Cours d'Eléments d'Algorithmique

1 cours magistral le jeudi de 14h à 15h30

Cours d'Eléments d'Algorithmique

La prof d'amphi: Anne Micheli

anne.micheli@irif.fr

https://www.irif.fr/∼amicheli/

Cours d'Eléments d'Algorithmique

Les chargés de TDs

Juliusz Chroboczek	Info 1	mardi 8h30-10h30
Antonio Bucciarelli	Info 2	mardi 16h45-18h45
Victor Lanvin	Info 3	vendredi 10h45-12h45
Ilias Garnier	Info 4	mercredi 8h30-10h30

Cours d'Eléments d'Algorithmique

40% de contrôle continu + 60% pour l'examen final

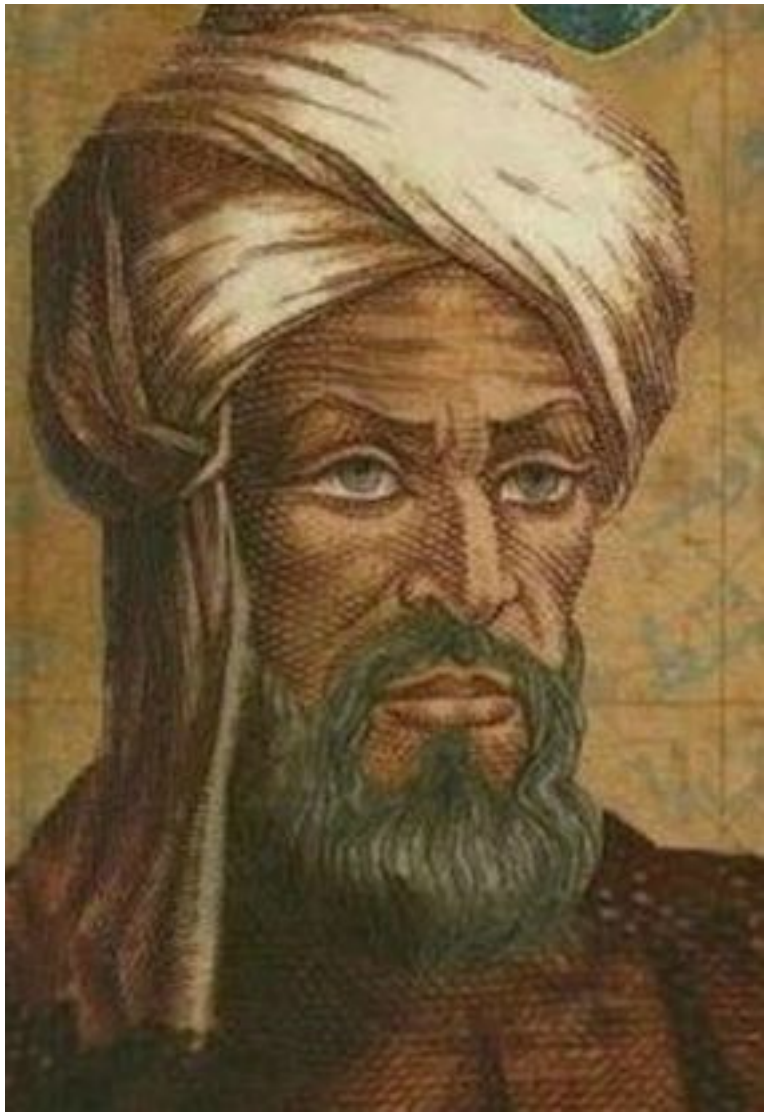
Partiel (le 18 octobre)

D'où vient le mot algorithme?

D'où vient le mot algorithme?

Muhammad Ibn Mûsâ al-Khuwârizmî ($\sim$ 780-850)

mathématicien, géographe, astrologue et astronome perse



Qu'est ce qu'un algorithme?

Qu'est ce qu'un algorithme?

Au départ, il y a un problème ? ? ?

Qu'est ce qu'un algorithme?

Au départ, il y a un problème ? ? ?

Comment faire un
gâteau au chocolat?

Qu'est ce qu'un algorithme?

Au départ, il y a un problème ? ? ?

Comment faire un gâteau au chocolat?

Comment trier dans l'ordre croissant une suite de nombres entiers?

Qu'est ce qu'un algorithme?

Au départ, il y a un problème ? ? ?

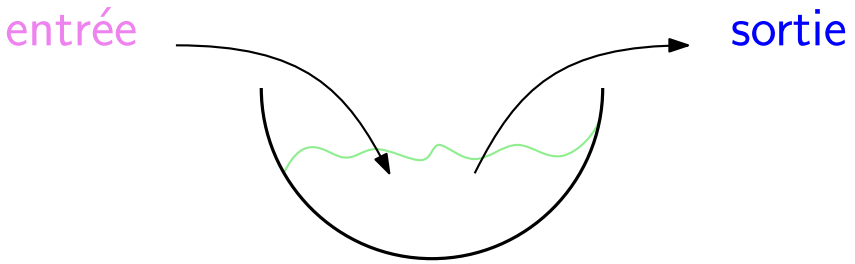
Comment trier dans l'ordre croissant une suite de nombres entiers?

Comment faire un gâteau au chocolat?

Comment additionner 2 nombres?

Qu'est ce qu'un algorithme?

Suite finie d'instructions qui transforment une donnée d'entrée en une donnée de sortie



Qu'est ce qu'un algorithme?

Pour 6 personnes:

200 g de chocolat noir

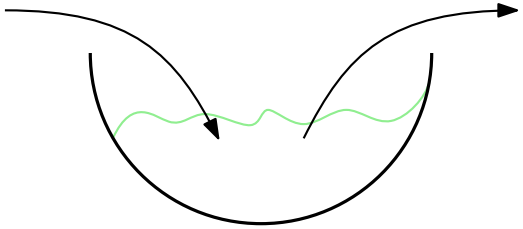
4 œufs

150 g de sucre

80 g de farine

200 g de beurre

entrée



sortie

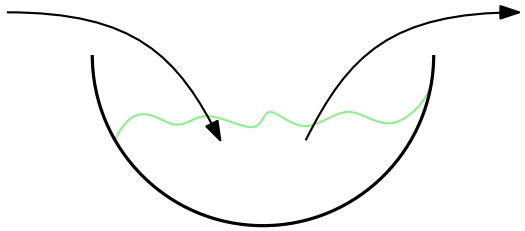
mélanger les jaunes d'œufs avec
...
mettre 20 min au four à 180 °

suite finie d'instructions

Qu'est ce qu'un algorithme?

$n_1 = 50839$
$n_2 = 160736$

entrée



res $= 211575$

sortie

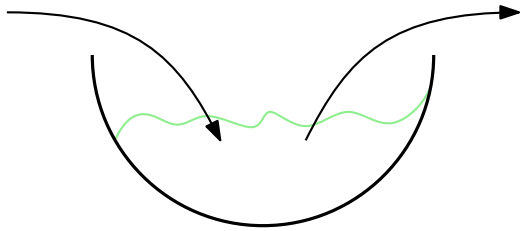
commencer par additionner les
2 chiffres des unités

suite finie d'instructions

Qu'est ce qu'un algorithme?

$n_1 = 50839$
$n_2 = 160736$

entrée



res $= 211575$

sortie

commencer par additionner les
2 chiffres des unités
puis répéter :
    noter le chiffre des unités
    retenir le chiffre des dizaines,
        s'il y en a un
    additionner les deux chiffres
        suivants de n_1 et n_2 et la
        retenue

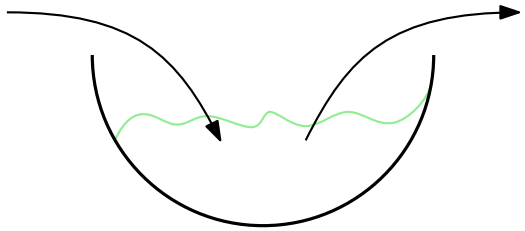
suite finie d'instructions

Qu'est ce qu'un algorithme?

La donnée d'entrée est appelée **instance** du problème

$n_1 = 50839$
$n_2 = 160736$

entrée



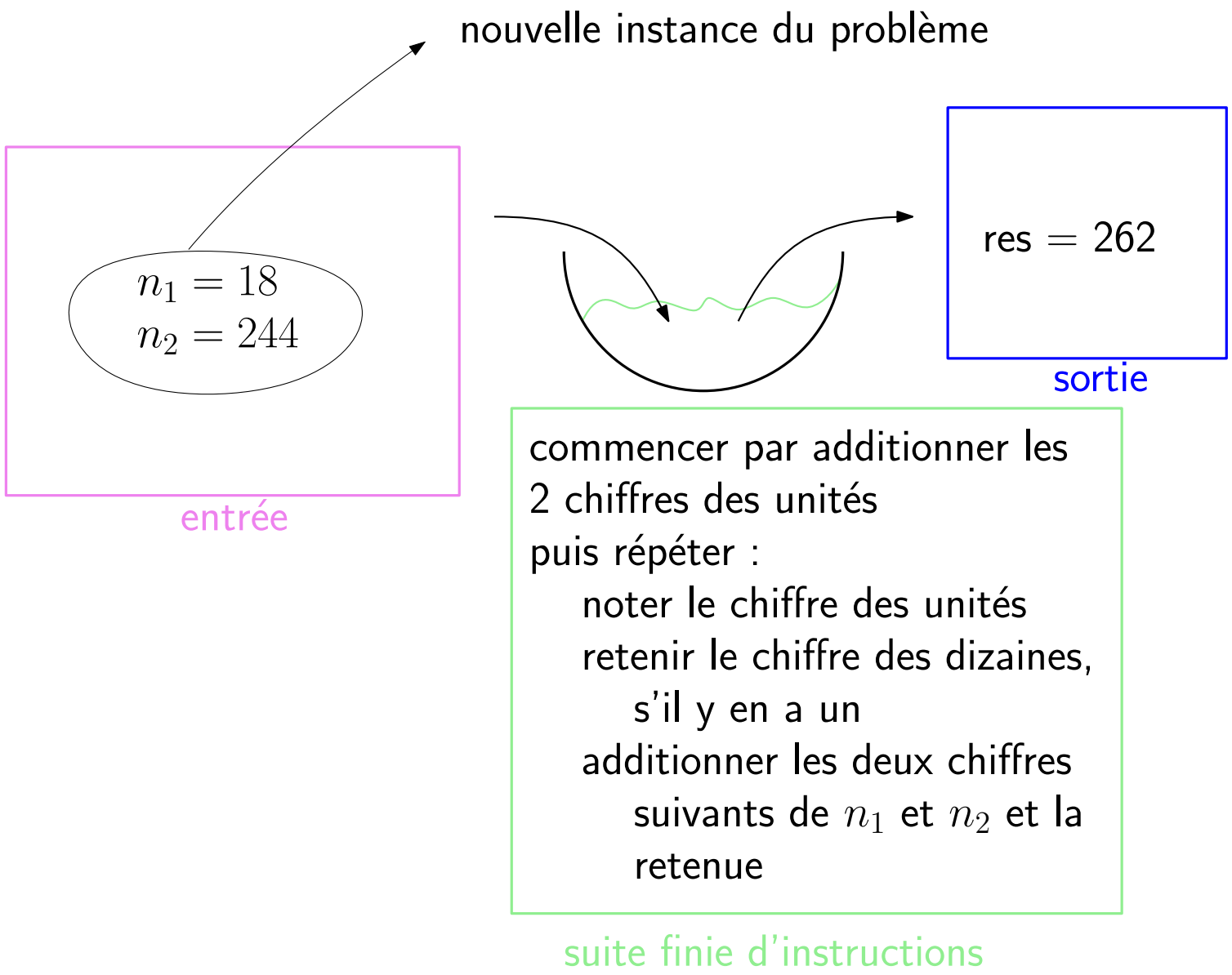
res $= 211575$

sortie

commencer par additionner les
2 chiffres des unités
puis répéter :
    noter le chiffre des unités
    retenir le chiffre des dizaines,
        s'il y en a un
    additionner les deux chiffres
        suivants de n_1 et n_2 et la
        retenue

suite finie d'instructions

Qu'est ce qu'un algorithme?



Qu'est ce qu'un algorithme?

Un algorithme est dit **correct** si, pour **toute instance** du problème, l'algorithme s'arrête lorsqu'il produit la bonne donnée de sortie.

Qu'est ce qu'un algorithme?

Un algorithme est dit **correct** si, pour **toute instance** du problème, l'algorithme s'arrête lorsqu'il produit la bonne donnée de sortie.

Exemples de problèmes résolus par l'algorithmique:

- Accéder rapidement à l'information sur internet
- Identifier les 100000 gènes de l'ADN humaine
- Archiver des données telles qu'elles occupent le moins de place possible

Écrire un algorithme

Il n'est pas nécessaire de choisir un langage de programmation

Écrire un algorithme

Il n'est pas nécessaire de choisir un langage de programmation

$\longrightarrow$ on ne veut pas dépendre des spécificités d'un langage

$\Longrightarrow$ on écrit en **pseudo-code**

Écrire un algorithme

Il n'est pas nécessaire de choisir un langage de programmation

$\longrightarrow$ on ne veut pas dépendre des spécificités d'un langage

$\Longrightarrow$ on écrit en **pseudo-code**

<u>Affectation</u>

$x \leftarrow$ ... ou $x =$ ...

<u>Tests</u>

$=$ ou $==$

$\neq, >, \geq, \ldots$

<u>Conditionnelle</u>

if ... **then{**
  ...
} else {
  ...
}

<u>Fonction</u>

mafonction(...){
  ...
  return ...
}

<u>Boucles</u>

for i **from** ... **to** ... **{**
  ...
}

while ... **{**
  ...
}

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: on initialise le minimum à la première entrée du tableau.

Puis on parcourt les autres entrées du tableau une à une

en mettant à jour le minimum si nécessaire.

6	19	3	7	11	9	2	15	5

min $\leftarrow 6$

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: on initialise le minimum à la première entrée du tableau.

Puis on parcourt les autres entrées du tableau une à une

en mettant à jour le minimum si nécessaire.

6	19	3	7	11	9	2	15	5

min $\leftarrow 6$

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: on initialise le minimum à la première entrée du tableau.

Puis on parcourt les autres entrées du tableau une à une

en mettant à jour le minimum si nécessaire.

6	19	3	7	11	9	2	15	5

min $\leftarrow 3$

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: on initialise le minimum à la première entrée du tableau.

Puis on parcourt les autres entrées du tableau une à une

en mettant à jour le minimum si nécessaire.

6	19	3	7	11	9	2	15	5

min $\leftarrow 3$

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: on initialise le minimum à la première entrée du tableau.

Puis on parcourt les autres entrées du tableau une à une

en mettant à jour le minimum si nécessaire.

6	19	3	7	11	9	2	15	5

min $\leftarrow 3$

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: on initialise le minimum à la première entrée du tableau.

Puis on parcourt les autres entrées du tableau une à une

en mettant à jour le minimum si nécessaire.

6	19	3	7	11	9	2	15	5

min $\leftarrow 3$

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: on initialise le minimum à la première entrée du tableau.

Puis on parcourt les autres entrées du tableau une à une

en mettant à jour le minimum si nécessaire.

6	19	3	7	11	9	2	15	5

min $\leftarrow 2$

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: on initialise le minimum à la première entrée du tableau.

Puis on parcourt les autres entrées du tableau une à une

en mettant à jour le minimum si nécessaire.

6	19	3	7	11	9	2	15	5

min $\leftarrow 2$

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: on initialise le minimum à la première entrée du tableau.

Puis on parcourt les autres entrées du tableau une à une

en mettant à jour le minimum si nécessaire.

6	19	3	7	11	9	2	15	5

min $\leftarrow 2$

À la fin du parcours, on obtient le minimum

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

RechercheMin (tab: un tableau de n entiers){

1 min ← tab[0] *on initialise le minimum à la première entrée du tableau*

}

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

RechercheMin (tab: un tableau de n entiers){

1 min ← tab[0] *on initialise le minimum à la première entrée du tableau*

2 **for** i **from** 1 **to** n-1 **{** *Puis on parcourt les autres entrées du tableau une à une*

}

}

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]              on initialise le minimum à la première entrée du tableau
  for i from 1 to n-1 {     Puis on parcourt les autres entrées du tableau une à une
    if min > tab[i] {       en mettant à jour le minimum si nécessaire
      min ← tab[i]
    }
  }

}
```

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]              on initialise le minimum à la première entrée du tableau
  for i from 1 to n-1 {     Puis on parcourt les autres entrées du tableau une à une
    if min > tab[i] {       en mettant à jour le minimum si nécessaire
      min ← tab[i]
    }
  }
  return min                À la fin du parcours, on obtient le minimum
}
```

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]                 on initialise le minimum à la première entrée du tableau
  for i from 1 to n-1 {        Puis on parcourt les autres entrées du tableau une à une
    if min > tab[i] {          en mettant à jour le minimum si nécessaire
      min ← tab[i]
    }
  }
  return min                   À la fin du parcours, on obtient le minimum
}
```

L'algorithme est correct!

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]
  for i from 1 to n-1 {
    if min > tab[i] {
      min ← tab[i]
    }
  }
  return min
}
```

6	19	3	7	11	9	2	15	5

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]
  for i from 1 to n-1 {        ← i = 1
    if min > tab[i] {
      min ← tab[i]
    }
  }
  return min
}
```

6	19	3	7	11	9	2	15	5

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]
  for i from 1 to n-1 {        ←  i = 2
    if min > tab[i] {
      min ← tab[i]
    }
  }
  return min
}
```

6	19	3	7	11	9	2	15	5

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]
  for i from 1 to n-1 {        ←  i = 3
    if min > tab[i] {
      min ← tab[i]
    }
  }
  return min
}
```

6	19	3	7	11	9	2	15	5

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]
  for i from 1 to n-1 {        ←  i = 4
    if min > tab[i] {
      min ← tab[i]
    }
  }
  return min
}
```

6	19	3	7	11	9	2	15	5

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]
  for i from 1 to n-1 {        ←  i = 5
    if min > tab[i] {
      min ← tab[i]
    }
  }
  return min
}
```

6	19	3	7	11	9	2	15	5

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]
  for i from 1 to n-1 {        ←  i = 6
    if min > tab[i] {
      min ← tab[i]
    }
  }
  return min
}
```

6	19	3	7	11	9	2	15	5

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]
  for i from 1 to n-1 {        ←  i = 7
    if min > tab[i] {
      min ← tab[i]
    }
  }
  return min
}
```

6	19	3	7	11	9	2	15	5

Écrire un algorithme

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]
  for i from 1 to n-1 {        ←  i = 8
    if min > tab[i] {
      min ← tab[i]
    }
  }
  return min
}
```

6	19	3	7	11	9	2	15	5

Écrire un algorithme

Problème: recherche ~~du minimum~~ de l'élément x dans un tableau d'entiers

données: un tableau d'entiers et un entier

résultat: l'indice de l'entier recherché dans le tableau s'il s'y trouve, -1 sinon

```
RechercheMin (tab: un tableau de n entiers){
  min ← tab[0]                 on initialise le minimum à la première entrée du tableau
  for i from 1 to n-1 {        Puis on parcourt les autres entrées du tableau une à une
    if min > tab[i] {          en mettant à jour le minimum si nécessaire
      min ← tab[i]
    }
  }
  return min                   À la fin du parcours, on obtient le minimum
}
```

Écrire un algorithme

Problème: recherche ~~du minimum~~ de l'élément x dans un tableau d'entiers

données: un tableau d'entiers et un entier

résultat: l'indice de l'entier recherché dans le tableau s'il s'y trouve, -1 sinon

```
RechercheElt (tab: un tableau de n entiers, x: entier){
  ind ← -1                  on initialise l'indice de x dans le tableau à -1, ie. x n'a pas encore été vu
  for i from 0 to n-1 {     Puis on parcourt toutes les entrées du tableau une à une
    if tab[i] = x {         en mettant à jour l'indice de x dans le tableau si nécessaire
      ind ← i
    }
  }
  return ind                À la fin du parcours, on obtient l'indice de x dans tab
                            s'il s'y trouve, -1 sinon
}
```

Écrire un algorithme

Problème: recherche ~~du minimum~~ de l'élément x dans un tableau d'entiers

données: un tableau d'entiers et un entier

résultat: l'indice de l'entier recherché dans le tableau s'il s'y trouve, -1 sinon

```
RechercheElt (tab: un tableau de n entiers, x: entier){
1   ind ← -1                  on initialise l'indice de x dans le tableau à -1, ie. x n'a pas encore été vu
2   for i from 0 to n-1 {     Puis on parcourt toutes les entrées du tableau une à une
3     if tab[i] = x {         en mettant à jour l'indice de x dans le tableau si nécessaire
4       ind ← i
5     }
6   }
7   return ind                À la fin du parcours, on obtient l'indice de x dans tab
                              s'il s'y trouve, -1 sinon
}
```

Amélioration: retourner l'indice de x dès qu'on le trouve dans le tableau

Écrire un algorithme

Problème: recherche ~~du minimum~~ de l'élément x dans un tableau d'entiers

données: un tableau d'entiers et un entier

résultat: l'indice de l'entier recherché dans le tableau s'il s'y trouve, -1 sinon

RechercheElt (tab: un tableau de n entiers, x: entier){

```
1   ~~ind ← -1~~
2   for i from 0 to n-1 { On parcourt toutes les entrées du tableau une à une
3     if tab[i] = x {     en retournant l'indice de x dans le tableau si on trouve x
4       return i
5     }
6   }
7   return -1           Si on arrive ici, c'est que x n'a pas été trouvé
                        et on retourne -1
}
```

Amélioration: retourner l'indice de x dès qu'on le trouve dans le tableau

Conclusion

Losqu'on connait bien un algorithme pour résoudre un problème, on peut en déduire d'autres pour résoudre d'autres problèmes similaires en effectuant de petits changements

Conclusion

Losqu'on connait bien un algorithme pour résoudre un problème, on peut en déduire d'autres pour résoudre d'autres problèmes similaires en effectuant de petits changements

Questions

On a parlé d'amélioration, mais qu'est-ce que cela signifie?

Comment comparer deux algorithmes résolvant le même problème sur les mêmes instances?

Un nouvel algorithme pour trouver le minimum d'un tableau

Problème: recherche du minimum dans un tableau d'entiers

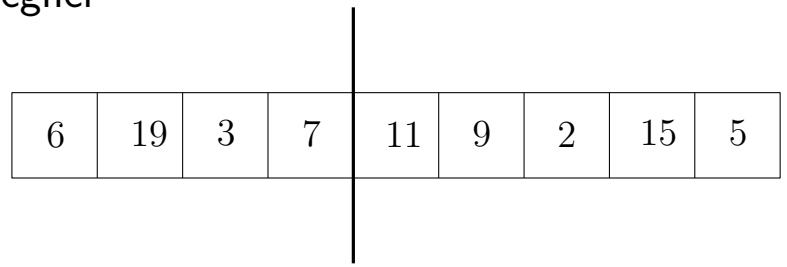
données: un tableau d'entiers résultat: l'élément minimum du tableau

Un nouvel algorithme pour trouver le minimum d'un tableau

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: diviser pour régner

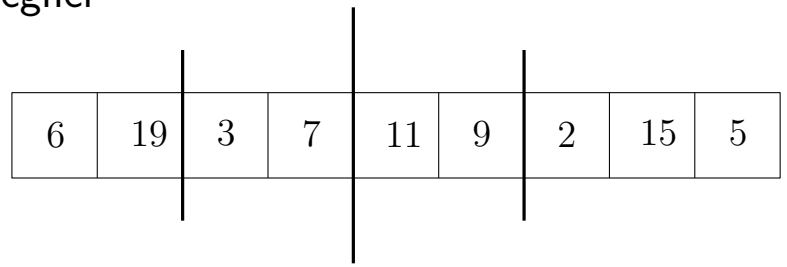


Un nouvel algorithme pour trouver le minimum d'un tableau

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: diviser pour régner

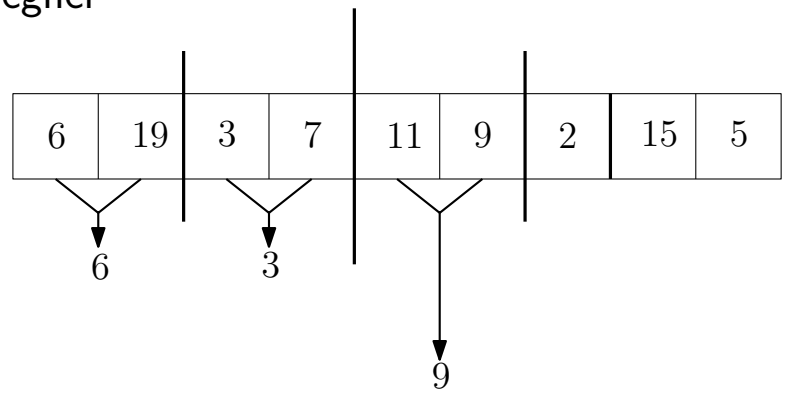


Un nouvel algorithme pour trouver le minimum d'un tableau

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: diviser pour régner

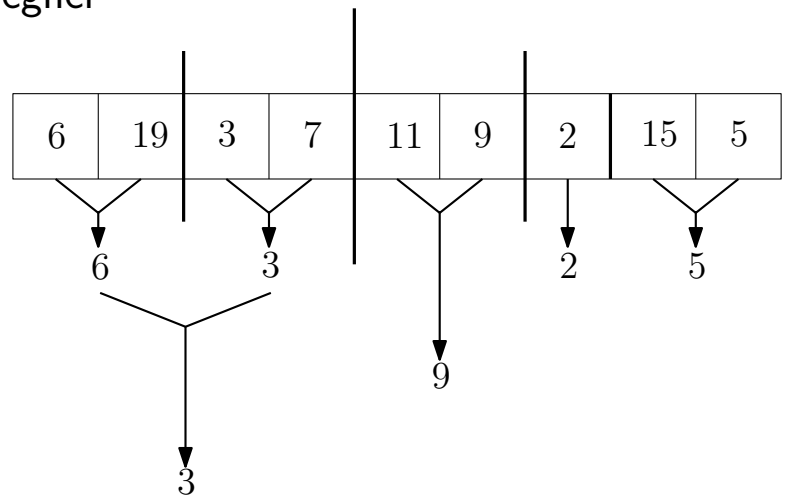


Un nouvel algorithme pour trouver le minimum d'un tableau

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: diviser pour régner

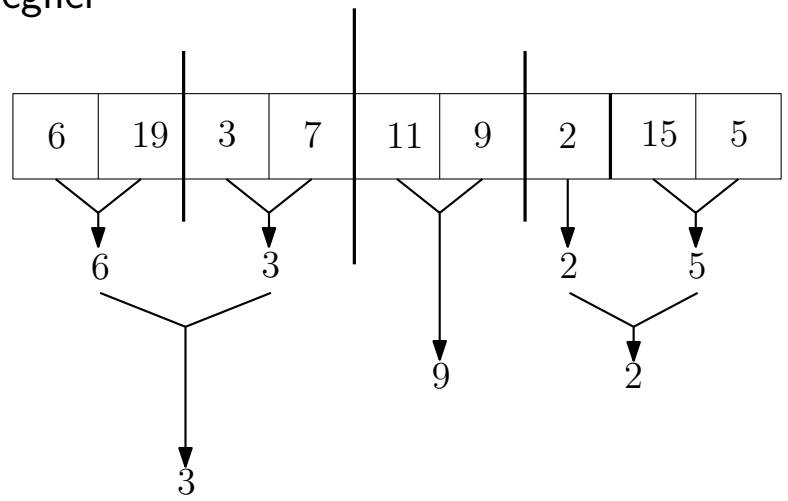


Un nouvel algorithme pour trouver le minimum d'un tableau

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: diviser pour régner

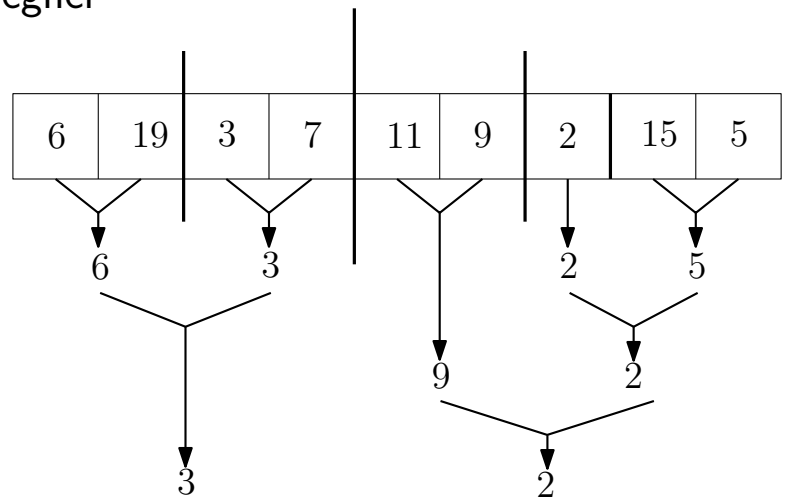


Un nouvel algorithme pour trouver le minimum d'un tableau

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: diviser pour régner

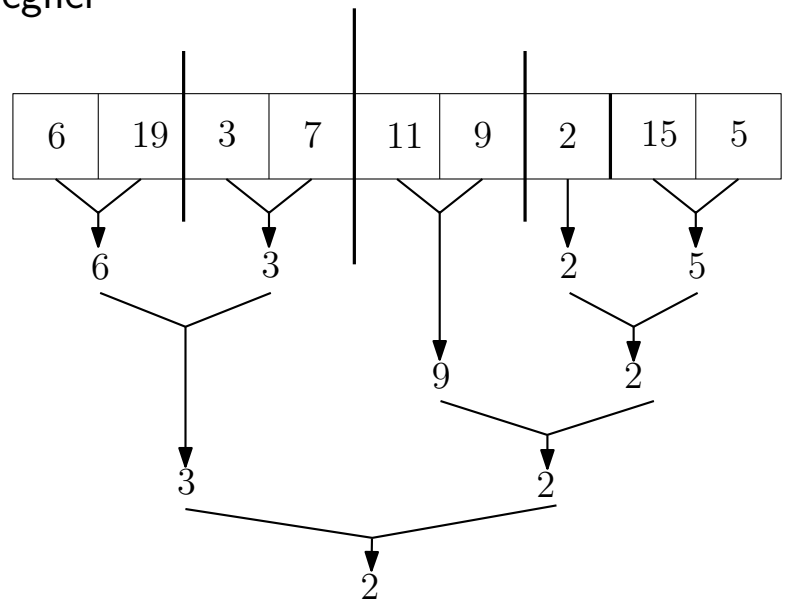


Un nouvel algorithme pour trouver le minimum d'un tableau

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: diviser pour régner

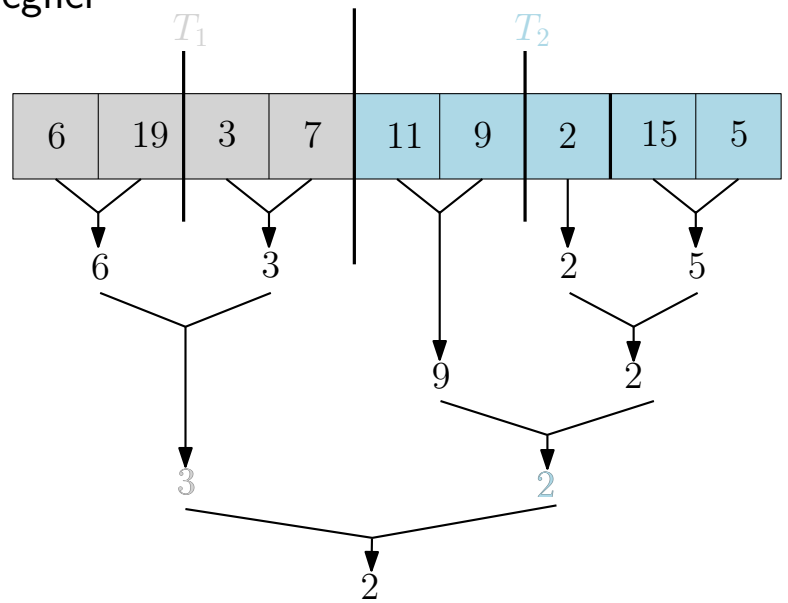


Un nouvel algorithme pour trouver le minimum d'un tableau

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: diviser pour régner



si T a plus qu'un élément

scinder le tableau en deux sous-tableaux T_1 et T_2

calculer le minimum, $\min_1$ de T_1

calculer le minimum, $\min_2$ de T_2

retourner le minimum de $\min_1$ et $\min_2$

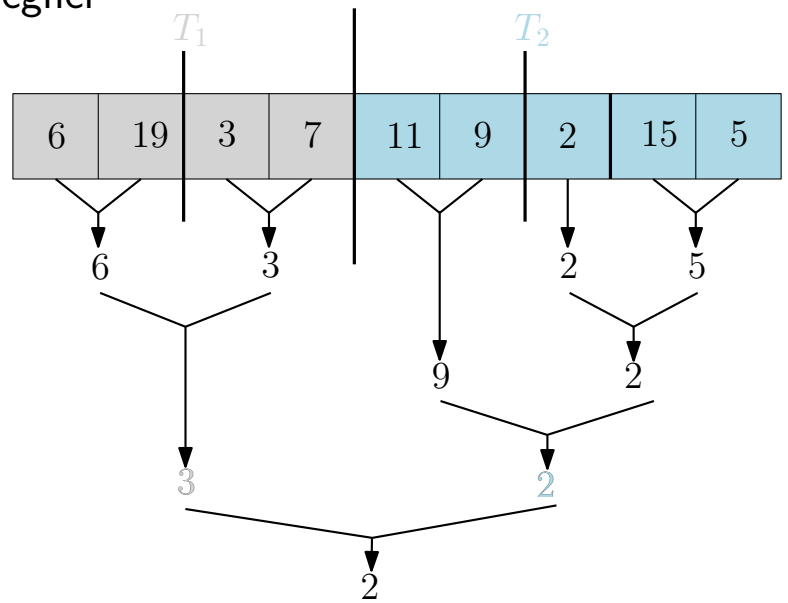
sinon retourner l'unique élément de T

Un nouvel algorithme pour trouver le minimum d'un tableau

Problème: recherche du minimum dans un tableau d'entiers

données: un tableau d'entiers résultat: l'élément minimum du tableau

Principe: diviser pour régner



si T a plus qu'un élément

scinder le tableau en deux sous-tableaux T_1 et T_2

calculer le minimum, $\min_1$ de T_1

calculer le minimum, $\min_2$ de T_2

retourner le minimum de $\min_1$ et $\min_2$

sinon retourner l'unique élément de T

Algorithme récursif

Questions

On a parlé d'amélioration, mais qu'est-ce que cela signifie?

Comment comparer deux algorithmes résolvant le même problème sur les mêmes instances?

Questions

On a parlé d'amélioration, mais qu'est-ce que cela signifie?

Comment comparer deux algorithmes résolvant le même problème sur les mêmes instances?

Comparer deux algorithmes

comparer les vitesses d'exécution sur les mêmes instances $\longrightarrow$ algorithme **efficace**

Questions

On a parlé d'amélioration, mais qu'est-ce que cela signifie?

Comment comparer deux algorithmes résolvant le même problème sur les mêmes instances?

Comparer deux algorithmes

comparer les vitesses d'exécution sur les mêmes instances $\longrightarrow$ algorithme **efficace**

compter le nombre d'opérations élémentaires
effectuées lors du déroulement de l'algorithme

aujourd'hui: 1 opération élémentaire $\sim$ 1 nanoseconde (10^{-9} s)

choisir une bonne structure de données

Étude de la complexité en temps

Questions

On a parlé d'amélioration, mais qu'est-ce que cela signifie?

Comment comparer deux algorithmes résolvant le même problème sur les mêmes instances?

Comparer deux algorithmes

comparer les vitesses d'exécution sur les mêmes instances → algorithme **efficace**

comparer l'espace nécessaire en mémoire selon les instances

→ choisir une bonne structure de données

Etude de la complexité en espace

Questions

On a parlé d'amélioration, mais qu'est-ce que cela signifie?

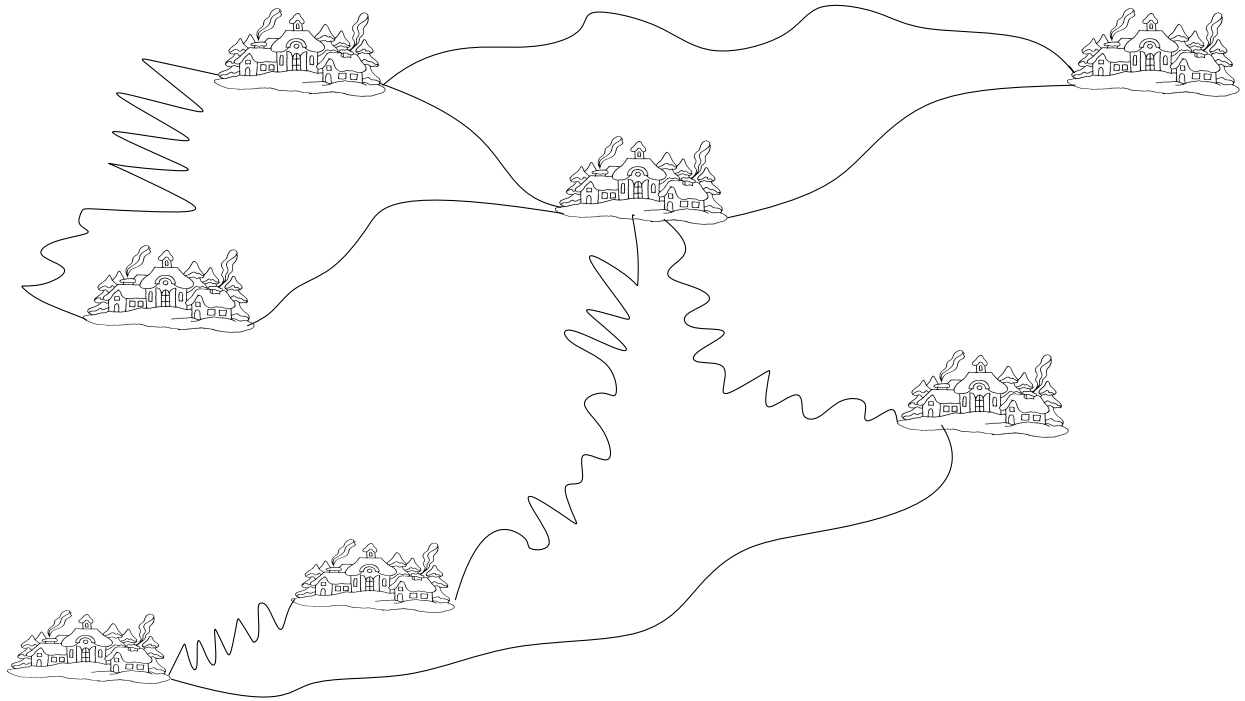
Comment comparer deux algorithmes résolvant le même problème sur les mêmes instances?

Comparer deux algorithmes

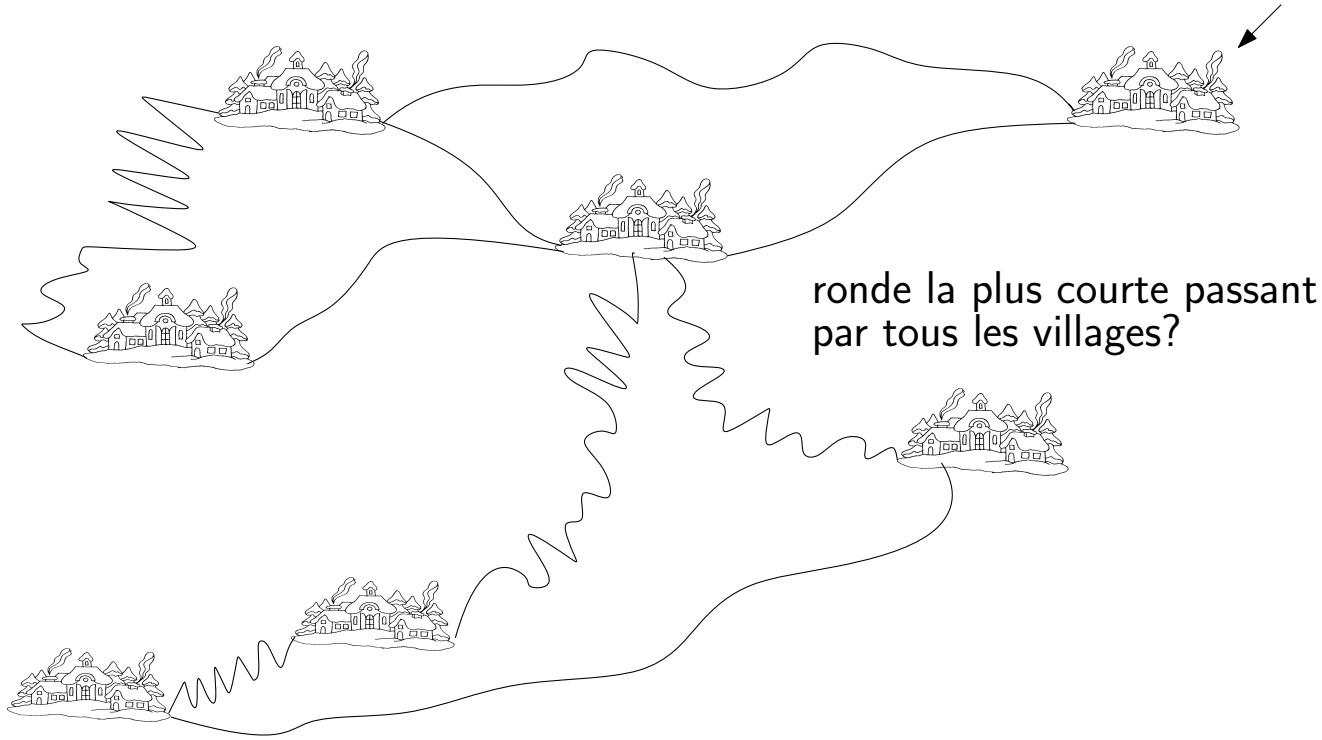
comparer les vitesses d'exécution sur les mêmes instances $\longrightarrow$ algorithme **efficace**
comparer l'espace nécessaire en mémoire selon les instances
lisibilité de l'algorithme, simplicité du principe, simplicité de l'implémentation...

Attention il existe des algorithmes pour lesquels on ne connait **pas de solution efficace!**

Le problème du voyageur de commerce

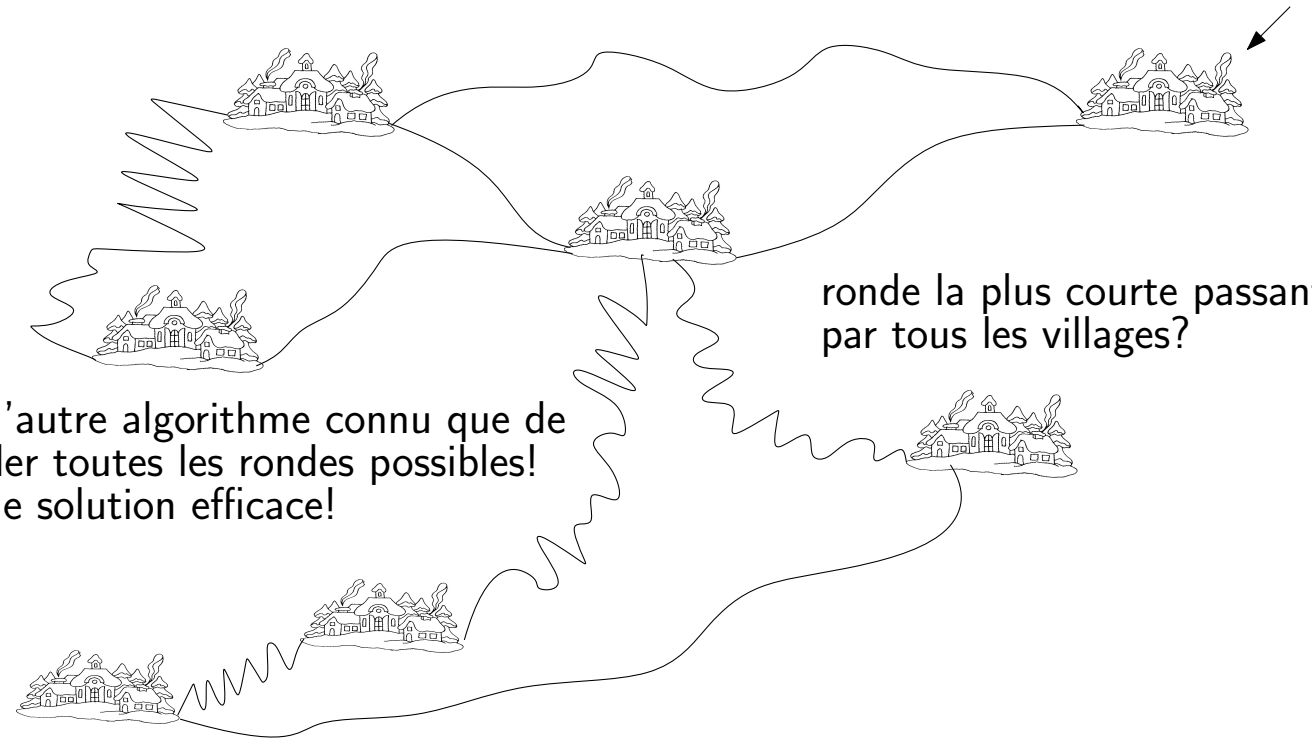


Le problème du voyageur de commerce



ronde la plus courte passant
par tous les villages?

Le problème du voyageur de commerce



ronde la plus courte passant par tous les villages?

Pas d'autre algorithme connu que de calculer toutes les rondes possibles!
Pas de solution efficace!

aujourd'hui: 1 opération élémentaire $\sim$ 1 nanoseconde (10^{-9} s)

n	20	40	60	100	300
n^2	$\frac{1}{2500}$ ms	$\frac{1}{625}$ ms	$\frac{1}{278}$ ms	$\frac{1}{100}$ ms	$\frac{1}{11}$ ms
n^5	3 ms	$\frac{1}{10}$ s	$\frac{78}{100}$ s	10 s	$40,5$ min
2^n	1 ms	$18,3$ min	$36,5$ années	4.10^{11} siècles	...
n^n	$3,3.10^7$ siècles	...			

taille de la donnée

nombres d'opérations élémentaires

aujourd'hui: 1 opération élémentaire $\sim$ 1 nanoseconde (10^{-9} s)

n	20	40	60	100	300
n^2	$\frac{1}{2500}$ ms	$\frac{1}{625}$ ms	$\frac{1}{278}$ ms	$\frac{1}{100}$ ms	$\frac{1}{11}$ ms
n^5	3 ms	$\frac{1}{10}$ s	$\frac{78}{100}$ s	10 s	$40,5$ min
2^n	1 ms	$18,3$ min	$36,5$ années	4.10^{11} siècles	...
n^n	$3,3.10^7$ siècles	...			

taille de la donnée

nombres d'opérations élémentaires

Et il existe des problème qu'on ne sait pas résoudre avec un algorithme!

Et il existe des problème qu'on ne sait pas résoudre avec un algorithme!

Le problème de l'arrêt : peut-on écrire une algorithme A qui

pour tout algorithme B et toute instance I

détermine si B s'arrête pour la donnée I?

Et il existe des problème qu'on ne sait pas résoudre avec un algorithme!

Le problème de l'arrêt : peut-on écrire une algorithme A qui

pour tout algorithme B et toute instance I

détermine si B s'arrête pour la donnée I?

NON

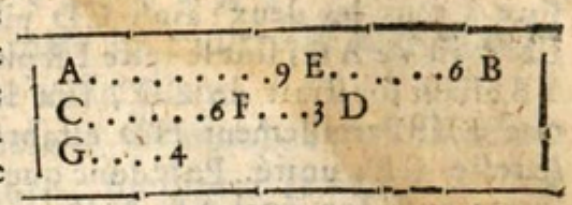
L'algorithme d'Euclide ($\sim$ 300 avant J.C.)

PROBL. I. PROP. II.

Estans donnez deux nombres non premiers entr'eux ; trouver la plus grande commune mesure d'iceux.

Soient donnez deux nombres non premiers entr'eux AB & CD, desquels CD soit le moindre : il faut trouver la plus grande commune mesure d'iceux.

Numérisé par Google



Soit soustrait le plus petit nombre CD, tant de fois que faire se pourra du plus grand AB: s'il reste quelque chose, comme EB, iceluy soit osté de CD, tellement qu'il reste encor FD; & ainsi soit soustrait continuellement le plus petit du plus grand, jusques à ce qu'on parvienne à un nombre qui mesure le precedent : ce qui aviendra necessairement : car si on parvenoit jusques à l'unité, les nombres AB, CD, par la prec. prop. seroient premiers entr'eux ; contre l'hypothese. Donc que FD mesurant EB, il ne reste rien. Je dis que FD est la plus grande commune mesure de AB & CD.

Car qu'il les mesure tous deux, nous le prouverons ainsi. D'autant que FD mesure EB, & EB mesure CF,

L'algorithme d'Euclide ($\sim$ 300 avant J.C.)

Calcule le pgcd de deux nombres

30 21

L'algorithme d'Euclide ($\sim$ 300 avant J.C.)

Calcule le pgcd de deux nombres

$$30 = 21 + 9$$

$$= 2 \times 9 + 3 + 9$$

L'algorithme d'Euclide ($\sim$ 300 avant J.C.)

Calcule le pgcd de deux nombres

$$30 = 21 + 9$$

$$= 2 \times 9 + 3 + 9$$

L'algorithme d'Euclide ($\sim$ 300 avant J.C.)

Calcule le pgcd de deux nombres

$$
\begin{aligned}
30 &= 21 + 9 \\
&= 2 \times 9 + 3 + 9 \\
&= 2 \times 3 \times 3 + 0 + 3 + 9
\end{aligned}
$$

L'algorithme d'Euclide ($\sim$ 300 avant J.C.)

Calcule le pgcd de deux nombres

$$
\begin{aligned}
30 &= 21 + 9 \\
&= 2 \times 9 + 3 + 9 \\
&= 2 \times 3 \times 3 + 0 + 3 + 9
\end{aligned}
$$

$\longrightarrow$ 3 est le pgcd de 30 et 21.

L'algorithme d'Euclide ($\sim$ 300 avant J.C.)

Calcule le pgcd de deux nombres

Principe:

pgcd$(n_1, 0) = n_1$

pgcd$(n_1, n_2) =$ pgcd$(n_2,$ reste de la division de n_1 par $n_2)$

L'algorithme d'Euclide ($\sim$ 300 avant J.C.)

Calcule le pgcd de deux nombres

Principe:

pgcd$(n_1, 0) = n_1$

pgcd$(n_1, n_2) =$ pgcd$(n_2,$ reste de la division de n_1 par $n_2)$

Algorithme:

```
pgcd(n1 et n2 entiers positifs ou nuls){
    if n2 = 0 {
        return n1
    }else {
        return pgcd(n2, reste(n1, n2))
}
```

Conclusion

Connaître des algorithmes permet d'en concevoir de nouveaux régit par un principe similaire

$\longrightarrow$ lecture, modification et écriture d'algorithmes

Un algorithme doit être efficace pour pouvoir donner une réponse dans un temps raisonnable en utilisant une place en mémoire raisonnable.

$\longrightarrow$ étude de la complexité d'un algorithme