Module EA4 – Éléments d'Algorithmique II

Outils pour l'analyse des algorithmes

Dominique Poulalhon

dominique.poulalhon@irif.fr

Université Paris Diderot

L2 Informatique & Math-Info

Année universitaire 2019-2020

Organisation du module

Emploi du temps (en théorie)

- Cours : 2h par semaine, mercredi 13h45-15h45, amphi 2A
  (naturellement) aussi obligatoire que les TD et TP
- TD : 2h par semaine
- TP : 2h par quinzaine (alternativement groupes pairs et impairs)

Organisation du module

Emploi du temps (en théorie)

- Cours : 2h par semaine, mercredi 13h45-15h45, amphi 2A
  (naturellement) aussi obligatoire que les TD et TP
- TD : 2h par semaine
- TP : 2h par quinzaine (alternativement groupes pairs et impairs)

Emploi du temps (en pratique)

dans un contexte de grève, il est probable que certaines des prochaines séances ne soient pas assurées

- pour le CM : impossible de rattraper plus tard, donc annulation pure et simple, et le programme de l'examen en tiendra compte
- pour les TD : très probablement impossible à rattraper, sauf éventuellement en utilisant des créneaux de TP ; on avisera, selon que les groupes sont impactés de manière équitable ou pas
- pour les TP : 6 séances, c'est déjà très peu, donc je voudrais qu'elles aient effectivement toutes lieu ; éventuellement, modifications des semaines utilisées

en particulier : sauf contrordre (via Moodle), *tous les groupes auront un TP la semaine prochaine (du 27 janvier)*

Communication

Responsable du cours : Dominique Poulalhon

`dominique.poulalhon@irif.fr`

Chargés de TD-TP

- Groupe INFO 1 : Matthieu Picantin `picantin@irif.fr`
- Groupe INFO 2 : Simon Mauras `simon.mauras@irif.fr`
- Groupe INFO 3 : Anne Micheli `anne.micheli@irif.fr`
- Groupe INFO 4 : Max Dupré La Tour `maxduprelatour@gmail.com`
- Groupe MI 2 : Giovanni Bernardi `gio@irif.fr`
- Groupe MI 2 : Roberto Mantaci `mantaci@irif.fr`

Communication

Responsable du cours : Dominique Poulalhon

`dominique.poulalhon@irif.fr`

Chargés de TD-TP

- Groupe INFO 1 : Matthieu Picantin `picantin@irif.fr`
- Groupe INFO 2 : Simon Mauras `simon.mauras@irif.fr`
- Groupe INFO 3 : Anne Micheli `anne.micheli@irif.fr`
- Groupe INFO 4 : Max Dupré La Tour `maxduprelatour@gmail.com`
- Groupe MI 2 : Giovanni Bernardi `gio@irif.fr`
- Groupe MI 2 : Roberto Mantaci `mantaci@irif.fr`

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

Communication

Responsable du cours : Dominique Poulalhon

`dominique.poulalhon@irif.fr`

Chargés de TD-TP

- Groupe INFO 1 : Matthieu Picantin `picantin@irif.fr`
- Groupe INFO 2 : Simon Mauras `simon.mauras@irif.fr`
- Groupe INFO 3 : Anne Micheli `anne.micheli@irif.fr`
- Groupe INFO 4 : Max Dupré La Tour `maxduprelatour@gmail.com`
- Groupe MI 2 : Giovanni Bernardi `gio@irif.fr`
- Groupe MI 2 : Roberto Mantaci `mantaci@irif.fr`

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

Un site Moodle pour les annonces, les énoncés et les rendus de TP
donc : Inscrivez-vous ! (dans le bon groupe)

Attention : j'ai désinscrit tout le monde hier soir pour ne pas polluer les boîtes mail des anciens étudiants. Réinscrivez-vous si vous l'aviez déjà fait

Modalités de contrôle des connaissances

Session 1 :

- 60% examen final
- 40% contrôle continu :
  - une note de TD/TP, basée sur l'assiduité, la participation et des rendus d'exercices (sur Moodle donc... Inscrivez-vous !)

Modalités de contrôle des connaissances

Session 1 :

- 60% examen final
- 40% contrôle continu :
  - une note de TD/TP, basée sur l'assiduité, la participation et des rendus d'exercices (sur Moodle donc... Inscrivez-vous !)
  - a priori, deux petits contrôles, date(s) à déterminer

Modalités de contrôle des connaissances

Session 1 :

- 60% examen final
- 40% contrôle continu :
  - une note de TD/TP, basée sur l'assiduité, la participation et des rendus d'exercices (sur Moodle donc... Inscrivez-vous !)
  - a priori, deux petits contrôles, date(s) à déterminer

Modalités de contrôle des connaissances

Session 1 :

- 60% examen final
- 40% contrôle continu :
  - une note de TD/TP, basée sur l'assiduité, la participation et des rendus d'exercices (sur Moodle donc... Inscrivez-vous !)
  - a priori, deux petits contrôles, date(s) à déterminer

Modalités de contrôle des connaissances

Session 1 :

- 60% examen final
- 40% contrôle continu :
  - une note de TD/TP, basée sur l'assiduité, la participation et des rendus d'exercices (sur Moodle donc... Inscrivez-vous !)
  - a priori, deux petits contrôles, date(s) à déterminer

Session 2 : max entre

- examen de session 2

Modalités de contrôle des connaissances

Session 1 :

- 60% examen final
- 40% contrôle continu :
  - une note de TD/TP, basée sur l'assiduité, la participation et des rendus d'exercices (sur Moodle donc... Inscrivez-vous !)
  - a priori, deux petits contrôles, date(s) à déterminer

Session 2 : max entre

- examen de session 2
- 60% examen de session 2 + 40% CC

Thème du cours

algorithmique = « conception et analyse des algorithmes »

Thème du cours

algorithmique = « conception et analyse des algorithmes »

Thème du cours

algorithmique = « conception et analyse des algorithmes »

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

Thème du cours

algorithmique = « conception et analyse des algorithmes »

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

concept non limité à l'informatique – d'ailleurs, de nombreux algorithmes ont été décrits bien avant l'invention des ordinateurs :

- des algorithmes de calcul
  (opérations arithmétiques, approximation de π, de $\sqrt{2}$...)

Thème du cours

algorithmique = « conception et analyse des algorithmes »

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

concept non limité à l'informatique – d'ailleurs, de nombreux algorithmes ont été décrits bien avant l'invention des ordinateurs :

- des algorithmes de calcul
  (opérations arithmétiques, approximation de π, de $\sqrt{2}$...)
- des constructions géométriques
  (milieu d'un segment, parallèles, centre d'un cercle, pentagone régulier...)

Thème du cours

algorithmique = « conception et analyse des algorithmes »

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

concept non limité à l'informatique – d'ailleurs, de nombreux algorithmes ont été décrits bien avant l'invention des ordinateurs :

- des algorithmes de calcul
  (opérations arithmétiques, approximation de π, de $\sqrt{2}$...)
- des constructions géométriques
  (milieu d'un segment, parallèles, centre d'un cercle, pentagone régulier...)
- des recettes de cuisine

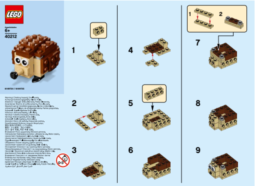
Thème du cours

algorithmique = « conception et analyse des algorithmes »

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

concept non limité à l'informatique – d'ailleurs, de nombreux algorithmes ont été décrits bien avant l'invention des ordinateurs :

- des algorithmes de calcul (opérations arithmétiques, approximation de π, de $\sqrt{2}$...)
- des constructions géométriques (milieu d'un segment, parallèles, centre d'un cercle, pentagone régulier...)
- des recettes de cuisine
- des manuels de construction...



Thème du cours

algorithmique = « conception et analyse des algorithmes »

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

concept non limité à l'informatique – d'ailleurs, de nombreux algorithmes ont été décrits bien avant l'invention des ordinateurs :

- des algorithmes de calcul
  (opérations arithmétiques, approximation de π, de $\sqrt{2}$...)
- des constructions géométriques
  (milieu d'un segment, parallèles, centre d'un cercle, pentagone régulier...)
- des recettes de cuisine
- des manuels de construction...

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

algorithmique = « conception et analyse des algorithmes »

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

Étymologie : Muhammad Ibn Mūsā al-Khuwārizmī

mathématicien perse du début du 9e siècle
« *Kitābu 'l-mukhtasar fī hisābi 'l-jabr wa'l-muqābalah* » ou « *Abrégé du calcul par la restauration et la comparaison* » : considéré comme le premier manuel d'algèbre, explique comment résoudre les équations du second degré
traduit en latin et diffusé en Europe à partir du 12e siècle
(c'est aussi grâce à un de ses livres que se répand la notation positionnelle décimale venue d'Inde)

le terme algorithme est d'abord utilisé pour désigner les méthodes (de calcul) utilisant des chiffres, par opposition au calcul traditionnel (du latin *calculus*, petit caillou) avec des abaques

Thème du cours

algorithmique = « conception et analyse des algorithmes »

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

Thème du cours

algorithmique = « conception et analyse des algorithmes »

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

trois axes d'étude :

- conception
- preuve de correction
- étude de l'efficacité

Thème du cours

algorithmique = « conception et analyse 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

Thème du cours

algorithmique = « conception et analyse 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 ?

Thème du cours

algorithmique = « conception et analyse 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 ?

Thème du cours

algorithmique = « conception et analyse 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 ?

(et au passage, on apprendra un peu de Python, parce que c'est un joli langage particulièrement adapté à l'algorithmique)

Exemple : les opérations arithmétiques

Addition de deux entiers :

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

Exemple : les opérations arithmétiques

Addition de deux entiers :

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

Exemple : les opérations arithmétiques

Addition de deux entiers :

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

Exemple : les opérations arithmétiques

Addition de deux entiers :

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

Exemple : les opérations arithmétiques

Addition de deux entiers :

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

Exemple : les opérations arithmétiques

Addition de deux entiers :

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

```
def addition(nb1, nb2) :
# nb1 et nb2 entiers représentés par des tableaux de chiffres décimaux
# (en commençant par les unités)
  res = []
  retenue = 0
  # parcours parallèle des deux tableaux :
  for (chiffre1, chiffre2) in zip(nb1, nb2) :
    tmp = chiffre1 + chiffre2 + retenue
    retenue = tmp//10   # division euclidienne (en python3)
    res.append(tmp%10)          # ajout à la fin du tableau
  return res + [retenue]      # concaténation de 2 tableaux
```

Exemple : les opérations arithmétiques

Addition de deux entiers :

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

correction : en montrant l'invariant :

« après i tours de boucle, $\texttt{res} = n_1 + n_2$ *modulo* 10^i »

Exemple : les opérations arithmétiques

Addition de deux entiers :

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

correction : en montrant l'invariant :

« après i tours de boucle, $\mathtt{res} = n_1 + n_2$ *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) :
# nb1  représenté par un tableau de chiffres
# nb2  un entier
  res = nb1[:]    # copie du tableau nb1
  for i in range(2, nb2+1) : # répéter nb2-1 fois
    res = addition(res, nb1)
  return res
```

correction : en montrant l'invariant :

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

1. (ou plus précisément $n_2 - 1$)

Exemple : les opérations arithmétiques

Multiplication de deux entiers (1)

```
def multiplication(nb1, nb2) :
# nb1  représenté par un tableau de chiffres
# nb2  un entier
  res = nb1[:]    # copie du tableau nb1
  for i in range(2, nb2+1) : # répéter nb2-1 fois
    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 du résultat $n_1 n_2$ – donc en $\log(n_1 n_2) = \log(n_1) + \log(n_2)$

$\Longrightarrow$ complexité en $O(n_2 \times \log(n_1 n_2))$,

c'est-à-dire $O(\ell \times 10^{\ell})$ si les deux entiers sont de taille ℓ

1. (ou plus précisément $n_2 - 1$)

Exemple : les opérations arithmétiques

Multiplication de deux entiers (2)

```
def multiplication_par_un_chiffre(nb1, chiffre2) :
# nb1  représenté par un tableau de chiffres
  res = []
  retenue = 0
  for chiffre1 in nb1 :
    tmp = chiffre1 * chiffre2 + retenue
    retenue = tmp//10     # division euclidienne
    res.append(tmp%10)
  return res + [retenue]
```

correction : en montrant l'invariant :

« après i tours de boucle, $\texttt{res} \equiv n_1 \times \mathit{chiffre}_2 \mod 10^i$ »

complexité en temps : un tour de boucle par chiffre de n_1, de coût constant

$\Longrightarrow$ complexité en $O(\log(n_1)) = O(\ell)$ si les nombres sont de taille ℓ

Exemple : les opérations arithmétiques

Multiplication de deux entiers (2)

```
def multiplication(nb1, nb2) :
# nb1, nb2  représentés par des tableaux de chiffres
  res = []
  # parcours du tableau avec itération sur les couples
  # (indice, contenu) de chaque case
  for (i, chiffre2) in enumerate(nb2) :
    tmp = multiplication_par_un_chiffre(nb1, chiffre2)
    res = addition(res, [0]*i + tmp)
  return res
```

correction : en montrant l'invariant :

« après l'étape i, $\texttt{res} \equiv n_1 \times n_2 \mod 10^i$ »

complexité en temps : un tour de boucle par chiffre de n_2, chacun de complexité linéaire en la taille du résultat

$\Longrightarrow$ complexité en $O(\ell^2)$ si les nombres sont de taille ℓ

Exemple : les opérations arithmétiques

Puissance (d'un entier par exemple) (1)

```
def puissance(nb1, nb2) :
# nb1 un élément supportant la multiplication, nb2 un entier
  res = 1
  for i in range(nb2) :
    res *= nb1
  return res
```

correction : en montrant l'invariant :

« après i tours de boucle, $\mathtt{res} = n_1^i$ »

Exemple : les opérations arithmétiques

Puissance (d'un entier par exemple) (1)

```
def puissance(nb1, nb2) :
# nb1 un élément supportant la multiplication, nb2 un entier
  res = 1
  for i in range(nb2) :
    res *= nb1
  return res
```

correction : en montrant l'invariant :

$$\text{« après } i \text{ tours de boucle, } \mathtt{res} = n_1^i \text{ »}$$

complexité : chaque tour de boucle a la complexité d'une multiplication – donc $O(\ell^2)$ par la méthode précédente pour n_1, n_2 entiers de taille ℓ

$\Longrightarrow \ O(\ell^2 \times 10^\ell)$ si n_1, n_2 entiers de taille ℓ

Exemple : les opérations arithmétiques

Puissance (2) : l'exponentiation binaire

```
def puissance(nb1, nb2) :
# nb1 un élément supportant la multiplication, nb2 un entier
  if nb2 == 0 : return 1
  tmp = puissance(nb1, nb2//2)
  carre = tmp * tmp
  if nb2%2 == 0 : return carre
  else : return nb1 * carre
```

correction ?

Exemple : les opérations arithmétiques

Puissance (2) : l'exponentiation binaire

```
def puissance(nb1, nb2) :
# nb1 un élément supportant la multiplication, nb2 un entier
  if nb2 == 0 : return 1
  tmp = puissance(nb1, nb2//2)
  carre = tmp * tmp
  if nb2%2 == 0 : return carre
  else : return nb1 * carre
```

correction ? par récurrence (forte) sur n_2

Exemple : les opérations arithmétiques

Puissance (2) : l'exponentiation binaire

```
def puissance(nb1, nb2) :
# nb1 un élément supportant la multiplication, nb2 un entier
  if nb2 == 0 : return 1
  tmp = puissance(nb1, nb2//2)
  carre = tmp * tmp
  if nb2%2 == 0 : return carre
  else : return nb1 * carre
```

correction ? par récurrence (forte) sur n_2

complexité ?

Exemple : les opérations arithmétiques

Puissance (2) : l'exponentiation binaire

```
def puissance(nb1, nb2) :
# nb1 un élément supportant la multiplication, nb2 un entier
  if nb2 == 0 : return 1
  tmp = puissance(nb1, nb2//2)
  carre = tmp * tmp
  if nb2%2 == 0 : return carre
  else : return nb1 * carre
```

correction ? par récurrence (forte) sur n_2

complexité ? chaque appel récursif nécessite 1 ou 2 multiplications,