

Algorithmique IN102-06

Michel Mauny

ENSTA

Prénom.Nom@ensta.fr

Cours 6

1 Recherche de motifs

- Le problème
- Algorithme de Rabin-Karp
- Automates

2 Perspectives

- Modèles de calculs
- Problèmes
- Classes de complexité

Recherche d'un motif M dans un texte T

```
texte T[1, ..., n] = // Portail, Théophile Gautier, 1938
```

```
Ne trouve pas étrange, homme du monde, artiste,  
Qui que tu sois, de voir par un portail si triste  
S'ouvrir fatalement ce volume nouveau.
```

```
Hélas ! Tout monument qui dresse au ciel son faite,  
Enfonce autant les pieds qu'il élève la tête.  
Avant de s'élancer tout clocher est caveau :
```

```
En bas, l'oiseau de nuit, l'ombre humide des tombes ;  
En haut, l'or du soleil, la neige des colombes,  
Des cloches et des chants sur chaque soliveau ;
```

```
En haut, les minarets et les rosaces frêles,  
Où les petits oiseaux s'enchevêtrent les ailes,  
Les anges accoudés portant des écussons ;
```

Recherche d'un motif M dans un texte T

```
motif M[1, ..., m] = " des "  
texte T[1, ..., n] =
```

```
Ne trouve pas étrange, homme du monde, artiste,  
Qui que tu sois, de voir par un portail si triste  
S'ouvrir fatalement ce volume nouveau.
```

```
Hélas ! Tout monument qui dresse au ciel son faite,  
Enfonce autant les pieds qu'il élève la tête.  
Avant de s'élancer tout clocher est caveau :
```

```
En bas, l'oiseau de nuit, l'ombre humide des tombes ;  
En haut, l'or du soleil, la neige des colombes,  
Des cloches et des chants sur chaque soliveau ;
```

```
En haut, les minarets et les rosaces frêles,  
Où les petits oiseaux s'enchevêtrent les ailes,  
Les anges accoudés portant des écussons ;
```

Recherche d'un motif M dans un texte T

```
texte T[1, ..., n]  
motif M[1, ..., m]
```

Problème : Trouver tous les décalages s de $[0, n-m]$ tels que $\forall j \in [1, m], M[j]=T[s+j]$

Applications :

- traitement de texte
- recherche de séquences de gènes
- ...

Algorithme naïf

Chercher les s , t.q. $T[s]$ commence une occ. de M dans T

Pour s variant de 0 à $n-m$:
si **Décalage-valide**(M, T, s), alors
classer s comme un décalage valide

Décalage-valide(M, T, s)

si $\forall j \in [1, m], M[j]=T[s+j]$, alors vrai
sinon faux

Complexité dans le cas le pire : $\Theta((n-m+1) \times m)$

Algorithme de Rabin-Karp

Idée : codage des mots en base d

- $p = M[1] \times d^{m-1} + M[2] \times d^{m-2} + \dots + M[m-1] \times d + M[m]$

Pour tout décalage s de $[0, n-m]$:

- $t_s = T[s+1] \times d^{m-1} + T[s+2] \times d^{m-2} + \dots + T[s+m]$

t_{s+1} peut se calculer facilement à partir de t_s :

$$t_{s+1} = (t_s - T[s+1] \times d^{m-1}) \times d + T[s+m+1]$$

Algorithme de Rabin-Karp

Exemple : alphabet $\{A, T, G, C\}$

$A = 0, T = 1, G = 2, C = 3, d = 4$

Pour $T = \text{«ACACGTT»}$:

$$\begin{aligned} t_0 &= 0 \times d^2 + 3 \times d + 0 = 12 \\ t_1 &= 3 \times d^2 + 0 \times d + 3 = 51 \\ t_2 &= 0 \times d^2 + 3 \times d + 2 = 14 \\ t_3 &= 3 \times d^2 + 2 \times d + 1 = 57 \\ t_4 &= 2 \times d^2 + 1 \times d + 1 = 37 \end{aligned}$$

Algorithme de Rabin-Karp

Rabin-Karp (M, T, d)

```

calculer  $h = d^{m-1}$ 
calculer  $p$ 
calculer  $t = t_0$ 
pour  $s$  variant de 0 à  $n-m-1$ :
  si  $p == t$  alors classer  $s$  comme valide
   $t \leftarrow (t - T[s+1] \times h) \times d + T[s+m+1]$ 
  si  $p == t$  alors classer  $s$  comme valide
    
```

Algorithme de Rabin-Karp

Complexité :

- calcul de h en $\Theta(\log(m))$
- calculs de p et t_0 en $\Theta(m)$
- $(n - m)$ calculs des t_s : $(n - m) \times \Theta(1)$

Complexité globale dans le cas le pire : $\Theta(n - m)$

Optimal !

Algorithme de Rabin-Karp

Problème : si d et/ou m grand, les nombres manipulés sont gigantesques!!!

$$26^{10} = 141167095653376$$

$$128^{20} \approx 10^{42}$$

Idee : effectuer les calculs modulo un entier «ni trop grand, ni trop petit»

$$128^{20} = 58986 \pmod{65521}$$

Algorithme de Rabin-Karp

Rabin-Karp-efficace (M, T, d)

```

calculer  $h = d^{m-1} \pmod{q}$ 
calculer  $p \pmod{q}$ 
calculer  $t = t_0 \pmod{q}$ 
pour  $s$  variant de 0 à  $n-m-1$ :
  si  $p == t$  alors
    si Décalage-valide( $M, T, s$ ) alors
      classer  $s$  comme valide
     $t \leftarrow ((t - T[s+1] \times h) \times d + T[s+m+1]) \pmod{q}$ 
  si  $p == t$  alors
    si Décalage-valide( $M, T, s$ ) alors
      classer  $s$  comme valide
    
```

Algorithme de Rabin-Karp

Complexité globale dans le cas le pire : $\Theta((n - m + 1) \times m)$

Complexité globale s'il y a peu de décalages valides (ce qui est très probable) : $\Theta(n + m)$

Automates à états finis

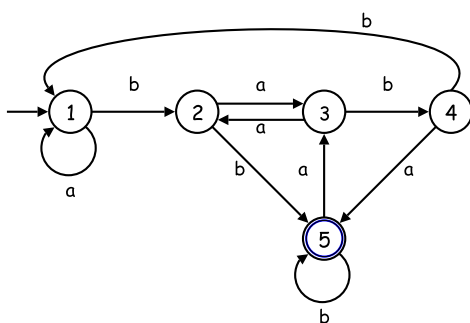
Reconnaissance mécanique de «mots»

Un automate fini déterministe A sur un alphabet fini Σ est la donnée de :

- Q ensemble fini d'états
- q_0 un état initial
- F ensemble d'états finaux
- $\delta : Q \times \Sigma \rightarrow Q$ fonction de transition

$$(Q \times \Sigma \rightarrow 2^Q?)$$

Automates finis



Exemple de mot reconnu : **abaababbbb**

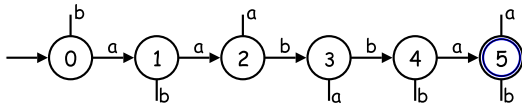
Recherche de motif

Pour tout motif M de longueur m , on peut construire un automate A qui le reconnait :

- A est doté de $m + 1$ états $\{0, 1, \dots, m\}$
- 0 est l'état initial
- m est l'unique état final

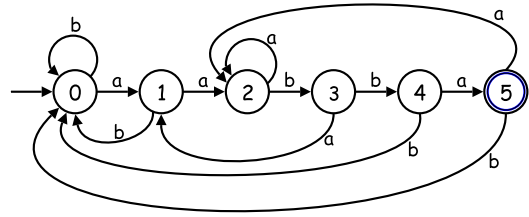
Exemple : reconnaissance de "aabba"

Recherche de motif



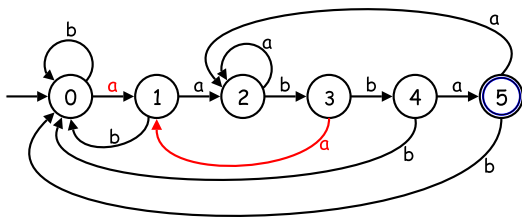
Exemple : reconnaissance de "aabba"

Recherche de motif



Exemple : reconnaissance de "aabba"

Construction de l'automate



En (3, a), on a reconnu aaba et on pourrait commencer à reconnaître aabba M="aabba"

Modèles de calcul

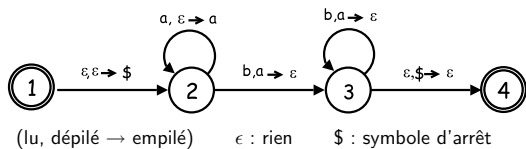
Automates finis :

- machine à café, ascenseur, additionneur, analyseur lexical, recherche de séquences ADN, ...
- puissance de calcul limitée
- ne peuvent reconnaître $a^n b^n$

Modèles de calcul

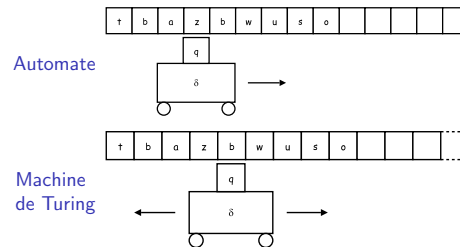
Automates à pile :

- comme automates finis, mais avec une pile
- plus puissants
- on peut reconnaître $a^n b^n$:



- mais ne peut reconnaître $a^n b^n c^n$

Modèles de calcul



Machines de Turing (ou autre modèle équivalent) : peut encoder toute fonction calculable

Classification de problèmes

Problème abstrait relation entre un ensemble d'instances (du problème) et un ensemble de solutions

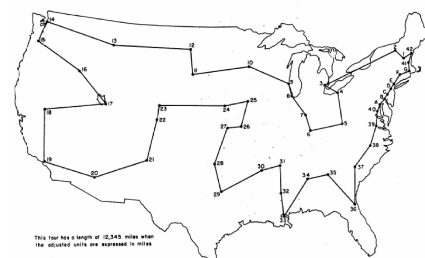
Ex : recherche du plus court chemin entre deux sommets d'un graphe

Problème de décision problème dont l'ensemble de solution possibles est simplement *vrai* ou *faux*

Ex : existence d'un chemin de longueur k

Problème de validation problème de décision visant à valider un certificat, candidat à devenir solution

Problème abstrait



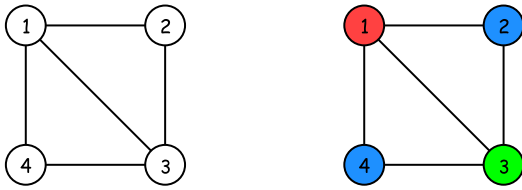
This tour has a length of 10,543 miles when the shortest route are expressed in miles.

Fig. 10. The optimal tour of 49 cities.

Le problème dit «du voyageur de commerce» : visiter n villes en parcourant une distance $\leq k$

Problème de décision

Coloriable avec 3 couleurs ?



Pour 60 sommets : $\approx 10^{30}$ coloriages possibles

Problème de décision

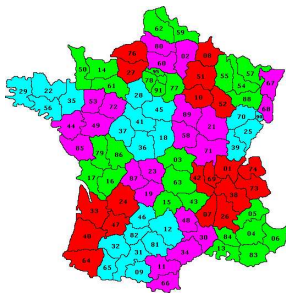
Coloriable avec 3 couleurs ?



Et avec 4 ?

Problème de validation

Théorème «des 4 couleurs»



Classes de complexité

Classe de complexité P : ensemble des problèmes de décision solubles en temps polynomial

Classe de complexité NP : ensemble des problèmes de validation solubles en temps polynomial

Clairement, $P \subseteq NP$, mais ...

$$P \stackrel{??}{=} NP$$

Note : il existe des classes de problèmes encore plus larges (EXPTIME, par exemple, voire pire).

Problèmes NP -complets

- Satisfiabilité d'une formule logique
- Sac à dos
- Voyageur de commerce
- «3-coloriabilité»
- ...

Problème NP -complet

Problème NP au moins aussi difficile que tout autre problème NP

- facile de vérifier une solution proposée
- difficile de trouver une solution, ou de savoir s'il en existe une
- si dans P , alors $P = NP$

Problèmes indécidables

Le problème de l'arrêt est indécidable

Preuve par l'absurde : supposons qu'il existe une fonction **termine** capable de tester si un programme donné en argument s'arrête

Terminer ou ne pas terminer...

```
absurde() {  
  tant que termine(absurde), faire  
  rien  
}
```

Problème indécidable

Pourquoi le problème de l'arrêt est-il si difficile ?

Soit une fonction **successeur** qui parcourt tous les entiers dans $[1, +\infty[\times [3, +\infty[$

```
fermat(x, y, z, n) {  
  tant que  $x^n + y^n \neq z^n$ , faire  
  (x, y, z, n) ← successeur(x, y, z, n)  
}
```

Problèmes difficiles

Problème de la factorisation : étant donné un entier n , trouver sa factorisation en produit de nombres premiers

- mathématiquement «facile»
- pas d'algorithme polynomial de factorisation
- de grande importance en cryptographie

Meilleures méthodes connues :

- crible quadratique : $\mathcal{O}(\exp((\ln(n))^{1/2}(\ln(\ln(n))))^{1/2}))$
- crible algébrique : $\mathcal{O}(\exp(1,923 \times (\ln(n))^{1/3}(\ln(\ln(n))))^{1/3}))$

http ://www.rsasecurity.com/

RSA-640

Prize: \$20,000

Status: Factored

Decimal Digits: 193

31074182404900437213507500358885679300373460
22842727545720161948823206440518081504556346
82967172328678243791627283803341547107310850
19195485290073377248227835257423864540146917
36602477652346609

http ://www.rsasecurity.com/

RSA-704

Prize: \$30,000

Status: Not Factored

Decimal Digits: 212

74037563479561712828046796097429573142593188
88923128908493623263897276503402826627689199
64196251178439958943305021275853701189680982
86733173273108930900552505116877063299072396
380786710086096962537934650563796359

Contexte

Logiciel omni-présent

- Aspect langage
 - haut niveau et efficacité
 - interopérabilité (échange de données sur le réseau, utilisation de bibliothèques d'autres langages)
 - qu'est-ce qu'un programme (ou un système) correct ?
- Aspect algorithmique
 - problème décidable (faisabilité)
 - complexité (faisabilité, passage à l'échelle, sécurité)
 - efficacité