Introduction à la complexité d'algorithmes

Résumé des épisodes précédents . . .

Nous avons vu les concepts suivants de **programmation impérative** :

- Instructions conditionnelles et itératives
- Fonctions (avec documentation et tests)
- Variables, tableaux (2D), collections
- Entrées-sorties, fichiers
- Modularité, compilation séparée
- Débogueur pas à pas

Résumé des épisodes précédents . . .

Nous avons vu les concepts suivants de **programmation impérative** :

- Instructions conditionnelles et itératives
- Fonctions (avec documentation et tests)
- Variables, tableaux (2D), collections
- Entrées-sorties, fichiers
- Modularité, compilation séparée
- Débogueur pas à pas

C'était la partie **technologique** du cours

Résumé des épisodes précédents . . .

Nous avons vu les concepts suivants de **programmation impérative** :

- Instructions conditionnelles et itératives
- Fonctions (avec documentation et tests)
- Variables, tableaux (2D), collections
- Entrées-sorties, fichiers
- Modularité, compilation séparée
- Débogueur pas à pas

C'était la partie **technologique** du cours

Pourquoi aller plus loin ?

Résumé des épisodes précédents . . .

Nous avons vu les concepts suivants de **programmation impérative** :

- Instructions conditionnelles et itératives
- Fonctions (avec documentation et tests)
- Variables, tableaux (2D), collections
- Entrées-sorties, fichiers
- Modularité, compilation séparée
- Débogueur pas à pas

C'était la partie **technologique** du cours

Pourquoi aller plus loin ?

Passage à l'échelle !

Résumé des épisodes précédents . . .

Nous avons vu les concepts suivants de **programmation impérative** :

- Instructions conditionnelles et itératives
- Fonctions (avec documentation et tests)
- Variables, tableaux (2D), collections
- Entrées-sorties, fichiers
- Modularité, compilation séparée
- Débogueur pas à pas

C'était la partie **technologique** du cours

Pourquoi aller plus loin ?

Passage à l'échelle !

Calculs longs ? Voire impossibles ?

Quelques questions

On a un calcul à faire

Prédire combien de temps il va prendre ?

Quelques questions

On a un calcul à faire

Prédire combien de temps il va prendre ?

On a deux algorithmes

Lequel est le plus rapide ?

Quelques questions

On a un calcul à faire

Prédire combien de temps il va prendre ?

On a deux algorithmes

Lequel est le plus rapide ?

On a un algorithme

Existe-t-il un meilleur algorithme ?

Quelques questions

On a un calcul à faire

Prédire combien de temps il va prendre ?

On a deux algorithmes

Lequel est le plus rapide ?

On a un algorithme

Existe-t-il un meilleur algorithme ?

On a un problème

Est-il résoluble en théorie ?

Quelques questions

On a un calcul à faire

Prédire combien de temps il va prendre ?

On a deux algorithmes

Lequel est le plus rapide ?

On a un algorithme

Existe-t-il un meilleur algorithme ?

On a un problème

Est-il résoluble en théorie ? en pratique ?

Exemple : recherche naïve dans un tableau

Je recherche le nom « Zorro » dans un annuaire comme suit :

1. Je pars du début de l'annuaire
2. Je compare le nom avec « Zorro »
3. Si oui, j'ai terminé
4. Sinon, je recommence en 2. avec le nom suivant

Exemple : recherche naïve dans un tableau

Je recherche le nom « Zorro » dans un annuaire comme suit :

1. Je pars du début de l'annuaire
2. Je compare le nom avec « Zorro »
3. Si oui, j'ai terminé
4. Sinon, je recommence en 2. avec le nom suivant

Questions

– Combien est-ce que cela va **me** prendre de temps ?

Exemple : recherche naïve dans un tableau

Je recherche le nom « Zorro » dans un annuaire comme suit :

1. Je pars du début de l'annuaire
2. Je compare le nom avec « Zorro »
3. Si oui, j'ai terminé
4. Sinon, je recommence en 2. avec le nom suivant

Questions

– Combien est-ce que cela va **me** prendre de temps ?
– Combien est-ce que cela prendra de temps **à un ordinateur** ?

Synthèse

On s'est donné :

– un **problème** : rechercher un mot dans un dictionnaire

Synthèse

On s'est donné :

- un **problème** : rechercher un mot dans un dictionnaire
- un **algorithme** pour le résoudre : recherche naïve

Synthèse

On s'est donné :

- un **problème** : rechercher un mot dans un dictionnaire
- un **algorithme** pour le résoudre : recherche naïve
- un **modèle de calcul** :
  1. Une mesure de **la taille d'une instance du problème** : le nombre n de mots du dictionnaire
  2. Un choix d'**opérations élémentaires** : comparer deux mots

Synthèse

On s'est donné :

- un **problème** : rechercher un mot dans un dictionnaire
- un **algorithme** pour le résoudre : recherche naïve
- un **modèle de calcul** :
  1. Une mesure de **la taille d'une instance du problème** : le nombre n de mots du dictionnaire
  2. Un choix d'**opérations élémentaires** : comparer deux mots

Définition (Complexité de l'algorithme)

Le **nombre d'opérations élémentaires** effectuées par l'algorithme pour résoudre un problème de taille n dans ce modèle de calcul

Synthèse

On s'est donné :

- un **problème** : rechercher un mot dans un dictionnaire
- un **algorithme** pour le résoudre : recherche naïve
- un **modèle de calcul** :
  1. Une mesure de **la taille d'une instance du problème** : le nombre n de mots du dictionnaire
  2. Un choix d'**opérations élémentaires** : comparer deux mots

Définition (Complexité de l'algorithme)

Le **nombre d'opérations élémentaires** effectuées par l'algorithme pour résoudre un problème de taille n dans ce modèle de calcul

Exemple d'application

Prédire le temps nécessaire pour résoudre n'importe quelle instance du problème

Variantes

Complexité au pire

Exemple : n opérations pour la recherche naïve

Variantes

Complexité au pire

Exemple : n opérations pour la recherche naïve

Complexité en moyenne

Exemple : $\frac{n}{2}$ opérations pour la recherche naïve

Variantes

Complexité au pire

Exemple : n opérations pour la recherche naïve

Complexité en moyenne

Exemple : $\frac{n}{2}$ opérations pour la recherche naïve

Complexité pour d'autres resources

La même stratégie peut s'appliquer à toutes les autres **ressources** :

- Bande passante sur le réseau
- Consommation électrique (10% de l'électrité mondiale !)
- ...

Variantes

Complexité au pire

Exemple : n opérations pour la recherche naïve

Complexité en moyenne

Exemple : $\frac{n}{2}$ opérations pour la recherche naïve

Complexité pour d'autres resources

La même stratégie peut s'appliquer à toutes les autres **ressources** :

- Bande passante sur le réseau
- Consommation électrique (10% de l'électrité mondiale !)
- ...

En particulier la **complexité en mémoire** : combien faut-il de mémoire pour exécuter l'algorithme (au pire, en moyenne, . . .)

Variantes

Complexité au pire

Exemple : n opérations pour la recherche naïve

Complexité en moyenne

Exemple : $\frac{n}{2}$ opérations pour la recherche naïve

Complexité pour d'autres resources

La même stratégie peut s'appliquer à toutes les autres **ressources** :

- Bande passante sur le réseau
- Consommation électrique (10% de l'électrité mondiale !)
- ...

En particulier la **complexité en mémoire** : combien faut-il de mémoire pour exécuter l'algorithme (au pire, en moyenne, . . .)

Exemple

Un algorithme a une complexité en mémoire de n^2 octets
Que peut-on dire de sa complexité en temps ?

Exercices

Donner des algorithmes et leur complexité au pire et en moyenne pour les problèmes suivants :

1. Calculer la somme de deux vecteurs de $\mathbb{Q}^n$

Exercices

Donner des algorithmes et leur complexité au pire et en moyenne pour les problèmes suivants :

1. Calculer la somme de deux vecteurs de $\mathbb{Q}^n$
2. Afficher une image

Exercices

Donner des algorithmes et leur complexité au pire et en moyenne pour les problèmes suivants :

1. Calculer la somme de deux vecteurs de $\mathbb{Q}^n$
2. Afficher une image
3. Rechercher la position d'un élément dans un tableau

Exercices

Donner des algorithmes et leur complexité au pire et en moyenne pour les problèmes suivants :

1. Calculer la somme de deux vecteurs de $\mathbb{Q}^n$
2. Afficher une image
3. Rechercher la position d'un élément dans un tableau
4. Rechercher la position d'un élément dans un tableau trié

Exercices

Donner des algorithmes et leur complexité au pire et en moyenne pour les problèmes suivants :

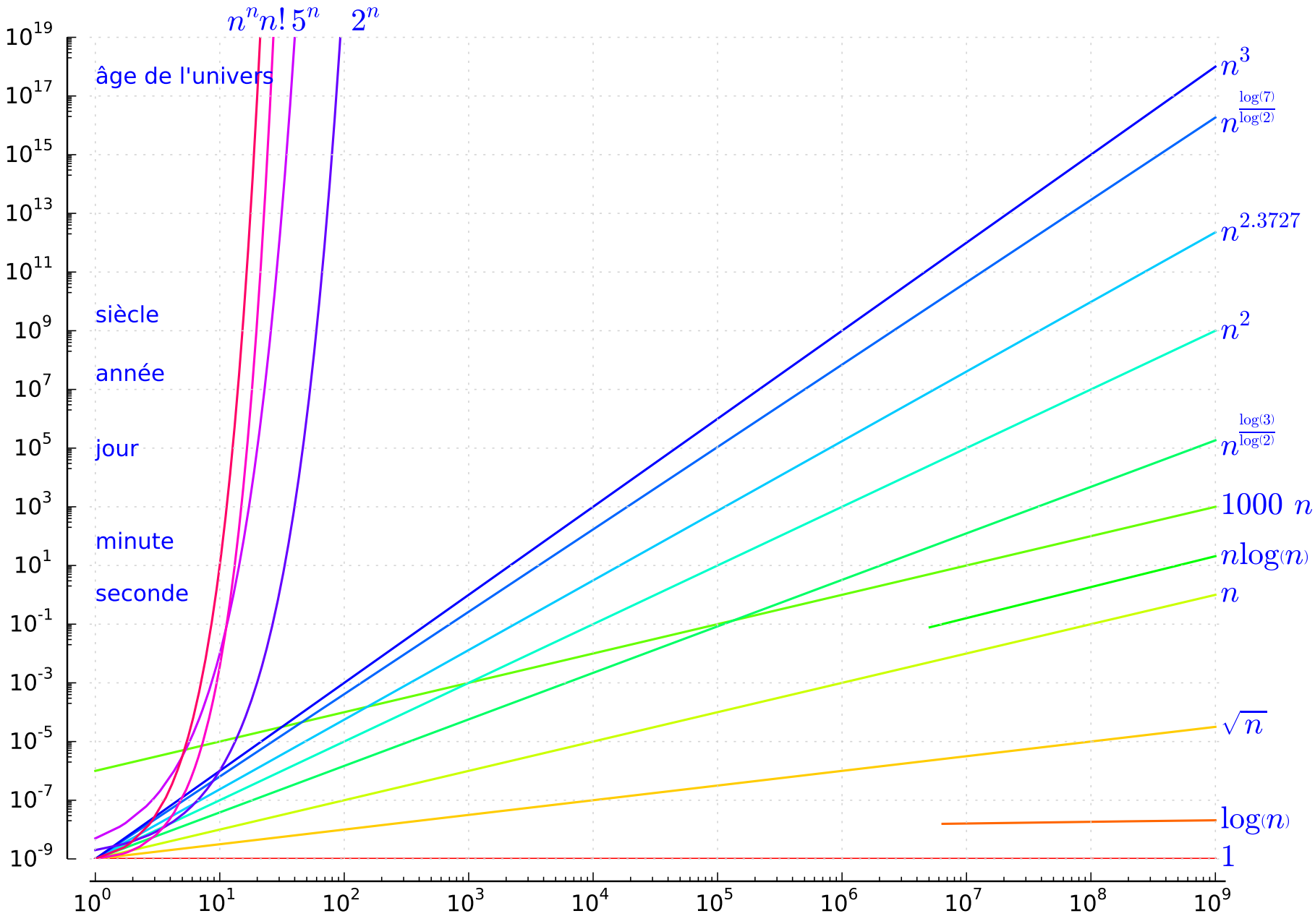
1. Calculer la somme de deux vecteurs de $\mathbb{Q}^n$
2. Afficher une image
3. Rechercher la position d'un élément dans un tableau
4. Rechercher la position d'un élément dans un tableau trié
5. Essayer tous les codes pour une porte à digicode ?

Exercices

Donner des algorithmes et leur complexité au pire et en moyenne pour les problèmes suivants :

1. Calculer la somme de deux vecteurs de $\mathbb{Q}^n$
2. Afficher une image
3. Rechercher la position d'un élément dans un tableau
4. Rechercher la position d'un élément dans un tableau trié
5. Essayer tous les codes pour une porte à digicode ?
6. Essayer tous les mots de passe pour un compte en ligne

Quelques courbes de complexité



Ordres de grandeurs

Exemple

Un algorithme en $1000n$ est meilleur qu'un algorithme en n^2 (pour des instances grandes)

Ordres de grandeurs

Exemple

Un algorithme en $1000n$ est meilleur qu'un algorithme en n^2
(pour des instances grandes)

Remarque

La plupart du temps, il suffit d'avoir un **ordre de grandeur** du nombre d'opérations : les constantes sont sans grande importance

Ordres de grandeurs

Exemple

Un algorithme en $1000n$ est meilleur qu'un algorithme en n^2
(pour des instances grandes)

Remarque

La plupart du temps, il suffit d'avoir un **ordre de grandeur** du nombre d'opérations : les constantes sont sans grande importance

Mais voir aussi l'article Constant Time Factors do Matter

Ordres de grandeurs

Exemple

Un algorithme en $1000n$ est meilleur qu'un algorithme en n^2
(pour des instances grandes)

Remarque

La plupart du temps, il suffit d'avoir un **ordre de grandeur** du nombre d'opérations : les constantes sont sans grande importance

Mais voir aussi l'article Constant Time Factors do Matter

Définitions (Ordres de grandeur)

Soient f et g deux fonctions de $\mathbb{N}$ dans $\mathbb{N}$
Par exemple : les complexités de deux algorithmes

$f = O(g)$ si f est au plus **du même ordre de grandeur** que g :
il existe une constante a telle que $f(n) \leq a\, g(n)$

Arithmétique sur les ordres de grandeur

Exemples

1. $O(4n + 3) = O(n)$
2. $O(\log n) + O(\log n) = O(\log n)$
3. $O(n^2) + O(n) = O(n^2)$
4. $O(n^3)O(n^2 \log n) = O(n^5 \log n)$

C. Complexité d'un problème

Problème

On a vu dans un exercice précédent un algorithme pour calculer la moyenne d'un tableau de nombres et on a obtenu sa complexité : $O(n)$

Existe-t-il un meilleur algorithme ?

C. Complexité d'un problème

Problème

On a vu dans un exercice précédent un algorithme pour calculer la moyenne d'un tableau de nombres et on a obtenu sa complexité : $O(n)$

Existe-t-il un meilleur algorithme ?

Définitions

- La **complexité d'un problème** est la complexité du meilleur algorithme pour le résoudre
- Un algorithme est **optimal** si sa complexité est celle du problème

C. Complexité d'un problème

Problème

On a vu dans un exercice précédent un algorithme pour calculer la moyenne d'un tableau de nombres et on a obtenu sa complexité : $O(n)$

Existe-t-il un meilleur algorithme ?

Définitions

- La **complexité d'un problème** est la complexité du meilleur algorithme pour le résoudre
- Un algorithme est **optimal** si sa complexité est celle du problème

Exercice

Problème : recherche de la position d'un élément dans un tableau trié

1. Quelle est la complexité du problème ?
2. La recherche dichotomique est-elle optimale ?

Exercices

Évaluer au mieux la complexité des problèmes suivants :

1. Calcul du n-ième nombre de Fibonacci
2. Calcul du pgcd de deux nombres
3. Recherche d'un échec et mat en 4 coups à partir d'une position donnée aux échecs
4. ♣ Recherche du plus court chemin entre deux stations de métro à Paris
5. Problème du sac-à-dos : étant donné un ensemble d'objets de hauteur et de poids variables, et un sac à dos de hauteur donnée, charger au maximum le sac-à-dos ?
6. Calcul de la n-ième décimale de $\sqrt{2}$

D. Calculabilité

Définition

Un problème est **calculable** s'il existe un algorithme pour le résoudre

D. Calculabilité

Définition

Un problème est **calculable** s'il existe un algorithme pour le résoudre

Théorème (Thèse de Chuch-Türing)

La calculabilité d'un problème est une notion robuste qui ne dépend ni du langage de programmation, ni de l'architecture, ni ...

- *Formalisation de plusieurs* **modèles de calculs**,
  dont les **machines de Türing**
- *Démonstration de l'équivalence de ces modèles*
- *Ces modèles coïncident avec la notion intuitive*

D. Calculabilité

Définition

Un problème est **calculable** s'il existe un algorithme pour le résoudre

Théorème (Thèse de Chuch-Türing)

La calculabilité d'un problème est une notion robuste qui ne dépend ni du langage de programmation, ni de l'architecture, ni ...

- *Formalisation de plusieurs* **modèles de calculs**, *dont les* **machines de Türing**
- *Démonstration de l'équivalence de ces modèles*
- *Ces modèles coïncident avec la notion intuitive*

Voir aussi

- Des notes d'un excellent exposé de David Harel : **Computers are not omnipotent**
- Une vidéo : une machine de Türing en lego

Exemples de problèmes non calculable

Tiré de p. 8 de **Computers are not omnipotent**

Exemples de problèmes non calculable

Tiré de p. 8 de **Computers are not omnipotent**

Problème de pavage

- Entrée : un ensemble de tuiles
- Règles : couleurs qui s'ajustent, pas de rotation
- Sortie : Peut-on paver le plan avec ces tuiles ?

Exemples de problèmes non calculable

Tiré de p. 8 de **Computers are not omnipotent**

Problème de pavage

- Entrée : un ensemble de tuiles
- Règles : couleurs qui s'ajustent, pas de rotation
- Sortie : Peut-on paver le plan avec ces tuiles ?

Exemples

- Peut-on paver le plan avec les tuiles suivantes ?

Exemples de problèmes non calculable

Tiré de p. 8 de **Computers are not omnipotent**

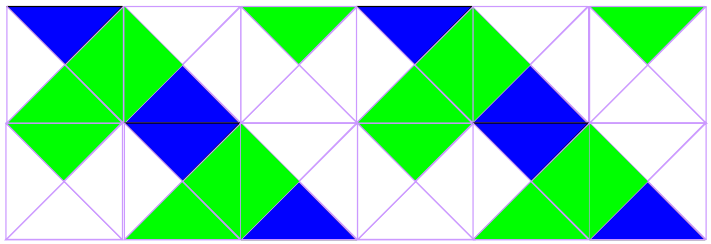
Problème de pavage

- Entrée : un ensemble de tuiles
- Règles : couleurs qui s'ajustent, pas de rotation
- Sortie : Peut-on paver le plan avec ces tuiles ?

Exemples

- Peut-on paver le plan avec les tuiles suivantes ?

Oui :



Exemples de problèmes non calculable

Tiré de p. 8 de **Computers are not omnipotent**

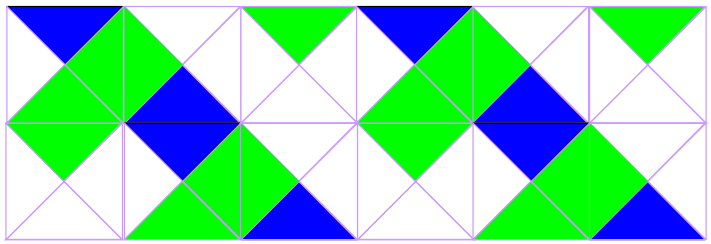
Problème de pavage

- Entrée : un ensemble de tuiles
- Règles : couleurs qui s'ajustent, pas de rotation
- Sortie : Peut-on paver le plan avec ces tuiles ?

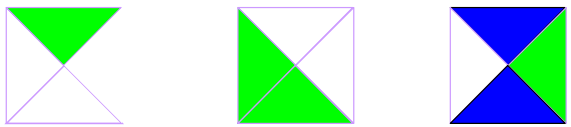
Exemples

- Peut-on paver le plan avec les tuiles suivantes ?

Oui :



- Peut-on paver le plan avec les tuiles suivantes ?



Exemples de problèmes non calculable

Tiré de p. 8 de **Computers are not omnipotent**

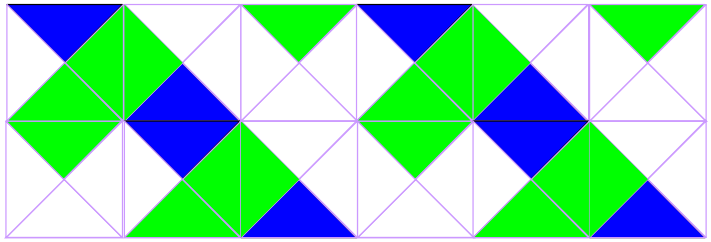
Problème de pavage

- Entrée : un ensemble de tuiles
- Règles : couleurs qui s'ajustent, pas de rotation
- Sortie : Peut-on paver le plan avec ces tuiles ?

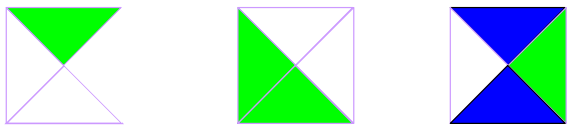
Exemples

- Peut-on paver le plan avec les tuiles suivantes ?

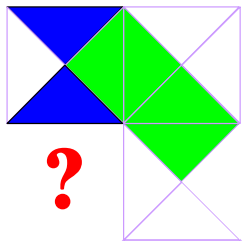
Oui :



- Peut-on paver le plan avec les tuiles suivantes ?



Non :



?

Exemples de problèmes non calculable

Tiré de p. 8 de **Computers are not omnipotent**

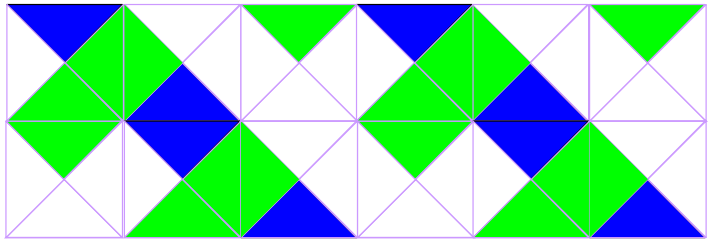
Problème de pavage

- Entrée : un ensemble de tuiles
- Règles : couleurs qui s'ajustent, pas de rotation
- Sortie : Peut-on paver le plan avec ces tuiles ?

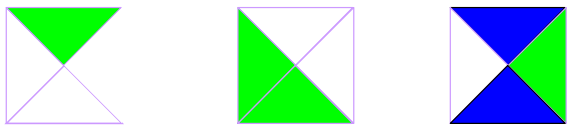
Exemples

- Peut-on paver le plan avec les tuiles suivantes ?

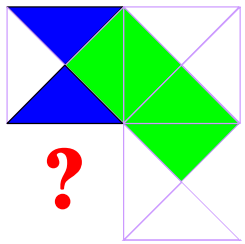
Oui :



- Peut-on paver le plan avec les tuiles suivantes ?



Non :



?

Théorème (Incalculabilité)

Il n'existe pas de programme `PAVE(T)` qui résout ce problème

Exemples de problèmes non calculable

Contexte : vérifier automatiquement qu'un programme fonctionne ?

Exemples de problèmes non calculable

Contexte : vérifier automatiquement qu'un programme fonctionne ?

Problème de l'arrêt

- P : un programme (par exemple écrit en C++)
- D : des données
- Question : est-ce que le programme P s'arrête si on le lance sur les données D ?

Exemples de problèmes non calculable

Contexte : vérifier automatiquement qu'un programme fonctionne ?

Problème de l'arrêt

- P : un programme (par exemple écrit en C++)
- D : des données
- Question : est-ce que le programme P s'arrête si on le lance sur les données D ?

Théorème

Il n'existe pas de programme `HALT(P,D)` qui résout ce problème

Exemples de problèmes non calculable

Contexte : vérifier automatiquement qu'un programme fonctionne ?

Problème de l'arrêt

- P : un programme (par exemple écrit en C++)
- D : des données
- Question : est-ce que le programme P s'arrête si on le lance sur les données D ?

Théorème

Il n'existe pas de programme `HALT(P,D)` qui résout ce problème

Démonstration.

Paradoxe du type :
« Le barbier rase tous ceux qui ne se rasent pas eux-mêmes » □

Calculabilité pratique

Un problème calculable en théorie peut être incalculable en pratique !

Calculabilité pratique

Un problème calculable en théorie peut être incalculable en pratique !

Exemple (Jeu d'échec)

- À chaque étape, il existe un nombre fini de coups possibles
- Donc il est possible d'explorer la suite du jeu pour chaque coup
- Mais il faut examiner jusqu'à 10^{19} coups pour décider de chaque déplacement
- Difficile !
- Programmes au niveau « champion du monde »

Calculabilité pratique

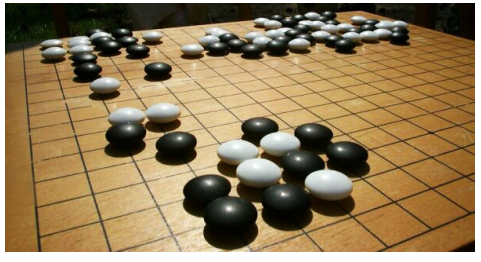
Un problème calculable en théorie peut être incalculable en pratique !

Exemple (Jeu d'échec)

- À chaque étape, il existe un nombre fini de coups possibles
- Donc il est possible d'explorer la suite du jeu pour chaque coup
- Mais il faut examiner jusqu'à 10^{19} coups pour décider de chaque déplacement
- Difficile !
- Programmes au niveau « champion du monde »

Exemple (Jeu de Go)

- Damier : 19x19, règles beaucoup plus simples
- Explosion beaucoup plus rapide : 10^{600} coups !
- Programmes au niveau « amateur »



Calculabilité pratique

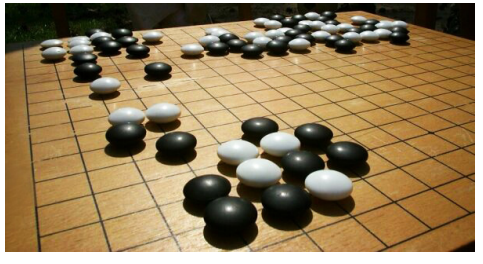
Un problème calculable en théorie peut être incalculable en pratique !

Exemple (Jeu d'échec)

- À chaque étape, il existe un nombre fini de coups possibles
- Donc il est possible d'explorer la suite du jeu pour chaque coup
- Mais il faut examiner jusqu'à 10^{19} coups pour décider de chaque déplacement
- Difficile !
- Programmes au niveau « champion du monde »

Exemple (Jeu de Go)

- Damier : 19x19, règles beaucoup plus simples
- Explosion beaucoup plus rapide : 10^{600} coups !
- Programmes au niveau ~~« amateur »~~
  2016 : AlphaGo et apprentissage profond !



Calculabilité pratique

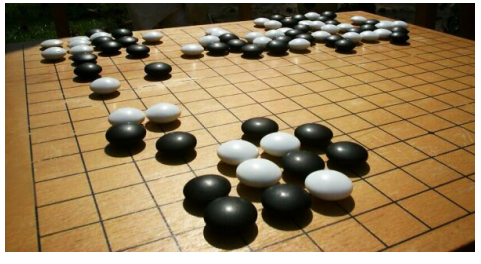
Un problème calculable en théorie peut être incalculable en pratique !

Exemple (Jeu d'échec)

- À chaque étape, il existe un nombre fini de coups possibles
- Donc il est possible d'explorer la suite du jeu pour chaque coup
- Mais il faut examiner jusqu'à 10^{19} coups pour décider de chaque déplacement
- Difficile !
- Programmes au niveau « champion du monde »

Exemple (Jeu de Go)

- Damier : 19x19, règles beaucoup plus simples
- Explosion beaucoup plus rapide : 10^{600} coups !
- Programmes au niveau ~~« amateur »~~
  2016 : AlphaGo et apprentissage profond !



Mais voir : p. 25 de **Computers are not omnipotent**

Résumé

Complexité

- Mesure quantitative de la difficulté d'un problème
- Mesure quantitative de la performance d'un algorithme
- Notions robustes !

Résumé

Complexité

- Mesure quantitative de la difficulté d'un problème
- Mesure quantitative de la performance d'un algorithme
- Notions robustes ! (Page Wikipedia)

Résumé

Complexité

- Mesure quantitative de la difficulté d'un problème
- Mesure quantitative de la performance d'un algorithme
- Notions robustes ! (Page Wikipedia)

Les ordinateurs ne sont pas omnipotents

- Problèmes incalculables / calculables
- Frontière robuste ! (Church-Türing)

Résumé

Complexité

- Mesure quantitative de la difficulté d'un problème
- Mesure quantitative de la performance d'un algorithme
- Notions robustes ! (Page Wikipedia)

Les ordinateurs ne sont pas omnipotents

- Problèmes incalculables / calculables
- Frontière robuste ! (Church-Türing) (Page Wikipedia)

Résumé

Complexité

- Mesure quantitative de la difficulté d'un problème
- Mesure quantitative de la performance d'un algorithme
- Notions robustes ! (Page Wikipedia)

Les ordinateurs ne sont pas omnipotents

- Problèmes incalculables / calculables
- Frontière robuste ! (Church-Türing) (Page Wikipedia)

Humains et ordinateurs

- Dans quels domaines l'humain est / sera plus fort ?
- Dans quels domaines l'ordinateur est / sera plus fort ?

Résumé

Complexité

- Mesure quantitative de la difficulté d'un problème
- Mesure quantitative de la performance d'un algorithme
- Notions robustes ! (Page Wikipedia)

Les ordinateurs ne sont pas omnipotents

- Problèmes incalculables / calculables
- Frontière robuste ! (Church-Türing) (Page Wikipedia)

Humains et ordinateurs

- Dans quels domaines l'humain est / sera plus fort ?
- Dans quels domaines l'ordinateur est / sera plus fort ?

- Qu'est-ce que l'intelligence ?
  Voir : Computing Machinery and Intelligence, Alan Türing, 1950 (Page Wikipedia)