COMPLEXITÉ ALGORITHMIQUE

Réalisé par: El amine El alami Noha
El hadrati Abir
Encadré par: Mr.El merouani Mohamed

PLAN:

- 1-Définition de l'algorithme
- 2-Définition de la complexité
- 3-Présentation de la machine de TURING
- 4-Principes de la machine de TURING
- 5-Les classes de la complexité
  - -Classe P
  - -Classe co-P
  - -Classe NP
- 6-Modèles de calcul
- 7-Coûts des opérations

INTRODUCTION:

- La recherche opérationnelle est une discipline dont le but est de fournir des méthodes pour répondre à un type précis de problème, c'est-à-dire à élaborer une démarche universelle pour un type de problème qui aboutit à la ou les solutions les plus efficaces.
- La particularité de la recherche opérationnelle est que les méthodes proposées sont des démarches rationnelles basées sur des concepts et outils mathématiques et/ou statistiques.

C 'EST QUOI UN ALGORITHME?

Un algorithme est une procédure finie et mécanique de résolution d'un problème.

Exemples : les algorithmes d'Euclide, l'algorithme de Dijkstra …

- Un algorithme doit se terminer sur toutes les données possibles du problème et doit fournir une solution correcte dans chaque cas.

- Pour résoudre informatiquement un problème donné, on implante donc un algorithme sur un ordinateur. Mais, pour un problème donné, il existe bien souvent plusieurs algorithmes.

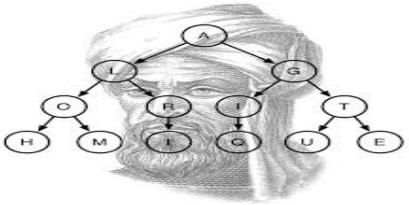
Y a-t-il un intérêt à choisir ? et si oui comment choisir ?

- En pratique, il n'est même pas suffisant de détenir un algorithme. Il existe des problèmes pour lesquels on a des algorithmes, mais qui restent comme «!informatiquement non résolus!».
- C'est parce que les temps d'exécution sont vite exorbitants.
- On cherche alors des heuristiques pour abaisser ces temps de calcul.

C'EST QUOI LA COMPLEXITÉ?

En informatique, le mot complexité recouvre en fait deux réalités :

- la complexité des algorithmes C'est l'étude de l'efficacité comparée des algorithmes.



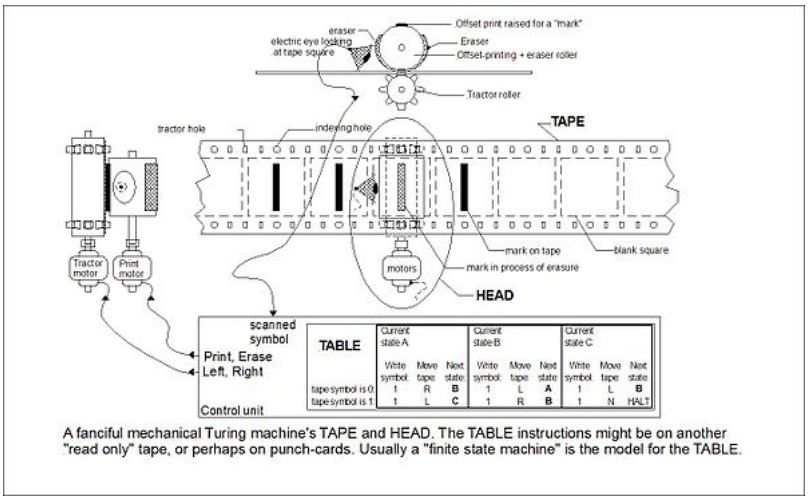
- On mesure ainsi le temps et aussi l'espace nécessaire à un algorithme pour résoudre un problème. Cela peut se faire de façon expérimentale ou formelle.

- La complexité des algorithmes a abouti à une classification des problèmes en fonction des performances des meilleurs algorithmes connus qui les résolvent.
- Techniquement, les ordinateurs progressent de jour en jour.

Elle a été conçue indépendamment des caractéristiques techniques des ordinateurs.

- Les domaines alliés de la complexité et de la preuve de programmes utilisent toutes les notions que nous avons vues jusqu'à présent dans ce cours.

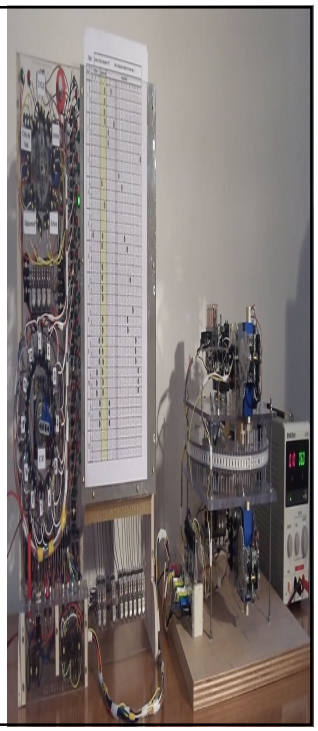
QU'EST CE QUE UNE MACHINE DE TURING



A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

QU'EST CE QUE UNE MACHINE DE TURING

- C'est un prototype de la machine imaginée par Alan Turing en 1936 pour modéliser le concept d'algorithme et donner une réponse négative au problème de la décision en logique.
- Ce prototype est directement issu de la description qu'il a donnée dans son article publié en 1937 intitulé : <*On computable Numbers*, with an Application to the Entscheidungs problem.>



- Cette machine est en fait le modèle le plus simple que l'on puisse concevoir et qui satisfait aux critères informels mais universels qui caractérisent un algorithme.
- C'est un modèle universel de calcul et elle peut calculer tout ce que n'importe quel ordinateur physique peut calculer.
- Inversement ce qu'elle ne peut pas calculer ne peut l'être non plus par un ordinateur.

Principe d'une machine de Turing

- une TM est une unité de contrôle munie d'une tête de lecture/écriture positionnée sur un ruban qui comporte un nombre infini dénombrable de cases
- l'unité de contrôle a un ensemble fini d'états possibles
- A chaque instant, la machine évolue en fonction de son état et de l'information lue sur la case courante : la tête écrit sur cette case en effaçant le contenu précédent, se déplace d'une case à gauche, à droite ou bien ne se déplace pas.

Classes de Complexité

Problème de décision

Un problème de décision est un problème pour lequel la réponse est dans l 'ensemble

{ **OUI - NON** }

La classe P

P

=

{L : il existe un programme M *en temps polynomial* sur une *machine de Turing Déterministe* pour lequel L_M=L}

Un problème est dans P s 'il existe un algorithme en temps polynomial pour le résoudre

La classe co-P

- La classe co-P est le complémentaire de P

- Pour un problème donné de P,

on montre facilement qu'il est dans co-P :

On échange les réponses OUI et NON

- Pour un problème de co-P,

on montre facilement qu 'il est dans P (idem)

P = co-P

La classe NP

NP
=

{L : il existe un programme M *en temps polynomial* sur une *machine de Turing Non Deterministe* pour lequel L_M=L}

Il existe un algorithme non déterministe en temps polynomial permettant de résoudre L.

Il existe un algorithme déterministe en temps polynomial permettant de vérifier une solution de L.

Modèles de calcul

- Les machines de Turing (TM) ou les Random Access Machines (RAM) sont des machines abstraites. Elles servent d'«!étalon!» à la mesure des complexités en temps et en espace des fonctions dites calculables.
- Thèse de Church*-Turing ** (indépendamment mais les 2 en 1936) Toute fonction effectivement calculable l'est par une machine de Turing.

* A.Church (1903-1995) : mathématicien et logicien américain
** A.Turing (1912-1954) : mathématicien britannique

- Notre optique ici n'est évidemment pas de présenter la théorie de la calculabilité mais juste d'en appréhender les principes les plus simples.
- Notre approche pragmatique sert à comparer des algorithmes résolvant un même problème afin d'estimer rigoureusement lesquels sont les meilleurs.

Repérer les opérations fondamentales

- C'est la nature du problème qui fait que certaines opérations deviennent plus fondamentales que d'autres dans un algorithme.
- Leur nombre intervient alors principalement dans l'étude de la complexité de l'algorithme.
-

Recherche d'un élément dans une liste, un tableau, un arbre …	**comparaisons**
Tri d'une liste, d'un tableau, d'un fichier…	comparaisons déplacements
Multiplication de polynômes, de matrices, de grands entiers	additions multiplications

Coût des opérations

- Dans un ordinateur (un vrai) comme dans une TM ou une RAM, toutes les opérations n'ont pas le même coût.

  Exemple : multiplication et addition. C'est peu étonnant.

  La multiplication est bien une généralisation de l'addition.
- Pourquoi coûterait-elle pareil ?
- Seules les opérations élémentaires sont à égalité de coût.

- Pour simplifier, on peut faire l'hypothèse que toutes les opérations ont un coût uniforme (même si cela manque souvent de réalisme). Ce coût est alors constant car il ne dépend plus de rien.
- Pour un programme donné, les profileurs permettent de savoir quelles sont les instructions qui prennent le plus de temps

ESTIMATION ASYMPTOTIQUE

- En complexité, on ne considère que des fonctions positives de IN dans IR. On ne veut pas évaluer précisément les temps d'exécution (d'autant que ça dépend des machines…). On se contente de trouver des approximations.
- On dit que f positive est asymptotiquement majorée (ou dominée) par g et on écrit f = O (g) quand il existe une constante c, c>0, telle que pour un n assez grand, on a : $f(n) \leq c \cdot g(n)$
- On définit symétriquement la minoration de f par g : f = W (g).
- On dit que f est du même ordre de grandeur que g et on écrit f = q

Merci pour votre attention.