# Algorithmique - Évaluation
# Journées de formation

Groupe Algorithmique de l'IREM de Clermont-Ferrand

26-27-28 mars 2012

L'enseignement de l'algorithmique en arrive à sa troisième année. Les premiers élèves ayant bénéficié de cet enseignement passent le baccalauréat dès cette année, et la question de l'évaluation se fait de plus en plus pressante.

Plusieurs questions se posent :

- Que doit-on évaluer ?

Plusieurs questions se posent :

- Que doit-on évaluer ?
- Comment l'évaluer ?

Plusieurs questions se posent :

- Que doit-on évaluer ?
- Comment l'évaluer ?
- Comment rendre cette évaluation constructive pour l'élève ?

Plusieurs capacités se dégagent :

- lire un algorithme ;
- exécuter un algorithme ;
- comprendre un algorithme ;
- écrire un algorithme.

# Lire un algorithme

Cette compétence suppose de savoir reconnaître un algorithme.

# Lire un algorithme

Cette compétence suppose de savoir reconnaître un algorithme.

Question : qu'est-ce qui n'est pas un algorithme ?

# Lire un algorithme

Cette compétence suppose de savoir reconnaître un algorithme.

Question : qu'est-ce qui n'est pas un algorithme ?

Exemple : une recette de cuisine.

# Lire un algorithme

Cette compétence suppose de savoir reconnaître un algorithme.
Question : qu'est-ce qui n'est pas un algorithme ?
Exemple : une recette de cuisine.
Pourquoi ?
Présence d'implicites, instructions ambigües, excessivement réducteur...

# Lire un algorithme

Cette compétence suppose de savoir reconnaître un algorithme.
Question : qu'est-ce qui n'est pas un algorithme ?
Exemple : une recette de cuisine.
Pourquoi ?
Présence d'implicites, instructions ambigües, excessivement réducteur...
Et résultat non garanti !

# Lire un algorithme

Un exemple d'algorithme :

# Lire un algorithme

Un exemple d'algorithme :

Je pars de 0,

# Lire un algorithme

Un exemple d'algorithme :

Je pars de 0,
j'ajoute 1 au résultat précédent,

# Lire un algorithme

Un exemple d'algorithme :

Je pars de 0,
j'ajoute 1 au résultat précédent,
puis j'ajoute 2,

# Lire un algorithme

Un exemple d'algorithme :

Je pars de 0,
j'ajoute 1 au résultat précédent,
puis j'ajoute 2,
je continue ainsi jusqu'à ce que j'ajoute 20.

# Lire un algorithme

Un exemple d'algorithme :

Je pars de 0,
j'ajoute 1 au résultat précédent,
puis j'ajoute 2,
je continue ainsi jusqu'à ce que j'ajoute 20.
J'annonce le dernier résultat obtenu.

# Lire un algorithme

Mais on a plus souvent l'habitude de le rencontrer sous cette forme :

$S \leftarrow 0$
**pour** $k$ **allant de** $1$ **à** $20$ **faire**
    $S \leftarrow S + k$
retourner $S$

# Lire un algorithme

Mais on a plus souvent l'habitude de le rencontrer sous cette forme :

$S \leftarrow 0$
**pour** $k$ **allant de** $1$ **à** $20$ **faire**
    $S \leftarrow S + k$
retourner $S$

Ces conventions vont faciliter l'écriture (et guider la réflexion) dans des situations plus complexes.

# Exécuter un algorithme

But :

- savoir exécuter un algorithme à la main

# Exécuter un algorithme

But :

- savoir exécuter un algorithme à la main
- retranscrire cette exécution sur une copie

# Exécuter un algorithme

But :

- savoir exécuter un algorithme à la main
- retranscrire cette exécution sur une copie

En cours de formation, vérification éventuelle via programmation !

# Exécuter un algorithme

## Exemple

**Entrées** : $x$ un entier
**Sorties** : $y$ un reéel
**si** $x \geq 0$ **alors**
    $y \leftarrow x$
**sinon**
    $y \leftarrow -x$
retourner $y$

Exécuter cet algorithme pour $x = 3$ :

# Exécuter un algorithme

## Exemple

**Entrées** : $x$ un entier
**Sorties** : $y$ un reéel
**si** $x \geq 0$ **alors**
    $y \leftarrow x$
**sinon**
    $y \leftarrow -x$
retourner $y$

Exécuter cet algorithme pour $x = 3$ :

| **Copie 1** | **Copie 2** |
|---|---|
| Si $3 \geq 0$ alors | $3 \geq 0$ |
| $y = 3$ | $y = 3$ |
| sinon $y = -3$ | Retourner 3 |
| Retourner 3 | |

# Exécuter un algorithme

## Exemple

**Entrées** : $x$ un entier
**Sorties** : $y$ un reéel
**si** $x \geq 0$ **alors**
    $y \leftarrow x$
**sinon**
    $y \leftarrow -x$
retourner $y$

Exécuter cet algorithme pour $x = 3$ :

| **Copie 1** | **Copie 2** |
|---|---|
| Si $3 \geq 0$ alors | $3 \geq 0$ |
| $y = 3$ | $y = 3$ |
| sinon $y = -3$ | Retourner 3 |
| Retourner 3 | |

On peut supposer que l'élève 1 n'a pas compris la structure conditionnelle : exécuter un algorithme est intimement lié à la compréhension de l'algorithme.

# Exécuter un algorithme

Ce qui en fait un exercice incontournable, même en fin de formation surtout sur des algorithmes plus complexes.

# Exécuter un algorithme

Ce qui en fait un exercice incontournable, même en fin de formation surtout sur des algorithmes plus complexes.
Une utilisation à l'écrit en temps limité peut être la suivante :

# Exécuter un algorithme

## Exemple

On définit deux suites $(a_n)$ et $(b_n)$ par $a_0 = 1$, $b_0 = 7$ et

$$\begin{cases} a_{n+1} = \dfrac{1}{3}(2a_n + b_n) \\ b_{n+1} = \dfrac{1}{3}(a_n + 2b_n) \end{cases}$$

Parmi les algorithmes suivants lequel permet de calculer les termes $a_n$ et $b_n$ d'un rang $n$ donné ?

# Exécuter un algorithme

## Exemple

`Algorithme 1`

**Entrées** : $n$ un entier
**Sorties** : $a$, $b$ deux réels
$a \leftarrow 1$
$b \leftarrow 7$
$k \leftarrow 0$
**tant que** $k < n$ **faire**
$a \leftarrow \dfrac{2a+b}{3}$
$b \leftarrow \dfrac{a+2b}{3}$
$k \leftarrow k+1$

Retourner $a$ et $b$

`Algorithme 2`

**Entrées** : $n$ un entier
**Sorties** : $a$, $b$ deux réels
$a \leftarrow 1$
$b \leftarrow 7$
$k \leftarrow 0$
**tant que** $k < n$ **faire**
$a \leftarrow \dfrac{2a+b}{3}$
$temp \leftarrow a$
$b \leftarrow \dfrac{temp+2b}{3}$
$k \leftarrow k+1$

Retourner $a$ et $b$

`Algorithme 3`

**Entrées** : $n$ un entier
**Sorties** : $a$, $b$ deux réels
$a \leftarrow 1$
$b \leftarrow 7$
$k \leftarrow 0$
**tant que** $k < n$ **faire**
$temp \leftarrow a$
$a \leftarrow \dfrac{2temp+b}{3}$
$b \leftarrow \dfrac{temp+2b}{3}$
$k \leftarrow k+1$

Retourner $a$ et $b$

# Exécuter un algorithme

Le passage par l'exécution pour calculer $a_1$ et $b_1$ est nécessaire pour déterminer le bon algorithme.

# Exécuter un algorithme

## Sujet Bac TES Juin 2011

(Commande) B(x) :=10*((1+ln(x))/x)

(Rép 1) $x- \quad > \quad 10 \quad * \left(\frac{1+\ln x}{x}\right)$

(Commande) dériver(B(x),x)

(Rép 2) $\frac{10}{x^2} \quad + \quad \frac{10*(1+\ln(x))*(-1)}{x^2}$

(Commande) résoudre(B(x)=0,x)

(Rép 3) $[\exp(-1)]$

(Commande) résoudre(B(x)>0,x)

(Rép 4) $[x > \exp(-1)]$

(Commande) maximum(B(x),[0.1 ;10])

(Rép 5) 10

Utiliser divers logiciels permet de se préparer à ce genre de questions :

1. Traduire sur le graphique donné en annexe, illustrant la courbe représentative de la fonction $B$, les réponses 3, 4 et 5 renvoyées par le logiciel de calcul formel.
2. Justifier la réponse 3 renvoyée par le logiciel de calcul formel. Interpréter cette valeur en terme de résultat mensuel pour l'entreprise.

# Que fait cet algorithme ?

## Algorithme mystère

Lire des entiers positifs $a$ et $b$
Donner à *res* la valeur 1
**tant que** $b > 0$ **faire**
    **si** *b impair* **alors**
        Donner à *res* la valeur $res \times a$
        Donner à $b$ la valeur $(b-1)/2$
    **sinon**
        Donner à $b$ la valeur $b/2$
    Donner à $a$ la valeur $a \times a$
Retourner *res*

# Que fait cet algorithme ?

## De quoi parle-t-on ?

- On peut exécuter un algorithme sans le comprendre : il est conçu dans ce but
- Plusieurs niveaux de compréhension
  - Comprendre le mot-à-mot d'un algorithme
  - Comprendre le but d'un algorithme
  - Comprendre le fonctionnement d'un algorithme
  - Comprendre l'intérêt d'un algorithme

# Que fait cet algorithme ?

## Algorithme mystère

Lire des entiers positifs $a$ et $b$
Donner à *res* la valeur 1
**tant que** $b > 0$ **faire**
    **si** *b impair* **alors**
        Donner à *res* la valeur $res \times a$
        Donner à $b$ la valeur $(b-1)/2$
    **sinon**
        Donner à $b$ la valeur $b/2$
    Donner à $a$ la valeur $a \times a$
Retourner *res*

- Mot-à-Mot

# Que fait cet algorithme ?

## Algorithme mystère

Lire des entiers positifs $a$ et $b$
Donner à *res* la valeur 1
**tant que** $b > 0$ **faire**
  **si** *b impair* **alors**
    Donner à *res* la valeur $res \times a$
    Donner à $b$ la valeur $(b-1)/2$
  **sinon**
    Donner à $b$ la valeur $b/2$
  Donner à $a$ la valeur $a \times a$
Retourner *res*

- Mot-à-Mot
  - L'algorithme retourne bien un résultat (terminaison)

# Que fait cet algorithme ?

## Algorithme mystère

Lire des entiers positifs $a$ et $b$
Donner à *res* la valeur 1
**tant que** $b > 0$ **faire**
    **si** *b impair* **alors**
        Donner à *res* la valeur $res \times a$
        Donner à $b$ la valeur $(b-1)/2$
    **sinon**
        Donner à $b$ la valeur $b/2$
    Donner à $a$ la valeur $a \times a$
Retourner *res*

- But ?

# Que fait cet algorithme ?

## Algorithme mystère

Lire des entiers positifs $a$ et $b$
Donner à *res* la valeur 1
**tant que** $b > 0$ **faire**
    **si** *b impair* **alors**
        Donner à *res* la valeur $res \times a$
        Donner à $b$ la valeur $(b-1)/2$
    **sinon**
        Donner à $b$ la valeur $b/2$
    Donner à $a$ la valeur $a \times a$
Retourner *res*

- But ?
  - Contexte
  - Tests

# Que fait cet algorithme ?

## Algorithme mystère

Lire des entiers positifs $a$ et $b$
Donner à *res* la valeur 1
**tant que** $b > 0$ **faire**
    **si** *b impair* **alors**
        Donner à *res* la valeur $res \times a$
        Donner à $b$ la valeur $(b-1)/2$
    **sinon**
        Donner à $b$ la valeur $b/2$
    Donner à $a$ la valeur $a \times a$
Retourner *res*

- Fonctionnement ?

# Que fait cet algorithme ?

## Algorithme mystère

Lire des entiers positifs $a$ et $b$
Donner à *res* la valeur 1
**tant que** $b > 0$ **faire**
  **si** *b impair* **alors**
    Donner à *res* la valeur $res \times a$
    Donner à $b$ la valeur $(b-1)/2$
  **sinon**
    Donner à $b$ la valeur $b/2$
  Donner à $a$ la valeur $a \times a$
Retourner *res*

- Fonctionnement ?
  - Maths
  - $x^{2^{i+1}} = \left(x^{2^i}\right)^2$
  - $5^{13} = 5^{2^0} \times 5^{2^2} \times 5^{2^3}$

# Que fait cet algorithme ?

## Algorithme mystère

Lire des entiers positifs $a$ et $b$
Donner à *res* la valeur 1
**tant que** $b > 0$ **faire**
  **si** *b impair* **alors**
    Donner à *res* la valeur $res \times a$
    Donner à $b$ la valeur $(b-1)/2$
  **sinon**
    Donner à $b$ la valeur $b/2$
  Donner à $a$ la valeur $a \times a$
Retourner *res*

- Utilité ?

# Que fait cet algorithme ?

## Algorithme mystère

Lire des entiers positifs $a$ et $b$
Donner à *res* la valeur 1
**tant que** $b > 0$ **faire**
  **si** *b impair* **alors**
    Donner à *res* la valeur $res \times a$
    Donner à $b$ la valeur $(b-1)/2$
  **sinon**
    Donner à $b$ la valeur $b/2$
  Donner à $a$ la valeur $a \times a$
Retourner *res*

- Utilité ?
  - Nombre de tours dans la boucle
  - Efficacité : $b$ vs. $\log b$

# Que fait cet algorithme ?

## Question piège

- Quel est le contexte ?
- Quel est le niveau de réponse attendu ?

## Question difficile

- Nécessite un recul critique
- Facilité par des questions ciblées

# Comprendre un algorithme

## Questions portant sur la correction

- « Pourquoi cet algorithme est-il faux ? »
- Souvent plus pertinent que « Pourquoi cet algorithme est-il correct ? »

# Comprendre un algorithme

## Chercher l'intrus

- *Contexte : établir, puis démontrer, une conjecture sur la parité du nombre de diviseurs d'un entier.*
- On se propose de mettre en œuvre un algorithme qui, pour un entier naturel $n$, non nul, choisi par l'utilisateur, affiche le nombre de diviseurs de $n$.
- On donne quatre versions :

# Comprendre un algorithme

## Chercher l'intrus

lire $n$
$c \leftarrow 0$
**pour** $k$ **allant de** 1 **à** $n$ **faire**
    | **si** $k$ *divise* $n$ **alors** $c \leftarrow c + 1$
afficher $c$

lire $n$
$c \leftarrow 2$
**pour** $k$ **allant de** 2 **à** $n - 1$ **faire**
    | **si** $k$ *divise* $n$ **alors** $c \leftarrow c + 1$
afficher $c$

lire $n$
$c \leftarrow 1$
**pour** $k$ **allant de** 2 **à** $n$ **faire**
    | **si** $k$ *divise* $n$ **alors** $c \leftarrow c + 1$
afficher $c$

lire $n$
$c \leftarrow 1$
**pour** $k$ **allant de** 1 **à** $n - 1$ **faire**
    | **si** $k$ *divise* $n$ **alors** $c \leftarrow c + 1$
afficher $c$

Parmi ces quatre versions, une ne répond pas au problème.
Laquelle et pourquoi ?

# Comprendre un algorithme

## Questions portant sur la terminaison

- « L'exécution de cet algorithme se termine-t-elle ? »
- « Quel argument assure que cet algorithme rend bien un résultat ? »
- Lien avec des notions mathématiques

# Comprendre un algorithme

## Convergence et terminaison

Soit $(u_n)$ la suite définie par

$$\begin{cases} u_0 &= 1 \\ u_{n+1} &= u_n - \ln\left(u_n^2 + 1\right) \text{ pour } n \in \mathbb{N}. \end{cases}$$

1. Démontrer par récurrence que, pour $n \geqslant 0$, $u_n \in [0\ ;\ 1]$.
2. Étudier le sens de variation de la suite $(u_n)$.
3. Démontrer que la suite $(u_n)$ converge vers 0.

# Comprendre un algorithme

## Convergence et terminaison

Soit $(u_n)$ la suite définie par $\begin{cases} u_0 &= 1 \\ u_{n+1} &= u_n - \ln\left(u_n^2 + 1\right) \text{ pour } n \in \mathbb{N}. \end{cases}$

4. On propose l'algorithme suivant :

```
u ← 1
k ← 0
tant que u > 0,05 faire
    | k ← k + 1
    | u ← u − ln(u² + 1)
retourner k
```

1. Que retourne cet algorithme à la fin de son exécution ?
2. Quel résultat mathématique, démontré précédemment, nous permet d'affirmer que l'exécution de cet algorithme se termine.

# Comprendre un algorithme

## Questions portant sur l'efficacité

- Des algorithmes effectuent la même tâche, pas de la même façon
- Comparaisons selon divers critères :
  - nécessité d'un test donné
  - type de boucle choisie
  - nombre d'instructions exécutées

# Comprendre un algorithme

## Utile ou pas ?

- *Contexte : calcul du nombre de triplets* $(a, b, c) \in \{0, \ldots, 9\}^3$ *tels que* $a \leq b \leq c$

```
s ← 0
pour a allant de 0 à 9 faire
    pour b allant de a à 9 faire
        pour c allant de b à 9 faire
            si a ≤ b ≤ c alors
                s ← s + 1
retourner s
```

Peut-on simplifier cet algorithme ?

# Comprendre un algorithme

## Réutiliser ou pas ?

- *Contexte : Compter le nombre de nombres premiers*
- On a déjà écrit une fonction `ComptePremier(`$n$`)` qui renvoie le nombre de premiers inférieurs ou égaux à $n$
- On se prépare à écrire un algorithme permettant de compter le nombre de premiers dans un intervalle $]a; b]$
- Questions orales :
  - Vaut-il mieux utiliser la fonction `ComptePremier(`$n$`)` déjà écrite ou réécrire entièrement une nouvelle fonction ?
  - Critère de choix ?

# Comprendre un algorithme

## On aborde les questions sérieuses

- Le cœur de l'algorithmique c'est concevoir des algorithmes
- Comprendre un algorithme pré-existant : on évalue une capacité analytique, pas encore créative
- Étape préalable au passage à l'écriture d'un algorithme
- Demander de compléter, modifier ou corriger un algorithme sont des modalités d'évaluation intermédiaires

# Comprendre un algorithme

## Algorithme à trous

$\rightsquigarrow$ S'assurer de la compréhension d'un point précis

*Compléter l'algorithme suivant afin qu'il retourne la somme des entiers naturels de 0 à $n$, pour $n$ donné*

lire $n$
$s \leftarrow 0$
**pour** $k$ **allant de** 0 **à** $n$ **faire**
    $s \leftarrow$.....
retourner $s$

On met l'accent sur le rôle d'une variable d'accumulation

# Comprendre un algorithme

## Variante d'un algorithme

*Modifier l'algorithme suivant afin qu'il retourne la somme des entiers naturels de 0 à n, pour n donné*

lire $n$
$s \leftarrow 0$
**pour** $k$ **allant de** 0 **à** $n$ **faire**
    $s \leftarrow s + 1$
retourner $s$

# Comprendre un algorithme

## Correction d'un algorithme

*Corriger l'algorithme suivant afin qu'il retourne la somme des entiers naturels de 0 à n, pour n donné*

lire $n$
$s \leftarrow 0$
$k \leftarrow 0$
**tant que** $k \leq n$ **faire**
    $s \leftarrow s + k$
retourner $s$

- Difficulté supplémentaire : trouver l'erreur
- On peut choisir des erreurs souvent relevées

# Écrire un algorithme

Les attendus tels qu'indiqués dans les programmes :

Instructions élémentaires
Dans le cadre de résolutions de problèmes.

- Écrire une formule permettant un calcul
- Écrire un programme calculant et donnant la valeur d'une fonction
- Écrire les instructions d'entrées et sorties nécessaires

# Écrire un algorithme

Les attendus tels qu'indiqués dans les programmes :

Boucle, itérateur et instruction conditionnelle
Dans le cadre de résolutions de problèmes.

- Programmer un calcul itératif, le nombre d'itérations étant donné
- Programmer une instruction conditionnelle
- Programmer une boucle avec fin de boucle conditionnelle

# Écrire un algorithme

**Travail en temps limité.**

Quelques propositions intégrant la contrainte de temps :

- écrire une variante d'un algorithme donné
- écrire un algorithme *standard*
- compléter un algorithme donné
- réécrire un algorithme donné en imposant certains critères

Dans ce cadre, les questions de programmation sont à manier avec précaution.

# Écrire un algorithme

*Écrire un algorithme retournant le plus petit entier $n$ tel que $u_n > 50$ avec $(u_n)$ la suite définie par $u_0 = 1$ et $u_{n+1} = \frac{3}{2}u_n + \frac{1}{3}$.*
un élève répond :

**Variables**
$n$ est du type nombre
$u$ est du type nombre
$F$ est du type nombre
**Début algorithme**
lire $F$ # `il faut choisir F très grand pour être sûre`
$n$ prend la valeur 0
$u$ prend la valeur 1
**pour** $k$ **allant de** *1* **à** $F$ **faire**
  $u \leftarrow \frac{3}{2}u + \frac{1}{3}$
  **si** $u \leq 50$ **alors** $n$ prend la valeur $n + 1$
Afficher $n$

le choix de la structure itérative la plus adaptée n'a pas été évidente pour l'élève.

# Écrire un algorithme

*Écrire un algorithme de résolution d'une équation du second degré à partir de ses coefficients donnés.*

comment réagir face à cette réponse :

#$a, b, c$ et $\Delta$ désignent des réels
Lire $a$, $b$ et $c$
$\Delta$ reçoit $b^2 - 4ac$
**si** $\Delta < 0$ **alors**
  afficher « l'équation n'admet pas de solution réelle »
**si** $\Delta = 0$ **alors**
  $x_0$ reçoit $-\frac{b}{2a}$
  afficher « l'équation admet une solution : »
  afficher $x_0$
**si** $\Delta > 0$ **alors**
  $x_1$ reçoit $\frac{-b-\sqrt{\Delta}}{2a}$
  $x_2$ reçoit $\frac{-b+\sqrt{\Delta}}{2a}$
  afficher « l'équation admet deux solutions : »
  afficher $x_1$ et $x_2$

# Écrire un algorithme

**Travail en temps libre.**

L'éventail des possibilités est ici beaucoup plus vaste :

- résoudre un problème ouvert (Convergences)
- modéliser, simuler (Lois de probabilités)
- découvrir (Vérification d'un numéro INSEE par sa clé)
- etc.

De plus, la programmation et l'expérimentation peuvent trouver toute leur place.

# Écrire un algorithme

Il est possible de valoriser la prise d'initiative en indiquant clairement à l'élève que les avancées dans son travail seront prises en compte, par la formule rituelle :
*Toute trace de recherche...*

Partant du constat qu'une évaluation est toujours liée à des pratiques d'enseignement, la question du devenir de ce genre de question dans le cadre d'une évaluation terminale au lycée dans quelques années va se poser.

- La mise en place des notions précédentes ne peut trouver un sens qu'au travers d'exercices avec une plus value dans la démarche mathématique.
- Il est intéressant de se demander vers quoi orienter cette évaluation ?

## Maintenant

En suivant l'idée de favoriser la prise d'initative des élèves, comme le suggèrent les questions du type

« *Toute trace de recherche sera prise en compte dans l'évaluation . . .* »

## Maintenant

En suivant l'idée de favoriser la prise d'initative des élèves, comme le suggèrent les questions du type

« *Toute trace de recherche sera prise en compte dans l'évaluation . . .* »

## Dans un futur proche ?

On peut alors imaginer possible ce genre de question ouverte au baccalauréat :

« *Réaliser une recherche exhaustive de tous les couples d'entiers naturels* $(x; y)$ *solutions de l'inéquation :* $5x^2 - 4xy + y^2 \leq 100$ »

## Quelques remarques

- En remplacaçant 100 par des valeurs plus petites, ce genre de question peut être traité par les mêmes méthodes, qui vont être mises en œuvre au travers de l'algorithmique, et cela sans avoir recours à une machine.
- Le cas de figure précédent n'apparaît pas souvent dans les évaluations que l'on donne à nos élèves.

# Un exemple de solution

Pour tous les réels $x$ et $y$, on a :

$$\begin{aligned} 5x^2 - 4xy + y^2 &= x^2 + 4x^2 - 4xy + y^2 \\ &= x^2 + (2x - y)^2 \end{aligned}$$

On constate que :
si $x^2 > 100$ ou $(2x - y)^2 > 100$ alors l'inéquation ne peut être vérifiée.
La première inéquation impose d'avoir

$$-10 \leq x \leq 10.$$

La seconde inéquation impose d'avoir

$$-10 \leq 2x - y \leq 10.$$

ce dernier résultat donne $y \leq 2x + 10$, puis $y \leq 30$.

# Un exemple de solution

- Nous avons ainsi trouvé un majorant des solutions.

# Un exemple de solution

- Nous avons ainsi trouvé un majorant des solutions.
- On peut alors établir la liste des solutions à partir d'un algorithme exhaustif.

# Un exemple de solution

- Nous avons ainsi trouvé un majorant des solutions.
- On peut alors établir la liste des solutions à partir d'un algorithme exhaustif.

**pour** $x$ **allant de** 0 **à** 10 **faire**
  **pour** $y$ **allant de** 0 **à** 30 **faire**
    **si** $5x^2 - 4xy + y^2 \leq 100$ **alors** afficher $(x; y)$

## Et un peu plus tard...

Déterminer, et mettre en œuvre, une méthode de résolution qui, pour un entier naturel $n$ donné, retourne la liste des couples d'entiers $(x; y)$ solutions de l'inéquation :

$$5x^2 - 4xy + y^2 \leq n$$

## Qu'est-ce que l'on évalue ?

- prise d'initiative de l'élève
- calcul algébrique, factorisation
- recherche d'un majorant
- écriture d'un algorithme de recherche exhaustive
  *en tenant compte des points d'algorithmique soulevés dans les parties précédentes*

# Écrire un algorithme

La résolution de ce type de question pourra alors reposer sur la connaissance par les élèves d'exemples d'algorithmes standards comme :

- Travaux autour des suites numériques
- Recherche de l'existence d'une solution de $f(x) = k$ et encadrement
- Calcul d'une valeur approchée d'une intégrale avec $f$ strictement monotone
- Simulation d'une marche aléatoire
- Mise en œuvre d'une recherche exhaustive