Programmation Efficace

Rappels algorithmiques

Université Paris Cité
Année universitaire 2023-2024

Qu'est-ce-qu'un algorithme ?

Suite d'instructions pour résoudre un problème ou effectuer un calcul

Deux types d'algorithmes (et de problèmes) :

algorithme de décision (sortie : Oui ou Non)

- déterminer si deux sommets dans un graphe sont liés par un chemin
- déterminer si une formule est satisfiable
- tester si deux nombres sont premiers entre eux

algorithmes de calcul

- déterminer tous les diviseurs d'un nombre
- déterminer toutes les composantes fortement connexes d'un graphe

Qu'est-ce-qu'un problème ?

Un problème est défini par une entrée et une sortie attendue.

Exemple : accessibilité dans un graphe

- Entrée : un graphe orienté G et deux sommets s et t
- Sortie : Oui si on peut atteindre t depuis s dans G et Non sinon

Attention : ne pas confondre un problème et une ***instance*** (c'est-à-dire une entrée particulière) du problème : un algorithme qui résout un problème doit fournir une solution pour ***toutes*** les instances du problème

Des problèmes plus difficiles que d'autres

Tous les problèmes n'ont pas la même complexité

Il existe des problèmes plus simples à résoudre que d'autres

Il existe des problèmes pour lesquels il n'existe pas d'algorithme (et on peut le prouver) : ce sont les *problèmes indécidables*

Exemple : l'arrêt des machines à deux compteurs

- Entrée : un programme manipulant deux variables entières `c0` et `c1` qui ne peuvent pas être négatives, initialisées à 0, et avec une suite d instructions de la forme suivante (où `c` vaut `c0` ou `c1`) :
  - `l : c:=c+1; goto l1`
  - `l : if (c!=0) c:=c-1; goto l1`
  - `l : if (c==0) goto l1 else goto l2`
  - `H : Halt`
- Sortie : l'instruction `Halt` est-elle atteinte ?

Ce problème est indécidable

Classification des problèmes

On peut classer les problèmes ***décidables*** dans des classes de complexité

Exemples :

- P (ou PTIME) : problèmes résolubles en temps polynomial
- PSPACE : problèmes résolubles en espace polynomial
- EXPTIME : problèmes résolubles en temps exponentiel
- EXPSPACE : problèmes résolubles en espace exponentiel
- ...

$P \subset PSPACE \subset EXPTIME \subset EXPSPACE \subset \ldots$

On sait que certains problèmes sont ***complets*** pour une classe, c'est-à-dire que tous les autres problèmes de cette classe sont (au sens large) « moins durs », c'est-à-dire que tout autre problème de la classe peut être ***réduit*** à ce problème

Classe NP : problèmes pouvant être résolus de façon non-déterministe en temps polynomial

À quoi correspond cette classe ?

ce sont les problèmes pour lesquels on peut donner un ***certificat de validité***, ie une preuve du résultat, qui est vérifiable en temps polynomial

Par exemple, pour savoir si t est atteignable depuis s dans un graphe, un certificat de validité est un chemin de s vers t

Il existe des problèmes dans NP que l'on ne sait pas résoudre en temps polynomial

On ne sait pas si P=NP ou si P$\neq$NP

En pratique, pour résoudre un problème qui est dans NP (mais dont on ne sait pas s'il est dans P), il faut un temps exponentiel pour énumérer tous les certificats

La frontière entre P et NP est parfois subtile

Exemple : les problèmes kSAT

- Entrée : une formule de logique propositionnelle en forme normale conjonctive et chaque clause ne contient au plus que k variables (par exemple, pour $k = 2$, $(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_3)$)
- Sortie : Oui si la formule est satisfaisable, non sinon.

2SAT est dans P, alors que 3SAT est dans NP

Exemple 2 : graphes eulériens ou hamiltoniens

- un graphe est *eulérien* s'il existe un cycle qui visite chaque arête une et une seule fois
- un graphe est *hamiltonien* s'il existe un cycle qui visite chaque sommet une et une seule fois

Déterminer si un graphe est eulérien est dans P,
déterminer si un graphe est hamiltonien est dans NP

Que faut-il savoir faire sur les graphes ?

Maîtriser le cours d'Algorithmique du premier semestre, en particulier :

- parcours en profondeur
- parcours en largeur
- algorithmes de plus court chemin
- déterminer les composantes fortement connexes