Présentation du cours

Rappel O-notation

D.E ZEGOUR

École Supérieure d'Informatique

ESI

Présentation du cours

Introduction

3 parties distinctes:

- Conception
- Complexité de programmes
- Construction

Chaque partie exige un pré-requis

O-notation et graphes

Problèmes de décision et machine de Turing

Systèmes formels et théorie du point fixe.

Présentation du cours

Conception de programmes

Diverses manières de conception d'algorithmes:

- Diviser pour résoudre
- Programmation dynamique
- Recherche systématique de solutions (Parcours en profondeur (Backtracking), Parcours en largeur)
- Recherche avec des heuristiques quand le graphe de solutions est exponentiel.

Illustration avec des problèmes bien connus:

- Tour de Hanoi
- Organisation d'un tournoi entre plusieurs équipes
- Série mondiale
- Multiplications chaînées de matrices
- Problème des 8 reines
- Tic-Tac-Toe
- Problème des cruches d'eau
- Sortie d'un labyrinthe
- Etc.

Présentation du cours

Conception de programmes

Introduit également une forme de l'intelligence artificielle.

Technique du Min-Max
Elagage Alpha-Béta
(Jeu de stratégies)

Présentation de plusieurs algorithmes avec des heuristiques:

- Hill Climbing
- Best First Search
- Branch and Bound
- l'algorithme A*,

Présentation du cours

Complexité

Comme pré-requis:

- Décidabilité des problèmes / défi : "ce qui peut être fait par un ordinateur et ce qui ne peut l'être"

- Machine de Turing a pour but d'étudier la décidabilité des programmes et permet donc de distinguer les problèmes possibles (polynomiales) et non possibles (exponentielles).

L'étude de la complexité a donc pour but de quantifier la mesure d'un algorithme (en général par les paramètres temps et espace

Présentation du cours

Complexité

La complexité a été étudié à travers un modèle de calcul.
Le plus utilisé est sans doute la machine de Turing.

Classification des problèmes :

- Classe P
- Classe NP
- Classe Exp
- Etc.

Notions de

- Réduction
- NP-complétude

Présentation du cours

Construction de programmes

Comme pré-requis :

- Systèmes formels : morphologie, théorie propre, principes de génération, décidabilité, exmples de systèmes formels.

- Théorie du point fixe: rappels mathématiques, concepts de treuilli sur lequel on définit la croissance et la continuité sur les fonctionnelles (fonctions de fonctions). Vu l'importance du théorème du point fixe, il sera présenté et démontré.

Présentation du cours

Construction de programmes

Elle s'intéresse à la forme des programmes.

Diverses manières de programmation sont présentées et étudiées:

- Programmation procédurale : la plus classique, dérive des machines de Von Neumann.
- Programmation non procédurale : basée sur un système formel.

Le Lambda-calcul est le système formel de la programmation fonctionnelle,

La logique des prédicats du premier ordre est celui de la programmation logique.

Présentation du cours

Construction de programmes

Programmation procédurale

Etat de l'art sur toute la programmation procédurale.

Aspects étudiés

- Les différents schémas
- Les transformations
- Les différentes formes de preuves

Présentation du cours

Construction de programmes

Programmation fonctionnelle

Couvre plusieurs parties

- Lambda-calcul
- Machine à réductions : évaluation des lambda-expressions (programme fonctionnel)
- Preuves

- Présentation de Lisp

Présentation du cours

Construction de programmes

Programmation logique

Couvre 3 parties

- Logique des prédicats du premier ordre, le système formel de tous les langages logiques.
- Principe des démonstrateurs automatiques pour interpréter des clauses (programme logique).

Une deuxième forme de l'intelligence artificielle est présentée (A, B, C , $\rightarrow$ But)

- Présentation de Prolog

Présentation du cours

Documentation

http://zegour.esi.dz/Site%20secondaire/Tpgo/Cours_tpgo.htm

http://zegour.esi.dz/Site%20secondaire/Tpgo/Cours_tpgo.htm

Présentation du cours

Module : Théorie de la programmation

D.E ZEGOUR & W.K HIDOUCI

Sommaire: (Sur 30 heures)

Présentation du cours

Séances (1H)	Cours	TD/TP
1	*Concepts préliminaires (1) :* Notation de Landau Parcours de graphes	O-notation Graphes
3	*Réduction de complexité* Méthode descendante (Diviser pour résoudre) Méthode ascendante (Programmation dynamique)	Exemples et TP
6	*Résolution de problèmes* Backtracking Hill-Climbing Best First Search Branch and Bound Algorithme A*	Exemples et TP
4	*Théorie de la complexité* (*Cours détaillé*) Problèmes de décision et langages Modèles de calcul Classes de complexité NP-Complétude	Machine de Turing Réduction Np-Complétude
1	*Concepts préliminaires (2) :* Théorie du point fixe Système Formel	
5	*Programmation impérative* Schémas de programmes Transformations de programmes Preuves formelles	Transformation- Preuve
	Programmation applicative	

http://zegour.esi.dz/Site%20secondaire/Tpgo/Cours_tpgo.htm

Présentation du cours

5	*Programmation applicative* Lambda-calcul Lisp et fonctions d'ordre supérieur Preuves par induction Interprétation des langages fonctionnels	Lambda-calcul et programmation Lisp
5	*Programmation déclarative* Démonstration automatique de théorèmes Prolog et manipulations symboliques Interprétation des langages logiques	Résolution et programmation Prolog

Sujets d'examens corrigés depuis 2013

Examen 2012-2013	Corrigé
Examen 2013-2014	Corrigé
Examen 2014-2015	Corrigé
Examen 2015-2016	Corrigé
Examen 2016-2017	Corrigé
Examen 2017-2018	Corrigé
Examen 2018-2019	Corrigé
Examen 2019-2020	Corrigé

New **Livre en ligne :** Conception de programmes

New **Livre en ligne :** Construction de programmes

http://zegour.esi.dz/Site%20secondaire/Tpgo/Cours_tpgo.htm

Présentation du cours

Présentation du cours

Cours : Théorie de la programmation

(Vidéos et PDFs)

[page en construction]

Pr ZEGOUR DJAMEL EDDINE
ESI : Ecole Supérieure d'Informatique (Alger)
http://zegour.esi.dz/

Cours	PDF	Vidéo
Présentation du cours	PDF	Vidéo
O-notation & Mesure des algorithmes itératifs	PDF	Vidéo
Mesure des algorithmes récursifs	PDF	Vidéo
Diviser pour Résoudre	PDF	Vidéo
Programmation dynamique	PDF	Vidéo
Graphes	PDF	Vidéo
Exploration 1	PDF	Vidéo
Exploration 2	PDF	Vidéo
Heuristiques 1	PDF	Vidéo
Heuristiques 2	PDF	Vidéo
Complexité 1	PDF	Vidéo
Complexité 2	PDF	Vidéo
Complexité 3	PDF	Vidéo
Programmation procédurale 1	PDF	Vidéo

Documents et logiciels annexes:

Livre en ligne : Conception de programmes

Livre en ligne : Construction de programmes

Plus d'informations sur le cours sur la page

Présentation du cours / Rappel O-notation

Introduction

Le temps d ’exécution d'un programme

T(n) = c f(n)

n : nombre de données (par exemple)
c : constante regroupant les facteurs
- la qualité du code généré par le compilateur
- la machine (microprocessur et Ram)

Exemple : $T(n) = c\ n^2$
T(n) : nombre d'instructions exécutées

En pratique, le temps moyen est souvent difficile à déterminer.

Essayer de trouver le cas moyen sinon trouver le cas le plus défavorable

On dira que T(n) est O(f(n))
s'il existe c > 0 et n0 > 0 telles que T(n) ≤ c f(n) pour tout n ≥ n0.

f(n) est le taux de croissance.

f(n) est une limite supérieure du taux de croissance de T(n).

La constante dépend des facteurs liés à la machine et au code

Présentation du cours / Rappel O-notation

Opérations : règle de la somme

Si
T1(n) = O (f(n)) et
T2(n) = O(g(n))
sont les temps d'exécution de 2 fragments de programme P1 et P2,
Alors
le temps d'exécution de P1 suivi de P2 est
T1(n) + T2(n) = O (max (f(n), g(n))

Conséquence : Si g(n) ≤ f(n) pour tout n > n0 alors
O (f(n) + g(n)) = O (f(n))
Exemple : $O(n^2 + n) = O(n^2)$

Opérations : règle du produit

Si
T1(n) = O(f(n)) et
T2(n) = O(g(n))
Alors
T1(n) * T2(n) = O (f(n) * g(n))

Conséquence : ***O(c f(n)) = O(f(n))*** si c > 0.
Exemple : $O(n^2/2) = O(n^2)$

Présentation du cours / Rappel O-notation

Mesure des algorithmes itératifs

1. Affectation, lecture ou écriture : O(1)

2. Séquence d'étapes : règle de la somme. C'est donc le temps de la séquence qui a le plus grand temps d ’exécution.

3. Alternative : se placer dans le cas le plus défavorble

4. Boucle : Règle du produit. C'est le produit du nombre d'itérations de la boucle par le plus grand temps possible pour une exécution du corps.

Présentation du cours / Rappel O-notation

Mesure des algorithmes itératifs : Exemple

```
(1) Pour i:= 1, n-1
(2)  Pour j := n,  i+1, -1
(3)    Si A(j-1) > A(j)
(4)         temps := A(j-1)
(5)         A(j-1) := A(j)
(6)         A(j) := temp
       Fsi
     Fpour
   Fpour
```

Tri par bulles d'un tableau A[1..n]

(4) (5) et (6) prennent chacun O(1)

Règle de la somme : (4) (5) et (6) est O(Max(1, 1, 1)) = O(1).

Pour l'instruction SI, O (Max(1, 1) = O(1)

Boucle (2) à (6) :corps : O(1) ;   boucle : O(n-i)
Règle du produit : O((n-i)*1) = O (n-i).

Boucle (1) (6): corps : O(n-i) ou O(n) (résultat précédent)
boucle : O(n-1) ou O(n)
Règle du produit : O(n.n) = $O(n^2)$

Présentation du cours / Rappel O-notation

Mesure des algorithmes récursifs : Méthode

3 méthodes de résolution

Pour mesurer un algorithme récursif

- trouver l'équation de récurrence
- la résoudre

a) Par substitution (dilater la récurrence)

b) Deviner une solution f(n) et la démontrer par récurrence.

c) Utiliser les solutions de certaines équations de récurrence connues.

Présentation du cours / Rappel O-notation

Résolution par dilatation (Exemple)

```
Tri(L, n) { Supposons n = 2^k }
Si n = 1
   Tri := L
Sinon
    L1 := L [1..n/2]
    L2 := L [n/2+1..n]
    Tri := Fusion( Tri(L1, n/2), tri(L2, n/2))
Fsi
```

T(n) = a si n=1
= 2 T(n/2) + bn Sinon

Équation de récurrence

T(n) = 2 T(n/2) + bn
= 2 [2 T(n/4) + bn/2] + bn
= 4 T[n/4] + 2 bn
= 8 T[n/8] + 3 bn
= ...
= 2^i T[$n/2^i$] + ibn
= ...
= n T[1] + Log(n) bn
= an + Log(n) bn
= n(a + bLog(n))

C'est O(n Log(n))

Présentation du cours / Rappel O-notation

Résolution par devinette (Exemple)

```
Tri(L, n) { Supposons n = 2^k }
Si n = 1
   Tri := L
Sinon
    L1 := L [1..n/2]
    L2 := L [n/2+1..n]
    Tri := Fusion( Tri(L1, n/2), tri(L2, n/2))
Fsi
```

$T(n) = c_1$ si n=1
$= 2\,T(n/2) + c_2 n$ Sinon **(1)**

Équation de récurrence

On veut montrer que T(n) = O(nlog2(n))
c'est à dire $T(n) \leq anLog(n) + b$ pour a et b donnés à partir d'un rang n.

Si n = 1 , $T(1) \leq b$ (On peut prendre $b= c_1$)

On suppose $T(k) \leq akLog(k) + b$ pour tout k < n
et on essaie d 'établir que $t(n) \leq anLogn + b$

Présentation du cours / Rappel O-notation

Résolution par équation de récurrence

1. Équations homogènes

$a_0\, t_n + a_1\, t_{n-1} + \ldots a_k\, t_{n-k} = 0$ **(1)**

admet comme équation caractéristiques

$a_0\, x^k + a_1\, x^{k-1} + \ldots + a_k = 0$

Solutions dans R :

$r_1, r_2, \ldots . r_k$.

Si toutes les solutions r_i sont distinctes, solution de (1) est $t_n = c_1(r_1)^n + c_2(r_2)^n + \ldots . . c_k(r_k)^n$

Si r_j est une solution multiple (de multiplicité m), solution de **(1)** est

$t_n = c_1(r_1)^n + c_2(r_2)^n + \ldots + \mathbf{(c_{j1}(r_j)^n + c_{j2} n\,(r_j)^n + \ldots c_{jm} n^{m-1}(r_j)^n\ )} + \ldots . . + c_k(r_k)^n$

Remarque : les constantes c_j sont déterminées par les conditions initiales

Présentation du cours / Rappel O-notation

Rappel solution

$t_n = c_1(r_1)^n + c_2(r_2)^n + ...c_k(r_k)^n$

Equation homogène (Exemple)

Equation homogène

$t_n - 3t_{n-1} - 4t_{n-2} = 0$ pour $n \geq 2$

$t_0 = 0, t_1 = 1$

Equation caractéristique : $x^2 - 3x - 4 = 0$

Solutions : -1 et 4.

$t_n = c_1(-1)^n + c_2 4^n$

Conditions initiales

$0 = c_1 + c_2$

$1 = - c_1 + 4c_2$

$c_1 = -1/5$ et $c_2 = +1/5$

$t_n = -(1/5)(-1)^n + (1/5)4^n$

c'est $O(4^n)$.

Présentation du cours / Rappel O-notation

Résolution par équation de récurrence

2. Équations non homogènes

$a_0t^n + a_1t^{n-1} + ...a_kt^{n-k} = b_1^nP1(n) + b_2^nP2(n)+....$

b_i : constantes

P_i : des polynômes de degré d_i

Equation caractéristique

$(a_0x^k + a_1x^{k-1}+...a_k)\ (x-b_1)^{d1+1}\ (x-b_2)^{d2+1}\ .. = 0$

Solutions dans R :

$r_1, r_2, ...r_k$.

Si toutes les solutions r_i sont distinctes, solution de (1) est $t_n = c_1(r_1)^n + c_2(r_2)^n + ...c_k(r_k)^n$

Si r_j est une solution multiple (de multiplicité m), solution de (1) est

$t_n = c_1(r_1)^n + c_2(r_2)^n + ... + \mathbf{(c_{j1}(r_j)^n + c_{j2}n\ (r_j)^n + ... c_{jm}n^{m-1}(r_j)^n\ )} + ...+ c_k(r_k)^n$

Remarque : les constantes c_j sont déterminées par les conditions initiales

Présentation du cours /
Rappel O-notation

Rappel : $a_0t^n + a_1t^{n-1} + ...a_kt^{n-k} = b_1{}^nP1(n) + b_2{}^nP2(n)+....$

Rappel $t_n = c_1(r_1)^n + c_2(r_2)^n + ...c_k(r_k)^n$

Equation non homogène (Exemple)

Suite de Fibonacci
Fib(n) := Fib(n-1) + Fib(n-2) si n > 1
Fib(0) = 0, Fib(1) = 1

Equation de récurrence
T(n) = T(n-1) + T(n-2) + b si n > 1
T(n) = a sinon

Equation non homogène
$t_n - t_{n-1} - t_{n-2} = b = 1^n\ b$
$t_0 = 0, t_1 = 1,$

$b_1 = 1, P_1 = b$, b étant un polynôme de degré 0

Équation caractéristique : $(x^2 - x - 1)(x-1) = 0$

$(x-r_1)\ (x-r_2)(x-1) = 0$

$r_1 = (1 + \sqrt{5})/2$
$r_2 = (1 - \sqrt{5})/2$
$r_3 = 1$

Donc $t_n = c_1\ ((1 + \sqrt{5})/2)^n + c_2\ ((1 - \sqrt{5})/2)^n + c_3\ 1^n$

c'est $O(1.6180339^n)$