopcroft, Ullman, 1974]

Ackermann

UMLV ©

fonction A : **N** x **N** → **N** définie par

$A(0, y) = 1$ $\quad y \geq 0$

$A(1, 0) = 2$

$A(x, 0) = x + 2$ $\quad x \geq 2$

$A(x, y) = A(A(x-1, y), y-1)$ $\quad x, y \geq 1$

Propriétés

$y = 0$ $\quad A(x, 0) = x+2$ $\quad x \geq 2$

$y = 1$ $\quad A(x, 1) = 2.x$ $\quad x \geq 1$

car $\quad A(1,1) = A(A(0,1),0) = A(1,0) = 2$

$A(x,1) = A(A(x-1,1),0) = A(x-1,1)+2, \ldots$

$y = 2$ $\quad A(x, 2) = 2^x$ $\quad x \geq 1$

car $\quad A(1,2) = A(A(0,2),1) = A(1,1) = 2$

$A(x,2) = A(A(x-1,2),1) = 2.A(x-1,2)\ , \ldots$

UMLV ©

$y = 3$ $\quad A(x,3) = 2^{2^{\cdot^{\cdot^{2}}}}$ « tour de 2 en x exemplaires »

$\alpha(n)$ = plus petit k tel que $n \leq A(k, 3)$

$A(4,4)$ = « tour de 2 en 65536 exemplaires »

si n raisonnable, $\alpha(n) \leq 4$