

TC Informatique

PC N° 1

9 Novembre 2000

- François Sillion projet iMAGIS
 IMAG-INRIA Rhône-Alpes
- Patrick Gros projet VISTA
 IRISA, Rennes

<http://w3.edu.polytechnique.fr/informatique>

Le Tronc Commun d'informatique

- Organisation
 - 10 PC + 10 TD
 - Compositions HC (13/12) + CC (14/02/01)
 - Projet info (~1000 lignes de code)
- Supports
 - Polycopié Cori-Levy
 - Transparents PC
(<http://w3.edu.polytechnique.fr/profs/informatique/Francois.Sillion/TC>)
 - Bibliothèque
- Délégués

Algorithmique et Programmation

- Structures de données
 - Listes, arbres, graphes, « tas »...
- Algorithmes fondamentaux
 - Tri, recherche
 - Parcours d'arbres, de graphes
- Programmation
 - Langage **Java**
 - Unix (ou machine personnelle)
 - Bibliothèque graphique **MacLib** (Philippe Chassignet)
- Utilisateur
 - Messagerie, WWW, etc.

Cursus d'informatique (1)

- Majeure *Algèbre, Informatique et Applications* (automne 2001)
 - 2 cours obligatoires :
 - Architecture du calcul
 - Systèmes formels, calculabilité et complexité
 - Imagerie tridimensionnelle
 - 1 cours à choisir parmi :
 - Bases de données
 - Langages de programmation
 - Algèbre, arithmétique
 - 1 Enseignement d'approfondissement parmi
 - Modélisation des réseaux de communication
 - Composants programmables
 - Géométrie et synthèse d'images
 - Automates finis
 - Modularité, objets et types

Cursus d'informatique (2)

- Majeure d 'Informatique (printemps 2002)
 - Initiation à C et ML
 - 2 cours obligatoires :
 - Conception et analyse d 'algorithmes
 - Principes et programmation des systèmes d 'exploitation
 - 1 cours parmi :
 - Compilation
 - Informatique distribuée
 - 1 cours parmi
 - Informatique parallèle
 - Composants programmables
 - Cryptologie
 - Images et vision
- DEA
 - Algorithmique
 - Sémantique, Preuves et Programmation
 - ...

Moi-même

- Directeur de recherche, INRIA
- iMAGIS, Laboratoire GRAVIR/IMAG,
<http://www-imagis.imag.fr>
- Thèmes de recherche (Image de Synthèse)
 - Simulation de l 'éclairage
 - Visualisation interactive de grands volumes de données
- Applications
 - Simulation architecturale
 - CAO, design
 - Effets spéciaux cinéma, jeux vidéo
 - Ondes radio
- Courriel: Francois.Sillion@imag.fr

Algorithmique

- Elements d'un problème informatique
 - Structure de données ?
 - Algorithme ?
 - Complexité ?
- Recherche d'un chemin dans le métro

www.sytadin.tm.fr



PC 1

François Sillion

7

Programmation: Java

- Langage récent (1994), Sun microsystems
- Principaux avantages
 - Typage fort
 - Gestion automatique de la mémoire
- Mais aussi
 - Orienté objet
 - Multitâche (*threads*)
 - Facilement distribué
 - Gestion des erreurs (*exceptions*)
- Execution :
 - Code intermédiaire : machine virtuelle
 - Compilation (très récent)
 - Sécurisation (Web)

PC 1

François Sillion

8

Le langage Java (1): variables

- Types et déclarations

```
boolean b = true; // ou false
char c = ' Z ';
int n = 18;
double pi = 31.415e-1;
int[] vecteur = {1,2,5,-4};
int[] vecteur = new int[5];
String msg = "Bonjour";
```

- Expressions

```
c
vecteur[0] = 2*pi - 0.1 // conversion
msg + " !..." // surcharge
new type[tailleEntière]
```

- Les types non scalaires sont manipulés par **référence**

Le Langage Java (2): instructions

- Simples

```
expression;
declaration;
```

- Composées

```
{ [ instr ... ] }
```

- Conditionnelle

```
if ( expr ) instr [else instr]
```

- Boucles

```
for ( expr1 ; expr2; expr3 ) instr
```

```
while ( expr ) instr
```

- Opérateurs

- Affectation: =
- Arithmétiques: + - * / %
- Comparaison: == (**attention !!**)
- Booléens: && || !
- Concaténation de chaînes: +

Le Langage Java (3): fonctions

- Fonctions

```
static typeRetour NomFonction( type1 var1,  
                             type2 var2,  
                             ... )  
  
{  
  declarations  
  instructions  
  return expression;  
}
```

- Procédures

= fonctions retournant `void` (rien)

Le Langage Java (4): Structure d'un programme

```
// Importation de modules externes  
import java.io.*  
  
public class program  
{  
  // constantes et variables globales  
  static final int max = 100;  
  
  // fonctions  
  static void FaitQuelqueChose( int p ... )  
  { . . . }  
  
  // point d'entrée  
  public static void main ( String[] arg )  
  {  
    ...  
  }  
}
```

Le Langage Java (5): Mise en œuvre

```
% javac program.java
% java program.class
```

- Exécute la fonction main()
- Autres classes nécessaires sont recherchées dans le chemin défini par la variable CLASSPATH.
- Quelques outils pratiques

```
void system.out.print ( String );
void system.out.println ( String );
double Math.random ();
long Math.round ( double );
int Math.min ( int, int );
...
```

Un programme ultra simple : Crible d'Eratosthène

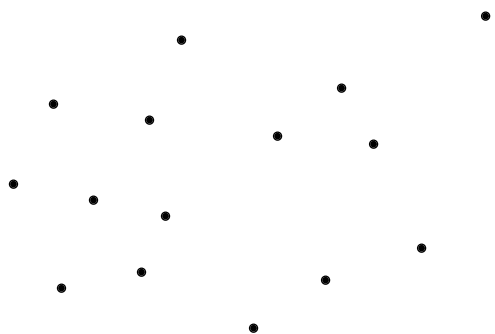
```
public class crible
{
    final int max = 100;
    boolean EstPremier[] = new boolean [max];

    static void rempli ( int p )
    {
        int k = 2;
        while ( k*p < max ) {
            EstPremier[k*p] = false;
            k++;
        }
    }

    static void Initialise()
    {
        for ( int i = 0 ; i < max ; ++i )
            EstPremier[i] = true;
    }

    public static void main ( String[] arg )
    {
        int i;
        Initialise();
        for ( i = 2 ; i < max ; ++i )
            rempli(i);
        for ( i = 1 ; i < max ; ++i )
            if ( EstPremier[i] )
                writeln(i + "est premier");
    }
}
```

Enveloppe convexe

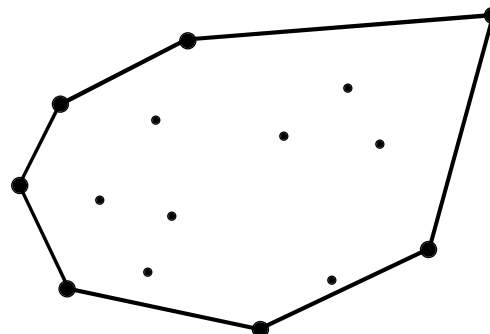


PC 1

François Sillion

15

Enveloppe convexe

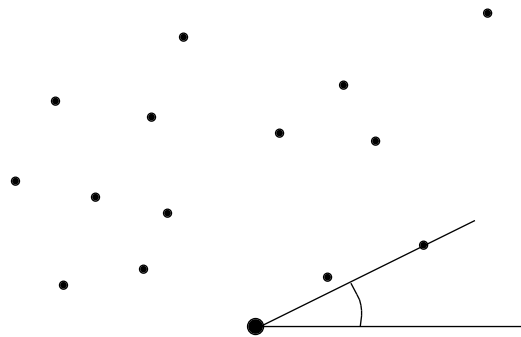


PC 1

François Sillion

16

Algorithme du cordeau

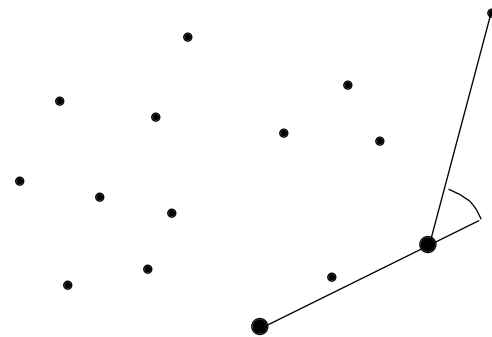


PC 1

François Sillion

17

Algorithme du cordeau



PC 1

François Sillion

18

Enveloppe convexe : Algorithme du cordeau

- Remarques

- La fonction Angle retourne une mesure de l'angle entre P[i]P[j] et l'horizontale.
Pas forcément l'angle lui-même...
- Échange de deux éléments du tableau :

```
static void Echange(int a, int b)
{
    Point p = P[a];
    P[a] = P[b];
    P[b] = p;
}
```

- Nouvel objet

```
class Point
{
    double x,y;
}
```

```
Point Pt = new Point();
Pt.x = 2.0;
Point[] P = new Point[5];
```

on verra qu'on peut définir des **constructeurs...**

Enveloppe convexe : Algorithme du cordeau

```
public class cordeau
{
    final static int max = 100;
    static Point[] P = new Point[max+1];

    static int wrap ( Point[] P, int max )
    {
        int min = TrouveOrdonnéeMinimum();
        // place une sentinelle
        P[max] = P[min];
        double angle = 0;
        for ( j=0 ; j < max ; ++j )
        {
            Echange(j,min);
            min = max; am = angle; angle = 360;
            for ( int k=j+1; k <= max; ++k )
                if ( Angle( P[j], P[k] ) > am )
                    if ( Angle ( P[j], P[k] ) < angle )
                        {
                            min = k; angle = Angle(P[j],P[k]);
                        }
                if ( min == max ) return j;
        }
    }
}

public static void main ( String[] arg )
{
    Initialise();
    int Nb = wrap ( P , max );
}
}
```

Notions de complexité

- Complexité des opérations demandées par un algorithme en fonction de la taille des données d'entrée
- Temps ou espace
- Comportement asymptotique
- Cas le pire, comportement en moyenne
- Notations
 - $O(n)$, $o(n \log(n))$...
- Complexité dépendant de la sortie

Complexité de l'algorithme du cordeau ?

- $O(n^2)$ dans le cas le pire (tous les points sur l'enveloppe convexe)
- $O(n p)$ si p est le nombre de points sur l'enveloppe
- Peut-on faire mieux ?

