Problèmes

Après avoir discuté des

traductions machines

d'un algorithme

Après avoir étudié des

algorithmes particuliers

Étudions

tous les algorithmes

solutions d'un même problème

Problèmes

Problèmes :

- **définition des entrées**
- **lien entre les entrées et les sorties (spécification)**

Problèmes

Problèmes :

- **définition des entrées**
- **définition de la taille des entrées**
- **lien entre les entrées et les sorties (spécification)**

pour pouvoir parler de complexité

Problèmes

Exemple : problème de la médiane

- **entrées : un ensemble S d'entiers**
- **taille : max(S)**
- **sortie :**

$$m / \begin{cases} m \in S \\ card\{n \in S, n < m\} = card\{n \in S, m < n\} \end{cases}$$

Problèmes

Exemple : problème de la médiane

- **entrées : un ensemble S d'entiers**
- **taille : somme(S)**
- **sortie :**

$$m / \begin{cases} m \in S \\ card\{n \in S, n < m\} = card\{n \in S, m < n\} \end{cases}$$

Problèmes

Exemple : problème de la médiane

- **entrées : un ensemble S d'entiers**
- **taille : card(S)**
- **sortie :**

$$m / \begin{cases} m \in S \\ card\{n \in S, n < m\} = card\{n \in S, m < n\} \end{cases}$$

Problèmes

Cout d'un problème :

coût de l'algorithme le plus rapide qui résoud le problème

Problèmes

Exemple : problème de la médiane

- **entrées : un ensemble S d'entiers**
- **taille : t=card(S)**
- **sortie :**

$$m / \begin{cases} m \in S \\ card\{n \in S, n < m\} = card\{n \in S, m < n\} \end{cases}$$

coût : O(t*t) en testant chaque m de S

Problèmes

Exemple : problème de la médiane

- **entrées : un ensemble S d'entiers**
- **taille : t=card(S)**
- **sortie :**

$$m / \begin{cases} m \in S \\ card\{n \in S, n < m\} = card\{n \in S, m < n\} \end{cases}$$

coût : O(t*ln(t)) en triant les élts de S

Problèmes

Problème de décision :

- **entrées**
- **taille des entrées**
- **sortie : un booléen**

souvent la réponse à

existe-t-il des sorties telles que ...

Problèmes

Problème d'optimisation :

- **entrées**
- **taille des entrées**
- **sortie : des valeurs**

rendant minimum une expression

souvent la réponse à

quelles sont les meilleures sorties telles que ...

Problèmes

Définition :

Un problème est dit

polynômial

s'il existe un algorithme qui le résoud avec une complexité polynômiale en la taille des entrées

Problèmes

Définition :

La classe des problèmes polynômiaux s'appelle

la classe P

Problèmes

Définition :

Deux problèmes sont dits de la même classe si on peut passer polynômialement d'une solution de l'un à une solution de l'autre

Remarque :

définition cohérente avec celle de P

Problèmes

Théorème :

À tout problème d'optimisation on peut associer un problème de décision de même classe

Problèmes

Exemple :

P1 :
- **in : un ensemble S d'entiers**
- **taille : card(S)**
- **out :**

$$A \subset S \,/\, \begin{cases} produit\{n \in A\} > somme\{n \in S, n \notin A\} \\ cardA \text{ minimum} \end{cases}$$

P2 :
- **in : un ensemble S d'entiers et k entier**
- **taille : card(S)**
- **out :**

$$\exists ?\, A \subset S \,/\, \begin{cases} produit\{n \in A\} > somme\{n \in S, n \notin A\} \\ cardA < k \end{cases}$$

Problèmes

Problème de décision :

- **entrées**
- **taille des entrées**
- **sortie :**

$$\exists ?\ \text{out}\ /\ P(\text{out})$$

Problèmes

Un problème de décision est dit de la classe NP si le calcul de *P* peut se faire en temps polynômial

$$\exists ? \text{ out } / P(\text{out})$$

!!! NP signifie

Non deterministic Polynomial

Problèmes

Un problème de décision est dit de la classe NP si le calcul de *P* peut se faire en temps polynômial

Il existe deux autres définitions classiques :

- **par un oracle**
- **par les certificats**

Problèmes

Exemples

- **trouver la médiane est dans P**
- **le sac à dos est dans NP**
- **le bin-packing est dans NP**
- **un entier est-il premier ? est dans P**
- **un graphe est-il connexe ? dans P**

Problèmes

Exemple :

l'entier n

possède-t-il un diviseur propre ?

Problèmes

Challenge :

démontrer ou infirmer

P=NP

On sait déjà que

$$P \subseteq NP$$

Problèmes

Certains problèmes de NP sont dans une même classe :

NP-complets

Problèmes

Certains problèmes de NP sont dans une même classe :

NP-complets

On demontre que tous les problèmes de NP ne sont pas plus difficiles que les NP-complets

Problèmes

Certains problèmes de NP sont dans une même classe :

NP-complets

On demontre que tous les problèmes de NP ne sont pas plus difficles que les NP-complets

donc

si un des NP-complets est dans P

Problèmes

Certains problèmes de NP sont dans une même classe :

NP-complets

On demontre que tous les problèmes de NP ne sont pas plus difficles que les NP-complets

donc

si un des NP-complets est dans P

alors P=NP

Problèmes

Exemples

- **le sac à dos** **est NP-complet**
- **le bin-packing** **est NP-complet**
- **la satisfiabilité** **est NP-complet**
- **le voyageur de commerce** **est NP-complet**
- **l'emploi du temps** **est NP-complet**

CALCULABILITÉ

Après avoir discuté des

traductions machines

d'un algorithme

Après avoir étudié des

algorithmes particuliers

Après avoir étudié

tous les algorithmes

solutions d'un même problème

Étudions l'existence d'un algorithme

solution d'un problème donné

CALCULABILITÉ

Théorème :

Il existe des problèmes pour lesquels

on ne peut pas

trouver d'algorithme solution

CALCULABILITÉ

Exemple :

- **le problème de la terminaison**
- **l'équivalence de programmes**

CALCULABILITÉ

Problème de la terminaison :
supposons que

```
termton(pgm,vals:string):string
```

renvoie “oui” si le programme `pgm`
termine sur l’entrée `vals`**,**
renvoie “non” sinon

CALCULABILITÉ

Problème de la terminaison :

supposons que

`termton(pgm,vals:string):string`

renvoie “oui” si le programme `pgm`
termine sur l’entrée `vals`**,**

renvoie “non” sinon

Et que `termton` termine toujours

CALCULABILITÉ

Problème de la terminaison :

posons

```
function soimeme(pgm:string):string;
begin
  soimeme:=termton(pgm,pgm)
end;
```

CALCULABILITÉ

Problème de la terminaison :

posons

```
function soimeme(pgm:string):string;
begin
  soimeme:=termton(pgm,pgm)
end;
```

Exercice : si `Id` est la définition de l'identité, évaluer

```
termton(Id,"coucou")
termton(Id,soimeme)
termton(soimeme,Id)
termton(Id,soimeme)
termton(soimeme,soimeme)
```

CALCULABILITÉ

Problème de la terminaison :

posons

```
function boucleton(pgm:string):string;
begin
   tant que soimeme(pgm)="oui" faire
      (* rien du tout *)
   fin tant que
   boucleton:="oui"
end;
```

CALCULABILITÉ

```
function boucleton(pgm:string):string;
begin
   tant que soimeme(pgm)="oui" faire
      (* rien du tout *)
   fin tant que
   boucleton:="oui"
end;
```

Exercice : si `Id` est la définition de l'identité, évaluer

```
boucleton(Id)
boucleton(soimeme)
termton(Id,boucleton)
termton(boucleton,Id)
```

CALCULABILITÉ

Problème de la terminaison :

que vaut

```
boucleton(boucleton)
```

??????

On parle d'

"argument diagonal"