Module EA4 – Éléments d'Algorithmique

Dominique Poulalhon
dominique.poulalhon@liafa.univ-paris-diderot.fr

Université Paris Diderot
L2 Informatique, Math-Info et EIDD
Année universitaire 2013-2014

Organisation du module

Emploi du temps

- Amphi : 1h30 par semaine, lundi 11h30-13h, amphi 12E
- TD : 2h par semaine
  - Info1 : vendredi 8h30 – 164E – I. Cristescu
    ioana.cristescu@pps.univ-paris-diderot.fr
  - Info2 : jeudi 8h30 – 477F – H. Arfaoui
    heger.arfaoui@liafa.univ-paris-diderot.fr
  - Info3 : vendredi 10h30 – 270F – M. Sighireanu
    mihaela.sighireanu@liafa.univ-paris-diderot.fr
  - Info4 + EIDD : lundi 14h30 – 477F – F. Bossière
    francois.bossiere@gmail.com
  - Math-Info : jeudi 8h30 – 476F – D. Poulalhon
    dominique.poulalhon@liafa.univ-paris-diderot.fr

Organisation du module

Emploi du temps

- Amphi : 1h30 par semaine, lundi 11h30-13h, amphi 12E
- TD : 2h par semaine
  - Info1 : vendredi 8h30 – 164E – I. Cristescu
    ioana.cristescu@pps.univ-paris-diderot.fr
  - Info2 : jeudi 8h30 – 477F – H. Arfaoui
    heger.arfaoui@liafa.univ-paris-diderot.fr
  - Info3 : vendredi 10h30 – 270F – M. Sighireanu
    mihaela.sighireanu@liafa.univ-paris-diderot.fr
  - Info4 + EIDD : lundi 14h30 – 477F – F. Bossière
    francois.bossiere@gmail.com
  - Math-Info : jeudi 8h30 – 476F – D. Poulalhon
    dominique.poulalhon@liafa.univ-paris-diderot.fr
- début des TD : jeudi 23 janvier

Organisation du module

Emploi du temps

- Amphi : 1h30 par semaine, lundi 11h30-13h, amphi 12E
- TD : 2h par semaine
- début des TD : jeudi 23 janvier

Pour nous écrire, toujours mentionner `[EA4]` dans le sujet

Un site web ? sans doute pas, mais un site `Didel`.
Inscrivez-vous !

MCC

- 60% examen final
- 40% contrôle continu (2 ou 3 interros dont probablement une lundi 17 février à la place de l'amphi)

Éléments d'« Algorithmique » ?

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

« algorithme » ?

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

« algorithme » ? = méthode (systématique) de résolution d'un problème

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

« algorithme » ? = méthode (systématique) de résolution d'un problème

pas limité au domaine informatique ; d'ailleurs, de nombreux algorithmes ont été décrits bien avant l'invention des ordinateurs :

- des algorithmes de calcul
- des constructions géométriques
- des recettes de cuisine...

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

« algorithme » ? = méthode (systématique) de résolution d'un problème

pas limité au domaine informatique ; d'ailleurs, de nombreux algorithmes ont été décrits bien avant l'invention des ordinateurs :

- des algorithmes de calcul
- des constructions géométriques
- des recettes de cuisine...

Exemple : comment nourrir un loup ami des lapins ?
(©Jean-Luc Fromental, Grégoire Solotareff)

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

« algorithme » ? = méthode (systématique) de résolution d'un problème

pas limité au domaine informatique ; d'ailleurs, de nombreux algorithmes ont été décrits bien avant l'invention des ordinateurs :

- des algorithmes de calcul
- des constructions géométriques
- des recettes de cuisine...

mais le concept a pris une importance particulière avec l'apparition de machines capables d'exécuter fidèlement et rapidement une suite d'opérations prédéfinie

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

« algorithme » ? = méthode (systématique) de résolution d'un problème

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

« algorithme » ? = méthode (systématique) de résolution d'un problème

trois axes d'étude :

- conception
- preuve de correction

- étude de l'efficacité

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

« algorithme » ? = méthode (systématique) de résolution d'un problème

trois axes d'étude :

- conception
- preuve de correction : un algorithme est *correct* si, pour chaque entrée, il termine en produisant la bonne sortie
- étude de l'efficacité

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

« algorithme » ? = méthode (systématique) de résolution d'un problème

trois axes d'étude :

- conception
- preuve de correction : un algorithme est *correct* si, pour chaque entrée, il termine en produisant la bonne sortie
- étude de l'efficacité : les ressources nécessaires (temps, mémoire) sont-elles raisonnables ? Est-il possible de faire mieux ?

Éléments d'« Algorithmique » ?

= « conception et étude des algorithmes »

« algorithme » ? = méthode (systématique) de résolution d'un problème

trois axes d'étude :

- conception : y a-t-il des techniques générales ?
- preuve de correction : un algorithme est *correct* si, pour chaque entrée, il termine en produisant la bonne sortie
- étude de l'efficacité : les ressources nécessaires (temps, mémoire) sont-elles raisonnables ? Est-il possible de faire mieux ?

Exemple : les opérations arithmétiques

Addition de deux entiers :

$$\begin{array}{rcccc} & 1 & 3 & 5 & 7 \\ + & 2 & 4 & 6 & 8 \\ \hline \end{array}$$

Exemple : les opérations arithmétiques

Addition de deux entiers :

$$\begin{array}{rcccc} & & & \color{magenta}{1} & \\ & 1 & 3 & 5 & \color{magenta}{7} \\ + & 2 & 4 & 6 & \color{magenta}{8} \\ \hline & & & & \color{magenta}{5} \end{array}$$

Exemple : les opérations arithmétiques

Addition de deux entiers :

$$\begin{array}{cccccc} & & & 1 & 1 & \\ & & 1 & 3 & 5 & 7 \\ + & & 2 & 4 & 6 & 8 \\ \hline & & & & 2 & 5 \end{array}$$

Exemple : les opérations arithmétiques

Addition de deux entiers :

$$\begin{array}{cccccc} & & 1 & 1 & & \\ & 1 & 3 & 5 & 7 & \\ + & 2 & 4 & 6 & 8 & \\ \hline & & 8 & 2 & 5 & \end{array}$$

Exemple : les opérations arithmétiques

Addition de deux entiers :

$$\begin{array}{cccccc} & & & 1 & 1 & \\ & & 1 & 3 & 5 & 7 \\ + & & 2 & 4 & 6 & 8 \\ \hline & & 3 & 8 & 2 & 5 \end{array}$$

Exemple : les opérations arithmétiques

Addition de deux entiers :

$$\begin{array}{cccccc} & & 1 & 1 & & \\ & 1 & 3 & 5 & 7 & \\ + & 2 & 4 & 6 & 8 & \\ \hline & 3 & 8 & 2 & 5 & \end{array}$$

```
def addition(nb1, nb2) :
# entiers représentés par des tableaux de chiffres décimaux
  res = []
  retenue = 0
  for (chiffre1, chiffre2) in zip(nb1, nb2) :
    tmp = chiffre1 + chiffre2 + retenue
    retenue = tmp/10
    res.append(tmp%10)
  return res + [retenue]
```

Exemple : les opérations arithmétiques

Addition de deux entiers :

$$\begin{array}{cccccc} & & 1 & 1 & & \\ & 1 & 3 & 5 & 7 & \\ + & 2 & 4 & 6 & 8 & \\ \hline & 3 & 8 & 2 & 5 & \end{array}$$

correction : en montrant l'invariant :

« après l'étape i, $\texttt{res} = n_1 + n_2$ *modulo* 10^i »

complexité en temps : autant d'additions élémentaires que de chiffres dans l'écriture décimale des entiers.

$\Longrightarrow$ « complexité linéaire » (en la taille ℓ des données, la taille étant ici le nombre de chiffres décimaux : $n_1, n_2 \in O(10^\ell)$)

Exemple : les opérations arithmétiques

Multiplication de deux entiers (1)

```
def multiplication(nb1, nb2) :
  res = nb1[:]
  for i in range(1, valeur(nb2)) :
    res = addition(res, nb1)
  return res
```

correction : en montrant l'invariant :

« après l'étape i, `res` $= i \times n_1$ »

complexité en temps : $n_2 - 1$ additions de (grands) entiers, chacune étant de coût linéaire en la taille de n_1

$\Longrightarrow$ complexité en $O(n_2 \times \log(n_1)) = O(\ell \times 10^{\ell})$

Exemple : les opérations arithmétiques

Multiplication de deux entiers (2)

```
def multiplication_par_un_chiffre(nb1, chiffre2) :
  res = []
  retenue = 0
  for chiffre1 in nb1 :
    tmp = chiffre1 * chiffre2 + retenue
    retenue = tmp/10
    res.append(tmp%10)
  return res + [retenue]
```

correction ?

complexité ?

Exemple : les opérations arithmétiques

Multiplication de deux entiers (2)

```
def multiplication(nb1, nb2) :
  res = []
  for (i, chiffre2) in enumerate(nb2) :
    tmp = multiplication_par_un_chiffre(nb1, chiffre2)
    res = addition(res, [0]*i + tmp)
  return res
```

correction ?

complexité ?

Exemple : les opérations arithmétiques

Multiplication de deux entiers (3) : la méthode du paysan russe

```
def multiplication_russe(nb1, nb2) :
# cette fois, les entiers sont vraiment des entiers
  res = 0
  while nb2 != 0 :
    if nb2%2 == 1 : res += nb1
    nb1 *= 2        # ou : nb1 << 1
    nb2 /= 2        # ou : nb2 >> 1
  return res
```

correction ?

complexité ?