UE 4I900
Master Informatique, Sorbonne Université

COMPLEX

COMPLEXité, algorithmes randomisés et approchés

Enseignants :
- cours et TDs : D. Vergnaud, F. Pascual (responsables),
- TDs : T. Bellitto, S. Bouaziz-Hermann, B. Escoffier.

Organisation

- Un cours (2h) par semaine.
- 4h TD par semaine les 2 premières semaines,
  2h TD et 2h TME les semaines 3 à 5 puis 7 à 10.

- Un projet sur la première partie de l'UE. Séances de TME des semaines 3 à 5 sur le projet. Soutenances en salle de TME semaine 6.

- Ressources sur moodle.

Contenu de l'UE

Semaines 1-5 :

* Introduction à la théorie du calcul et à la complexité de problèmes.
* Méthodes arborescentes.
* Algorithmes d'approximation.

Semaines 6-10 :

* Algorithmes randomisés (Las Vegas, Monte Carlo)
* Classes de complexité probabilistes
* Test de primalité ; MAX3SAT ; Tests d'identité polynomiale.

Contrôle des connaissances

- Projet : 20 %
- Examen réparti 1 : 40 %
- Examen réparti 2 : 40 %

- Le projet est pris en compte en seconde session (note : 20 % projet, 80 % examen).

Chapitre 1
Introduction à la théorie du calcul

a. Y a-t-il des **problèmes** que l'on ne **peut pas** résoudre avec un ordinateur, i.e. pour lesquels il n'existe pas d'**algorithme** ?

Problème ?

Algorithme ?

Introduction à la théorie du calcul

b. Parmi les problèmes que l'on peut résoudre, y en a-t-il de plus difficiles que d'autres ?

GPS

Trajet optimal en quelques secondes

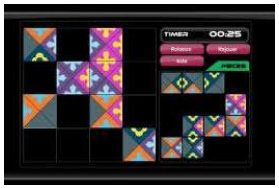
Eternity II

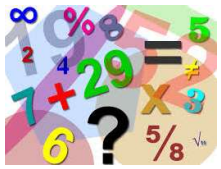
Puzzle 16*16

2 000 000 $

Introduction à la théorie du calcul

I. Problème, algorithme





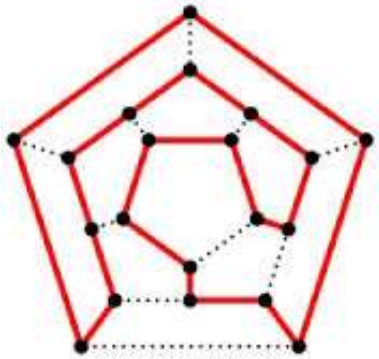
- Une donnée/ une instance
- Une question

Une séquence d'instructions/ un algorithme

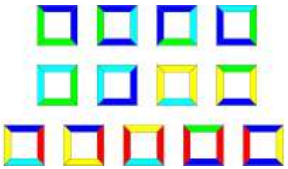
Une réponse/ une solution

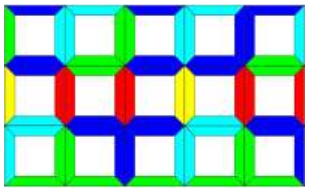
Exemple 1 : chaîne entre deux sommets

Exemple 2 : chaîne hamiltonienne



Exemple 3 : pavage du plan





...

...

I. Problème, algorithme

Muhammad ibn Mūsā Al-Khwârizmî
(environ 780-850, Ouzbékistan)

Algorithme : Suite finie et non ambiguë d'instructions simples.

But : donner la réponse à une question (résoudre un problème).

I. Problème, algorithme

David Hilbert
(1862-1943, Prusse/Allemagne)

Congrès international des mathématiciens, Paris, 1900.
23 problèmes ouverts.

Enoncé 10 : Trouver un algorithme déterminant si un polynôme à coefficients entiers a une racine entière.

II. Formalisation

Alan Turing
(1912-1954, Angleterre)

- 1936 : la machine de Turing
Article fondateur de la science informatique : *« On Computable Numbers, with an Application to the Entscheidungsproblem »*.
- 1936 : décodage du code secret allemand ENIGMA.
- 1950 : intelligence artificielle.

II. Formalisation

1. Problème et langage

II. Formalisation

1. Problème et langage
2. Machine de Turing

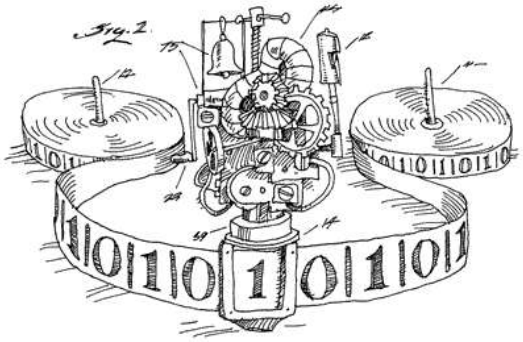
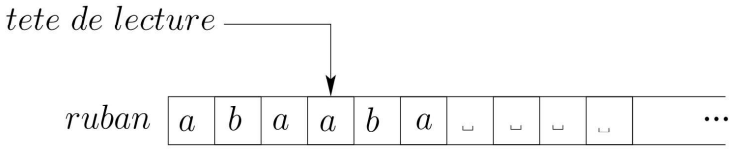
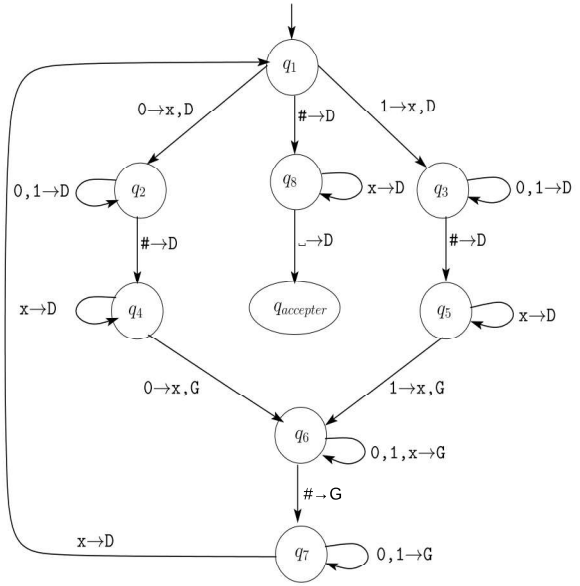


Schéma d'une machine de Turing :



Exemple 1 :
Problème/langage : {w#w : w mot sur {0,1}}

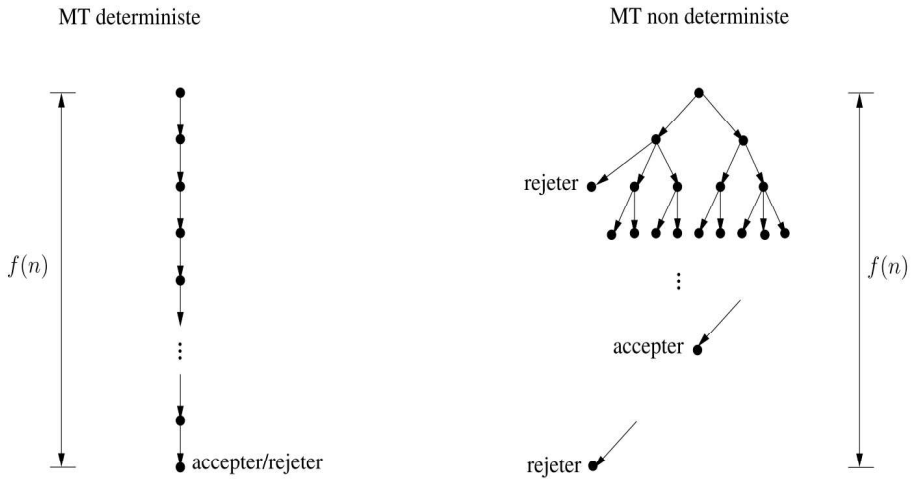
```
0 0 1 0 1 1 1 0 1 0 1 # 0 0 1 0 1 1 1 0 1 0 1 ␣
x 0 1 0 1 1 1 0 1 0 1 # 0 0 1 0 1 1 1 0 1 0 1 ␣
x 0 1 0 1 1 1 0 1 0 1 # x 0 1 0 1 1 1 0 1 0 1 ␣
x x 1 0 1 1 1 0 1 0 1 # x 0 1 0 1 1 1 0 1 0 1 ␣
                      ⋮
x x x x x x x x x x x # x x x x x x x x x x x ␣
```



Exemple 2 :
Problème/langage : éléments distincts

```
# 0 1 0 1 # 0 1 1 0 # 0 1 0 1 1 # 1 0 ␣
# 0 1 0 1 # 0 1 1 0 # 0 1 0 1 1 # 1 0 ␣
# 0 1 0 1 # 0 1 1 0 # 0 1 0 1 1 # 1 0 ␣
# 0 1 0 1 # 0 1 1 0 # 0 1 0 1 1 # 1 0 ␣
# 0 1 0 1 # 0 1 1 0 # 0 1 0 1 1 # 1 0 ␣
# 0 1 0 1 # 0 1 1 0 # 0 1 0 1 1 # 1 0 ␣
                    ⋮
```

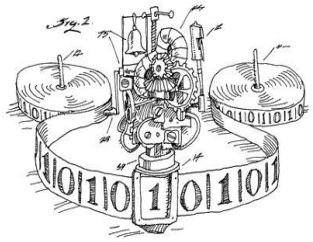
3. Machine de Turing non déterministe



Remarques pour conclure

- D'autres modèles ?

Thèse de Church (Turing)

-  =