Algorithmique et structure de données

Rohan Fossé

2021-2022

Trivia

Intervenants

- Rohan Fossé, rfosse@labri.fr
- Jonathan Narboni, jnarboni@labri.fr

Organisation

- Cours : $6 \times 1h20$
- EI :$6 \times 2h40$
- TP : $5 \times 1h20$

Modalités de contrôle

- Examen écrit sans document : 2h

Objectifs de l'UE

- Se familiariser avec des problèmes classiques et leur solutions ;
- Se préparer à trouver des solutions algorithmiques à des problèmes en sachant comparer leurs performances ;
- S'entraîner à écrire des algorithmes et connaître les différentes structures de données.

Introduction

Algorithmique et Structures de Données

- **Algorithme** : séquence d'opérations de calcul élémentaires, organisé selon des règles précises dans le but de résoudre un problème donné.
- **Structures de données** : moyen de stocker et organiser des données pour faciliter l'accès à ces données et leur modification.

Algorithmes

Exemple d'algorithme

Boire son café

1. Prenez une dosette de café.

Exemple d'algorithme

Boire son café

1. Prenez une dosette de café.
2. Mettez-la dans la machine à café.

Exemple d'algorithme

Boire son café

1. Prenez une dosette de café.
2. Mettez-la dans la machine à café.
3. Vérifiez si la machine à café est allumée. Si ce n'est pas le cas, allumez la machine.

Exemple d'algorithme

Boire son café

1. Prenez une dosette de café.
2. Mettez-la dans la machine à café.
3. Vérifiez si la machine à café est allumée. Si ce n'est pas le cas, allumez la machine.
4. Vérifiez si le filtre à eau est suffisamment rempli. - Si ce n'est pas le cas, ajoutez de l'eau.

Exemple d'algorithme

Boire son café

1. Prenez une dosette de café.
2. Mettez-la dans la machine à café.
3. Vérifiez si la machine à café est allumée. Si ce n'est pas le cas, allumez la machine.
4. Vérifiez si le filtre à eau est suffisamment rempli. - Si ce n'est pas le cas, ajoutez de l'eau.
5. Placez une tasse à café sous le distributeur de café.

Exemple d'algorithme

Boire son café

1. Prenez une dosette de café.
2. Mettez-la dans la machine à café.
3. Vérifiez si la machine à café est allumée. Si ce n'est pas le cas, allumez la machine.
4. Vérifiez si le filtre à eau est suffisamment rempli. - Si ce n'est pas le cas, ajoutez de l'eau.
5. Placez une tasse à café sous le distributeur de café.
6. Appuyez sur le bouton café.

Exemple d'algorithme

Boire son café

1. Prenez une dosette de café.
2. Mettez-la dans la machine à café.
3. Vérifiez si la machine à café est allumée. Si ce n'est pas le cas, allumez la machine.
4. Vérifiez si le filtre à eau est suffisamment rempli. - Si ce n'est pas le cas, ajoutez de l'eau.
5. Placez une tasse à café sous le distributeur de café.
6. Appuyez sur le bouton café.
7. Le café est en train d'être servi. - Attendez que la machine indique que le café est prêt.

Exemple d'algorithme

Boire son café

1. Prenez une dosette de café.
2. Mettez-la dans la machine à café.
3. Vérifiez si la machine à café est allumée. Si ce n'est pas le cas, allumez la machine.
4. Vérifiez si le filtre à eau est suffisamment rempli. - Si ce n'est pas le cas, ajoutez de l'eau.
5. Placez une tasse à café sous le distributeur de café.
6. Appuyez sur le bouton café.
7. Le café est en train d'être servi. - Attendez que la machine indique que le café est prêt.
8. Si le café est prêt - Sortez la tasse à café de la machine à café.

Autre exemple d'algorithme

Se laver les mains

1. Ouvrir l'eau

Autre exemple d'algorithme

Se laver les mains

1. Ouvrir l'eau
2. Mettre du savon

Autre exemple d'algorithme

Se laver les mains

1. Ouvrir l'eau
2. Mettre du savon
3. Nettoyer ses mains avec l'eau

Autre exemple d'algorithme

Se laver les mains

1. Ouvrir l'eau
2. Mettre du savon
3. Nettoyer ses mains avec l'eau
4. Éteindre l'eau

Autre exemple d'algorithme

Se laver les mains

1. Ouvrir l'eau
2. Mettre du savon
3. Nettoyer ses mains avec l'eau
4. Éteindre l'eau
5. Se sécher les mains

Autre exemple d'algorithme

Se laver les mains

1. Ouvrir l'eau
2. Mettre du savon
3. Nettoyer ses mains avec l'eau
4. Éteindre l'eau
5. Se sécher les mains

Qu'est ce qu'un algorithme ?

Un algorithme

- Un algorithme est une méthode systématique (comme une recette) pour résoudre un problème donné ;
- Il se compose d'une suite d'opérations simples à effectuer pour résoudre ce problème ;
- En informatique, **cette méthode doit être applicable par un ordinateur**.

Importance de l'algorithmique

- Pour un problème donné, il existe plusieurs algorithmes ;
- Il est facile d'écrire des algorithmes faux ou inefficaces ;
- Une erreur peut faire la différence entre quelques minutes de calculs et plusieurs heures ;
- C'est souvent une question d'utilisation de structures de données ou d'algorithmes connus dans la littérature.

Exemple : la ville et la pizzeria

Le problème

Nous allons considérer 3 villes contenant 14 maisons et 1 pizzeria. Nous souhaitons savoir quelle ville permet aux livreurs de pizza de faire les trajets les plus rapides et pourquoi.

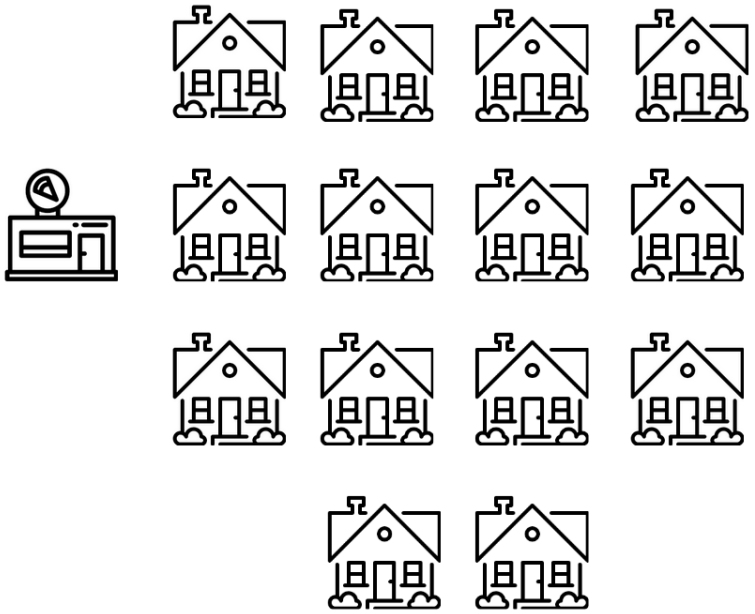


Figure 1: 1 pizzeria et 14 maisons

Exemple : la ville et la pizzeria

Le problème

Nous allons considérer 3 villes contenant 14 maisons et 1 pizzeria. Nous souhaitons savoir quelle ville permet aux livreurs de pizza de faire les trajets les plus rapides et pourquoi.

Temps de calcul

On suppose qu'il faut une unité de temps pour passer d'une maison à un autre, en suivant une rue.

Figure 2: Trajet entre la pizzeria et une maison

Exemple : la ville et la pizzeria

Le problème

Nous allons considérer 3 villes contenant 14 maisons et 1 pizzeria. Nous souhaitons savoir quelle ville permet aux livreurs de pizza de faire les trajets les plus rapides et pourquoi.

Temps de calcul

On suppose qu'il faut une unité de temps pour passer d'une maison à un autre, en suivant une rue.

Figure 2: Trajet entre la pizzeria et une maison

Dans le pire cas, quel est le temps mis par un livreur pour aller de la pizzeria jusqu'à une maison ?

Ville A

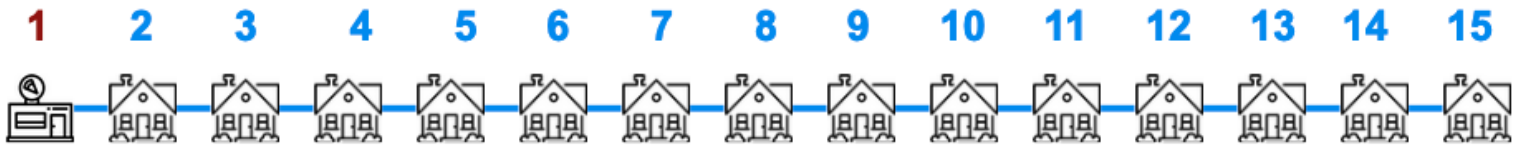
Organisation de la ville

Les maisons sont rangées dans l'ordre croissant en ligne droite. La pizzeria se trouve au numéro 1.

Ville A

Organisation de la ville

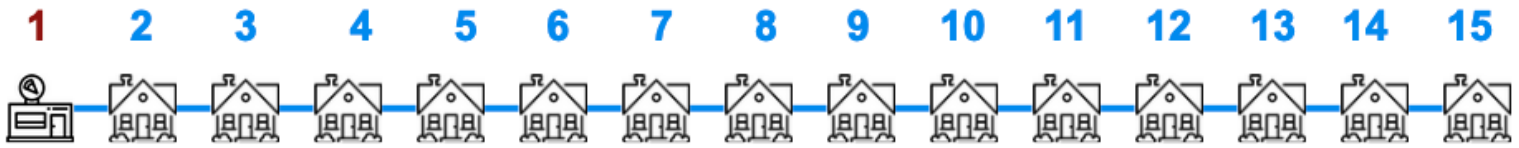
Les maisons sont rangées dans l'ordre croissant en ligne droite. La pizzeria se trouve au numéro 1.



Ville A

Organisation de la ville

Les maisons sont rangées dans l'ordre croissant en ligne droite. La pizzeria se trouve au numéro 1.

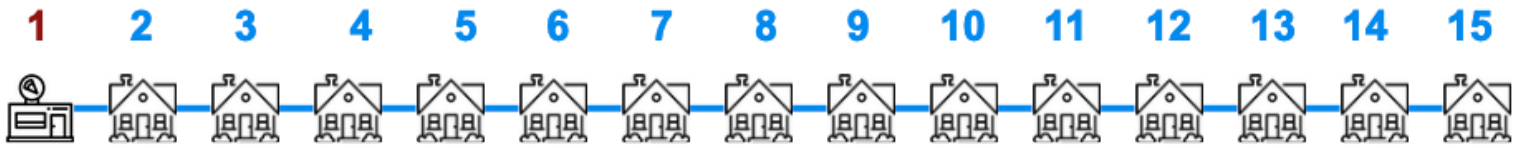


Quel est le pire temps possible ?

Ville A

Organisation de la ville

Les maisons sont rangées dans l'ordre croissant en ligne droite. La pizzeria se trouve au numéro 1.



Quel est le pire temps possible ? **14**

Ville B

Organisation de la ville

Même organisation que dans la ville A, mais cette fois-ci la pizzeria est au numéro **8**

Ville B

Organisation de la ville

Même organisation que dans la ville A, mais cette fois-ci la pizzeria est au numéro **8**

Ville B

Organisation de la ville

Même organisation que dans la ville A, mais cette fois-ci la pizzeria est au numéro **8**

Quel est le pire temps possible ?

Ville B

Organisation de la ville

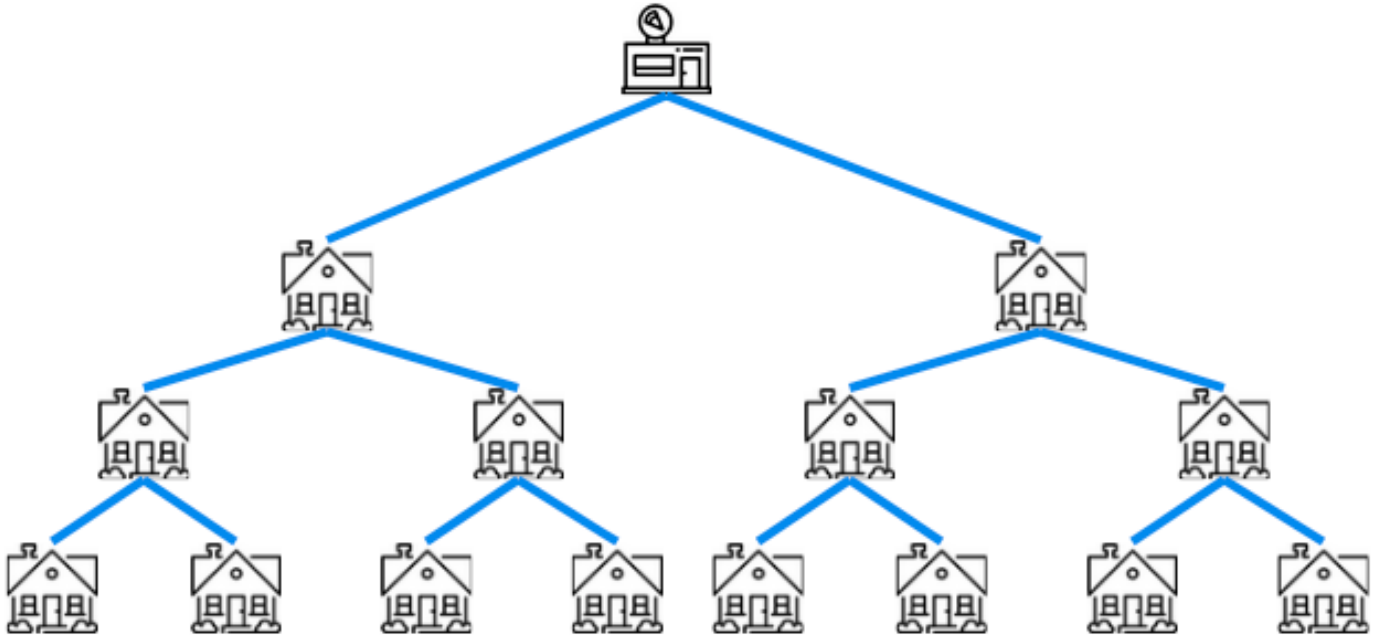
Même organisation que dans la ville A, mais cette fois-ci la pizzeria est au numéro **8**

Quel est le pire temps possible ? **7**

Ville C

Organisation de la ville

Cette fois-ci, les maisons sont organisées en embranchements, la pizzeria est tout en **haut**.

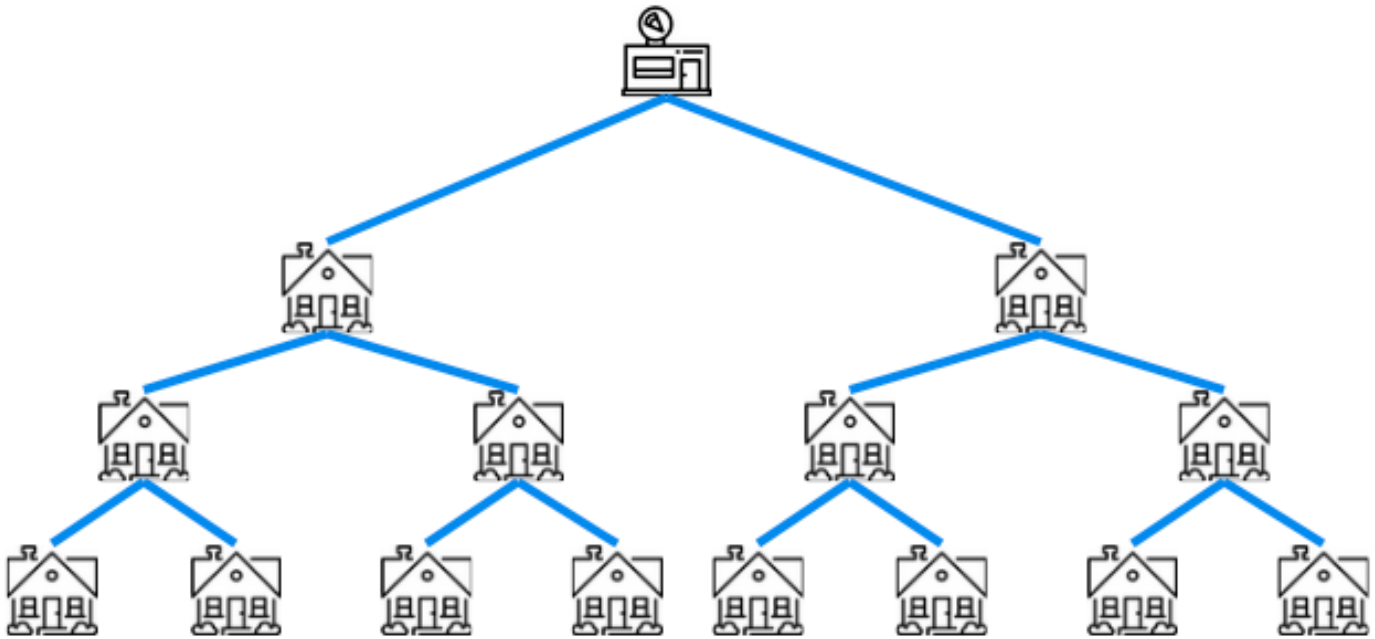


Quel est le pire temps possible ?

Ville C

Organisation de la ville

Cette fois-ci, les maisons sont organisées en embranchements, la pizzeria est tout en **haut**.

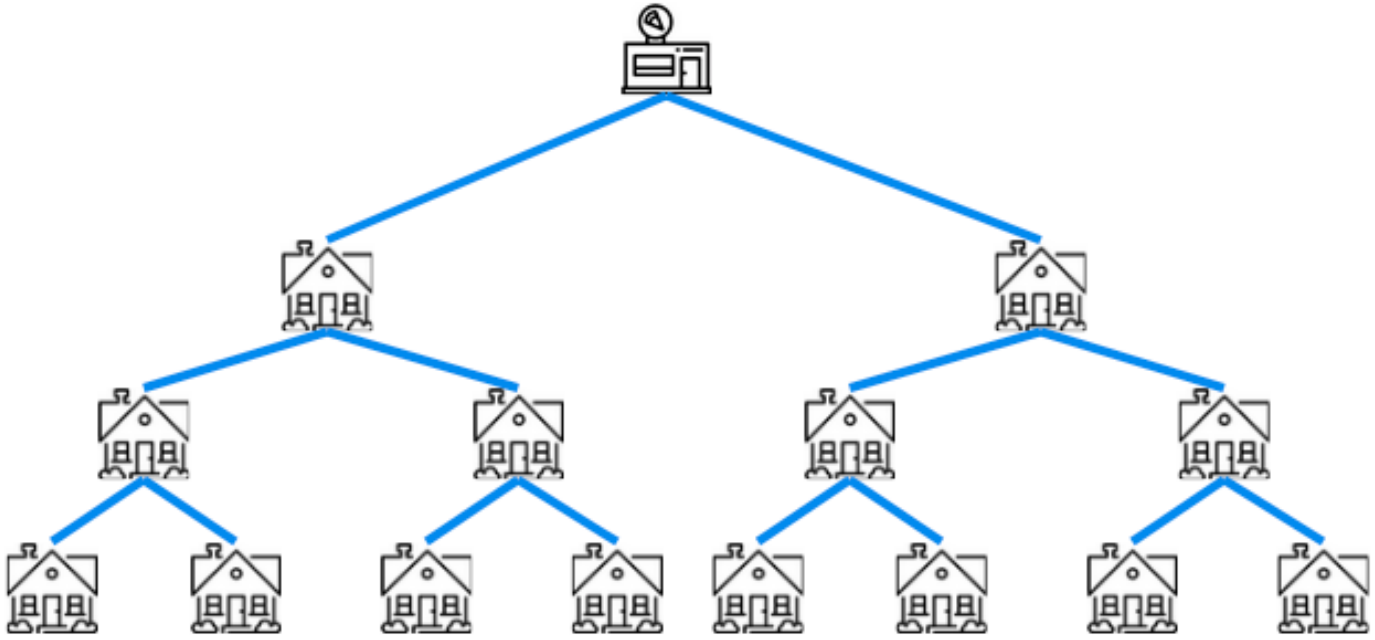


Quel est le pire temps possible ? **3**

Ville C

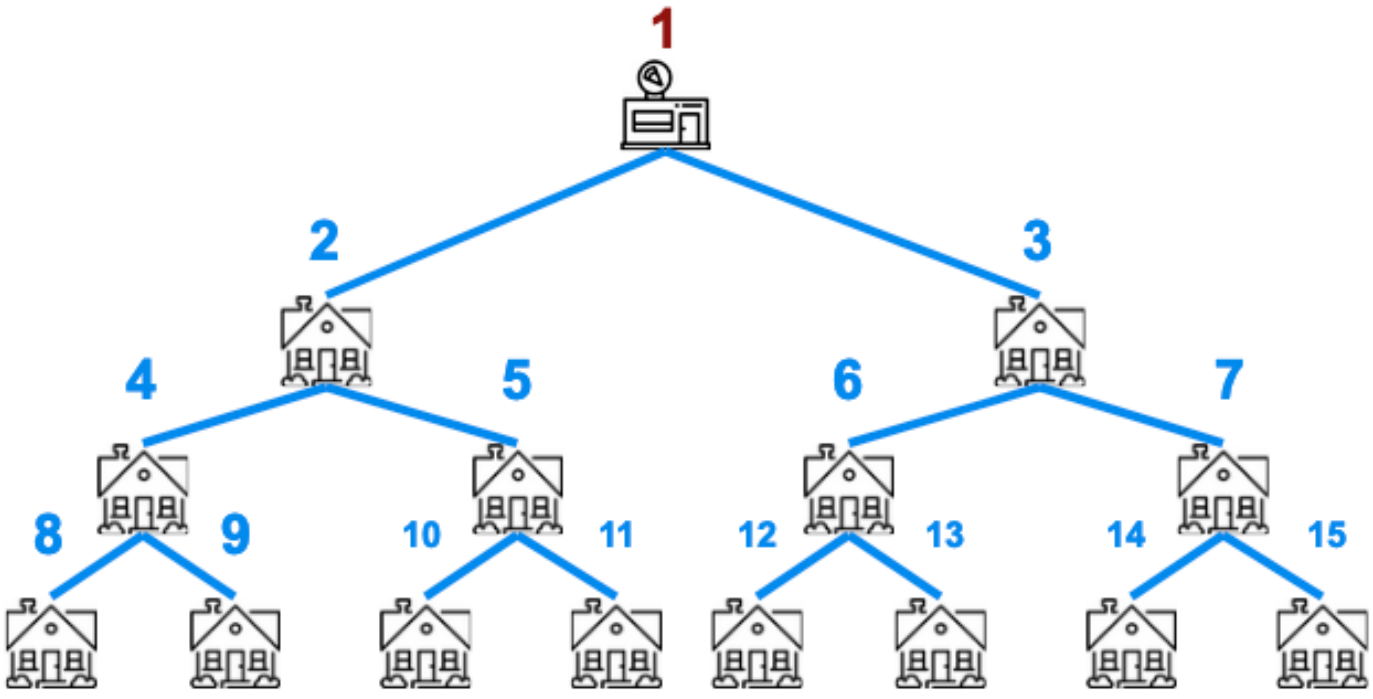
Organisation de la ville

Cette fois-ci, les maisons sont organisées en embranchements, la pizzeria est tout en **haut**.

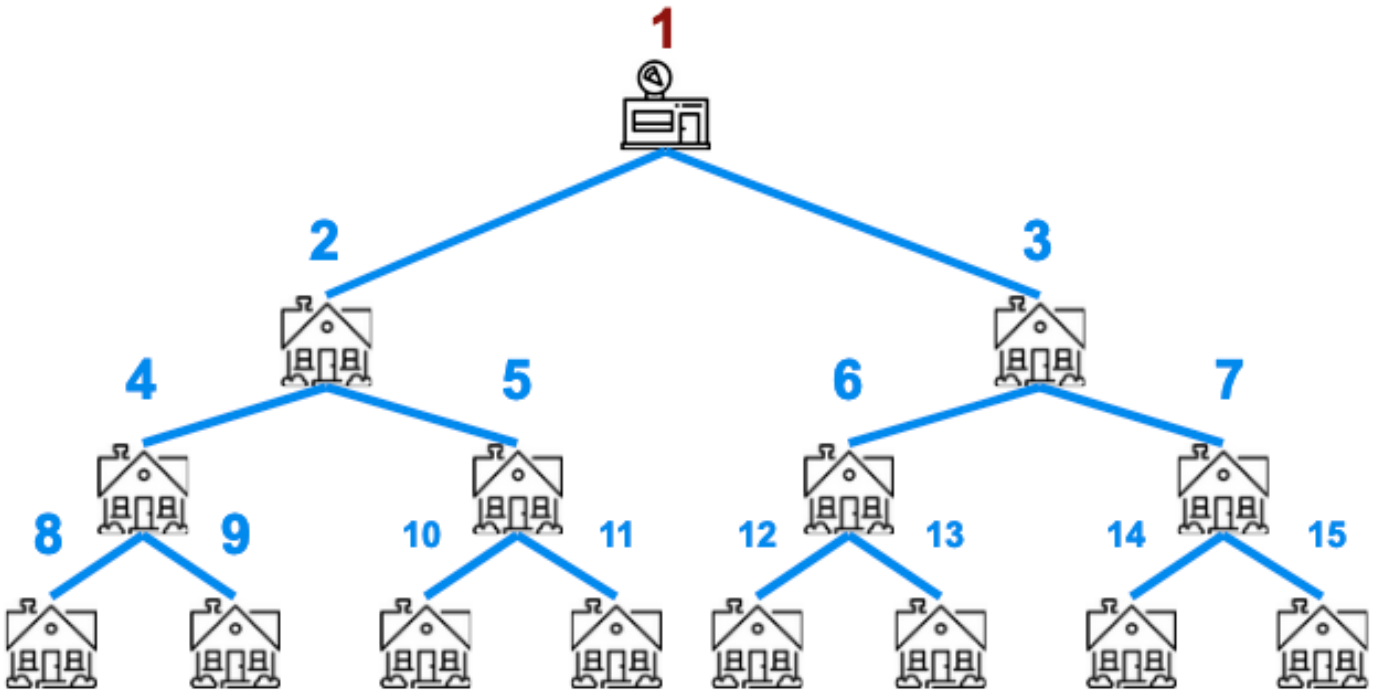


Comment numéroter les maisons ?

Ville C : première numérotation

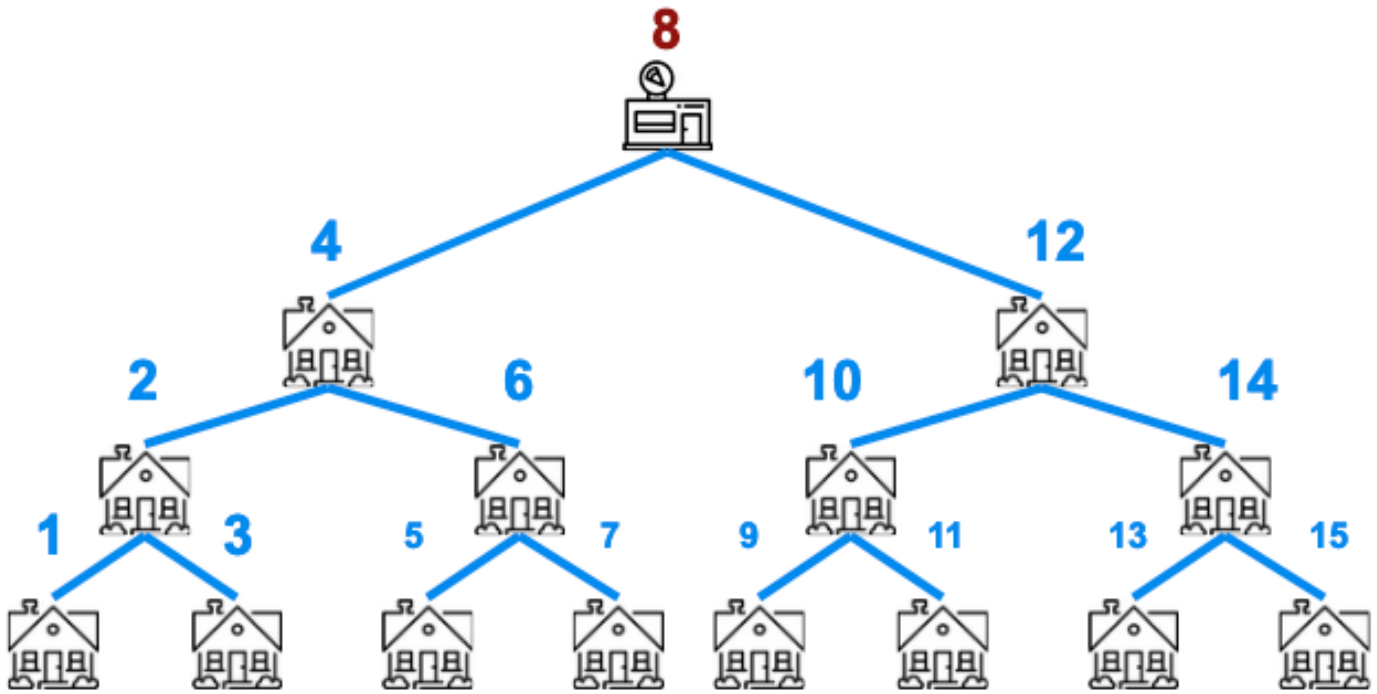


Ville C : première numérotation

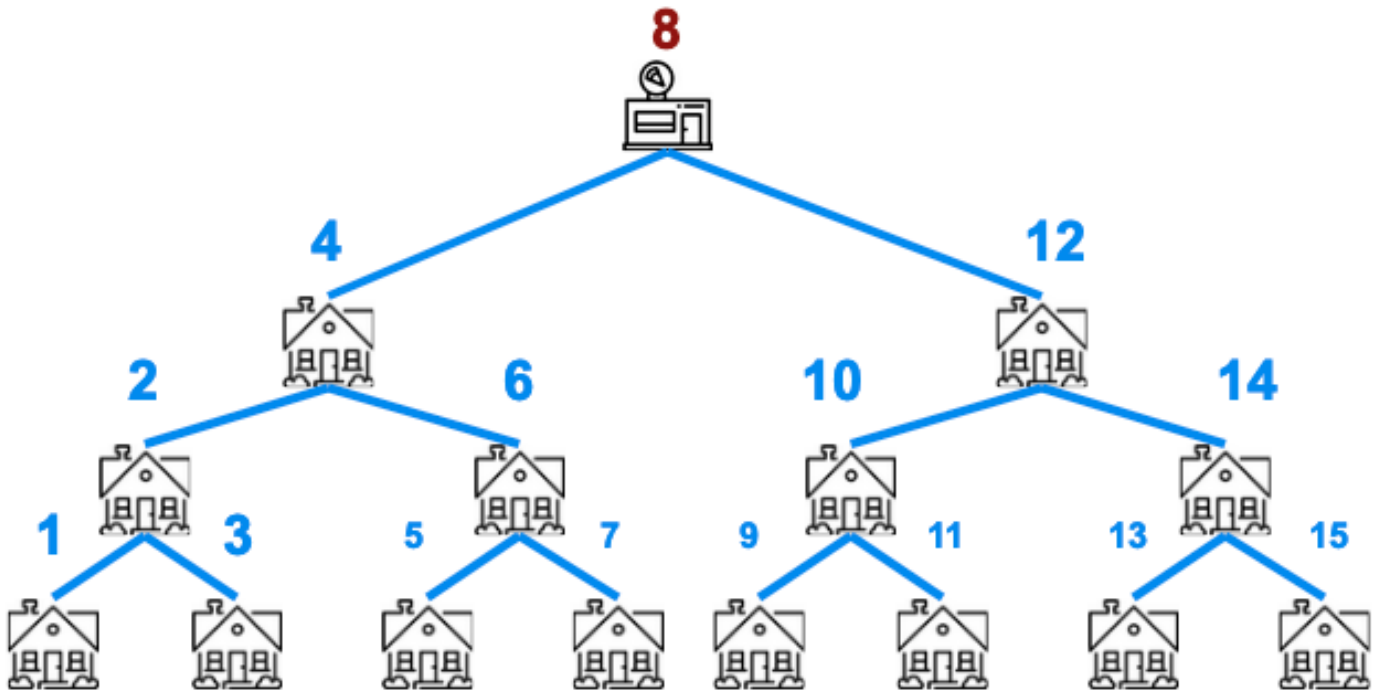


Comment choisir la bonne rue à prendre ?

Ville C : deuxième numérotation



Ville C : deuxième numérotation



Dans ce cas là, si le numéro de la maison à livrer est plus petit que celui sur lequel on se trouve, on va à gauche, sinon on va à droite.

Tableau récapitulatif

Nombre de maisons	Ville A	Ville B	Ville C
15	14	7	3

Tableau récapitulatif

Nombre de maisons	Ville A	Ville B	Ville C
15	14	7	3
1023			

Tableau récapitulatif

Nombre de maisons	Ville A	Ville B	Ville C
15	14	7	3
1023	1022	511	9

Tableau récapitulatif

Nombre de maisons	Ville A	Ville B	Ville C
15	14	7	3
1023	1022	511	9
n			

Tableau récapitulatif

Nombre de maisons	Ville A	Ville B	Ville C
15	14	7	3
1023	1022	511	9
n	n-1		

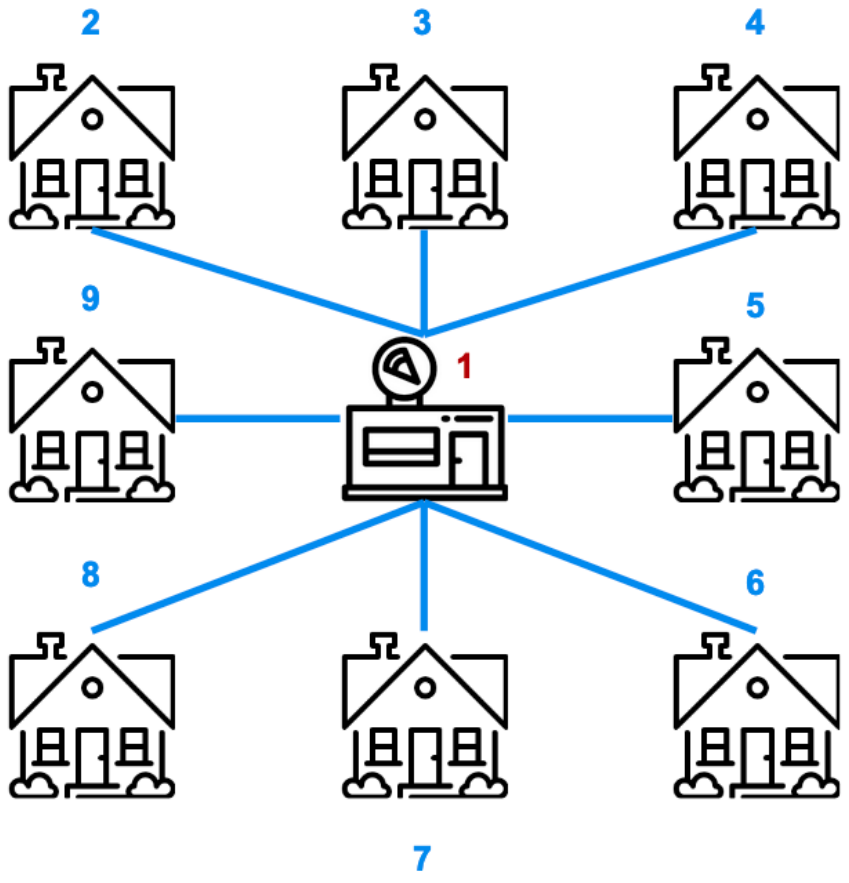
Tableau récapitulatif

Nombre de maisons	Ville A	Ville B	Ville C
15	14	7	3
1023	1022	511	9
n	n-1	$\frac{n-1}{2}$	

Tableau récapitulatif

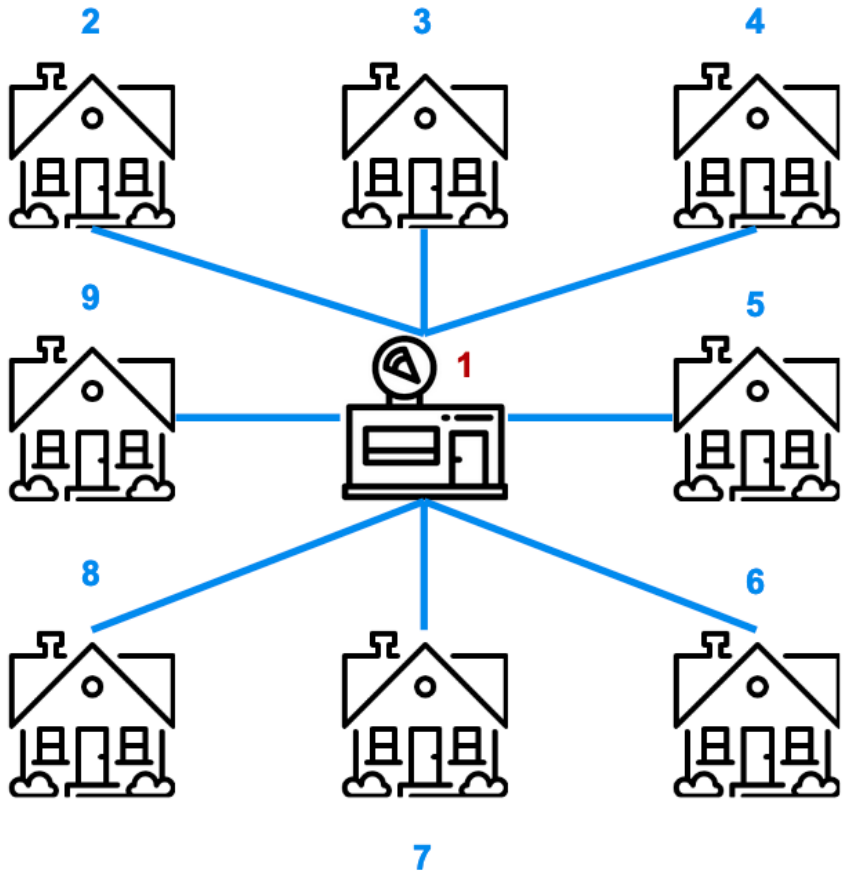
Nombre de maisons	Ville A	Ville B	Ville C
15	14	7	3
1023	1022	511	9
n	n-1	$\frac{n-1}{2}$	$\approx log_2(n)$

Et pourquoi pas en étoile ?



Cette configuration semble optimale, quel peut être le problème ?

Et pourquoi pas en étoile ?



Une organisation en étoile avec la pizzeria au milieu permet des trajets très courts, mais choisir la bonne rue prend du temps.

Complexité : quelques définitions

Complexité

La **complexité** dans le pire cas d'un algorithme est la fonction mathématique $\mathcal{T}$ qui donne le nombre maximal d'instructions élémentaires que l'algorithme effectue en fonction de la taille des données manipulées.

Exemple précédent

Dans notre exemple précédent, la taille des données manipulées étaient le nombre de maisons.

Complexité dans le pire cas : A quoi ça sert ?

- **Évaluer le temps d'exécution** d'un algorithme en fonction de la longueur de l'énoncé ;
- **Comparer les performances** de différents algorithmes résolvant le même problème ;
- **Évaluer la taille maximale** des énoncés qu'un algorithme peut traiter ;
- Mesure du temps **indépendante des machines** puisque l'on compte le nombre d'instructions.

Ordre de grandeur

Pour comparer des algorithmes, il n'est pas nécessaire d'utiliser la fonction $\mathcal{T}$, mais seulement l'ordre de grandeur asymptotique, noté $\mathcal{O}$ (prononcé "*grand O*").

Définition formelle

Une fonction $\mathcal{T}(n)$ est en $\mathcal{O}(f(n))$ ("*en grand O de f(n)*") si :

$$\exists n_o \in \mathbb{N}, \exists c \in \mathbb{R}^+, \forall n \in \mathbb{R}^+, n \geq n_0 \implies |T(n)| \leq c|f(n)|$$

Autrement dit, $\mathcal{T}(n)$ est en $\mathcal{O}(f(n))$ s'il existe un seuil n_0 à partir duquel la fonction $\mathcal{T}$ est toujours dominée par la fonction f, à une constance multiplicative fixée c.

Quelques exemples

Exemples

- $T_1(n) = 7 =$
- $T_2(n) = 12n + 5$
- $T_3(n) = 4n^2 + 2n + 6$
- $T_4(n) = 2 + (n - 1) \times 5$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) =$
- $T_2(n) = 12n + 5$
- $T_3(n) = 4n^2 + 2n + 6$
- $T_4(n) = 2 + (n - 1) \times 5$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5$
- $T_3(n) = 4n^2 + 2n + 6$
- $T_4(n) = 2 + (n - 1) \times 5$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5 = \mathcal{O}(12n)$
- $T_3(n) = 4n^2 + 2n + 6$
- $T_4(n) = 2 + (n - 1) \times 5$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5 = \mathcal{O}(12n) = \mathcal{O}(n)$
- $T_3(n) = 4n^2 + 2n + 6$
- $T_4(n) = 2 + (n - 1) \times 5$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5 = \mathcal{O}(12n) = \mathcal{O}(n)$
- $T_3(n) = 4n^2 + 2n + 6 = \mathcal{O}(4n^2 + 2n)$
- $T_4(n) = 2 + (n - 1) \times 5$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5 = \mathcal{O}(12n) = \mathcal{O}(n)$
- $T_3(n) = 4n^2 + 2n + 6 = \mathcal{O}(4n^2 + 2n) = \mathcal{O}(n^2)$
- $T_4(n) = 2 + (n - 1) \times 5$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5 = \mathcal{O}(12n) = \mathcal{O}(n)$
- $T_3(n) = 4n^2 + 2n + 6 = \mathcal{O}(4n^2 + 2n) = \mathcal{O}(n^2)$
- $T_4(n) = 2 + (n - 1) \times 5 = \mathcal{O}(n - 1)$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5 = \mathcal{O}(12n) = \mathcal{O}(n)$
- $T_3(n) = 4n^2 + 2n + 6 = \mathcal{O}(4n^2 + 2n) = \mathcal{O}(n^2)$
- $T_4(n) = 2 + (n-1) \times 5 = \mathcal{O}(n-1) = \mathcal{O}(n)$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5 = \mathcal{O}(12n) = \mathcal{O}(n)$
- $T_3(n) = 4n^2 + 2n + 6 = \mathcal{O}(4n^2 + 2n) = \mathcal{O}(n^2)$
- $T_4(n) = 2 + (n-1) \times 5 = \mathcal{O}(n-1) = \mathcal{O}(n)$

A vous de jouer

Quelle est la complexité dans le pire des cas des exemples suivants ?

- $T_5(n) = 3n + log(n)^4 + 2$
- $T_6(n) = 1024$
- $T_7(n) = 2^n + n^{10}$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5 = \mathcal{O}(12n) = \mathcal{O}(n)$
- $T_3(n) = 4n^2 + 2n + 6 = \mathcal{O}(4n^2 + 2n) = \mathcal{O}(n^2)$
- $T_4(n) = 2 + (n-1) \times 5 = \mathcal{O}(n-1) = \mathcal{O}(n)$

A vous de jouer

Quelle est la complexité dans le pire des cas des exemples suivants ?

- $T_5(n) = 3n + log(n)^4 + 2 = \mathcal{O}(n)$
- $T_6(n) = 1024$
- $T_7(n) = 2^n + n^{10}$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5 = \mathcal{O}(12n) = \mathcal{O}(n)$
- $T_3(n) = 4n^2 + 2n + 6 = \mathcal{O}(4n^2 + 2n) = \mathcal{O}(n^2)$
- $T_4(n) = 2 + (n-1) \times 5 = \mathcal{O}(n-1) = \mathcal{O}(n)$

A vous de jouer

Quelle est la complexité dans le pire des cas des exemples suivants ?

- $T_5(n) = 3n + log(n)^4 + 2 = \mathcal{O}(n)$
- $T_6(n) = 1024 = \mathcal{O}(1)$
- $T_7(n) = 2^n + n^{10}$

Quelques exemples

Exemples

- $T_1(n) = 7 = \mathcal{O}(7) = \mathcal{O}(1)$
- $T_2(n) = 12n + 5 = \mathcal{O}(12n) = \mathcal{O}(n)$
- $T_3(n) = 4n^2 + 2n + 6 = \mathcal{O}(4n^2 + 2n) = \mathcal{O}(n^2)$
- $T_4(n) = 2 + (n-1) \times 5 = \mathcal{O}(n-1) = \mathcal{O}(n)$

A vous de jouer

Quelle est la complexité dans le pire des cas des exemples suivants ?

- $T_5(n) = 3n + log(n)^4 + 2 = \mathcal{O}(n)$
- $T_6(n) = 1024 = \mathcal{O}(1)$
- $T_7(n) = 2^n + n^{10} = \mathcal{O}(2^n)$

Rappels des règles utiles

Les quelques règles suivantes permettent de simplifier les complexités en omettant des termes dominés :

- Les coefficients peuvent être omis : $14n^2$ devient n^2 ;
- n^a domine n^b si $a > b$: par exemple, n^2 domine n ;
- Une exponentielle domine un polynôme : $3^n = exp^{nlog3}$ domine n^5 ;
- De même un polynôme domine un logarithme : n domine $(logn)^3$. Cela signifie également que, par exemple, n^2 domine $nlogn$.

Reprenons notre pizzeria

Nombre de maisons	Ville A	Ville B	Ville C
15	14	7	3
1023	1022	511	9
n	n-1	$\frac{n-1}{2}$	$log_2(n)$

Reprenons notre pizzeria

Nombre de maisons	Ville A	Ville B	Ville C
15	14	7	3
1023	1022	511	9
n	n-1	$\frac{n-1}{2}$	$log_2(n)$
Complexité pour n	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(log(n))$

Reprenons notre pizzeria

Nombre de maisons	Ville A	Ville B	Ville C
15	14	7	3
1023	1022	511	9
n	n-1	$\frac{n-1}{2}$	$log_2(n)$
Complexité pour n	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(log(n))$

On voit que les complexités dans le pire cas de la ville A et de la ville B **sont les mêmes**

Reprenons notre exemple

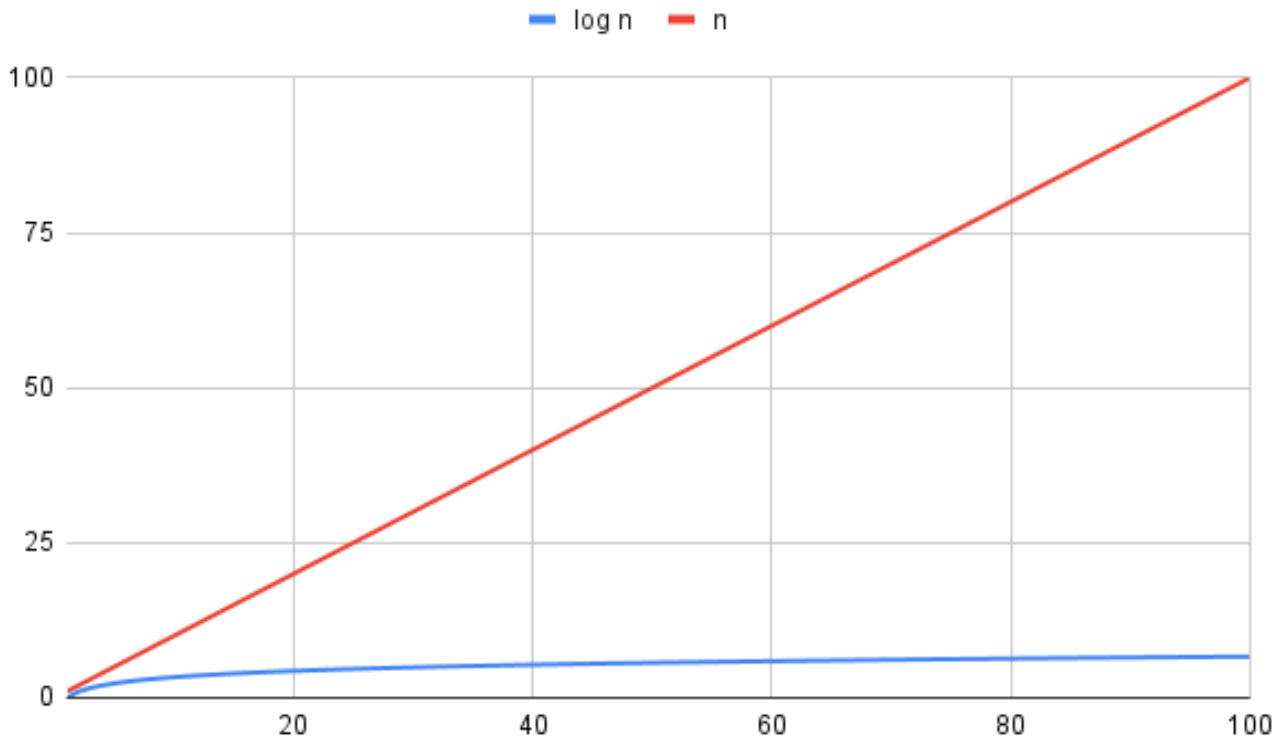
- Dans la ville A et B, l'algorithme naturel pour trouver une maison a une complexité linéaire $\mathcal{O}(n)$
- Dans la ville C, l'algorithme naturel pour trouver une maison a une complexité logarithmique $\mathcal{O}(log(n))$
- L'organisation de la ville C est beaucoup plus efficace pour livrer les pizzas.

Reprenons notre exemple

- Dans la ville A et B, l'algorithme naturel pour trouver une maison a une complexité linéaire $\mathcal{O}(n)$
- Dans la ville C, l'algorithme naturel pour trouver une maison a une complexité logarithmique $\mathcal{O}(log(n))$
- L'organisation de la ville C est beaucoup plus efficace pour livrer les pizzas.

À quel point est-ce que c'est plus efficace ?

Différence entre n et log n



- Si $n = 10^6$, alors $log_2(n) \approx 20$. Le livreur fait 50 000 fois moins de déplacements si les maisons sont organisées comme dans la ville C.

Classe de complexité

Complexité	Notation
constante	$\mathcal{O}(1)$
logarithmique	$\mathcal{O}(log(n))$
racinaire	$\mathcal{O}(\sqrt{x})$
linéaire	$\mathcal{O}(n)$
quasi-linéaire	$\mathcal{O}(nlog(n))$
quadratique (polynomial)	$\mathcal{O}(n^2)$
cubique (polynomial)	$\mathcal{O}(n^3)$
sous-exponentielle	$\mathcal{O}(2^{poly(log(n))})$
exponentielle	$\mathcal{O}(2^{poly(n)})$)
factorielle	$\mathcal{O}(n!)$

Table 1: Évolution du temps de calcul en fonction de la complexité d'un algorithme et de la taille des données

Il existe d'autres complexités

Complexité en moyenne

- Il s'agit de la complexité en moyenne sur toutes les entrées ;
- Intéressant car certains problèmes ont une complexité dans le pire cas très élevées, mais les entrées qui correspondent à ces cas ne se produisent que très rarement en pratique ;
- Étudié essentiellement dans les algorithmes de tri (que nous verrons plus tard)

Complexité en mémoire

- On souhaite ici mesurer la mémoire utilisée par un algorithme ;
- Certains algorithmes sont en effet très rapide mais prennent beaucoup d'espace mémoire (ou inversement) ;

Il existe plusieurs autres complexités mais qui ne nous concernent pas directement.

Langage de Description Algorithmique

Plus grand diviseur commun (pgcd)

Problème :

- Entrées : 2 entiers $a > b \geq 0$
- Sortie : le pgcd de a et b

Exemple : $pgcd(123, 82) = 41$

Plus grand diviseur commun (pgcd)

Problème :

- Entrées : 2 entiers $a > b \geq 0$
- Sortie : le pgcd de a et b

Exemple : $pgcd(123, 82) = 41$

Opérations élémentaires : $+, \times, -$ et division euclidienne

Plus grand diviseur commun (pgcd)

Problème :

- Entrées : 2 entiers $a > b \geq 0$
- Sortie : le pgcd de a et b

Exemple : $pgcd(123, 82) = 41$

Opérations élémentaires : $+, \times, -$ et division euclidienne

Observation : $\begin{cases} pgcd(a,b) & = pgcd(b,r) \quad \text{r reste de la div. de a par b} \\ pgcd(a,0) & = a \end{cases}$

Exemple : $pgcd(123, 82) = pgcd(82, 41) = pgcd(41, 0) = 41$

Pgcd : écriture d' un algorithme

Observation : $\begin{cases} pgcd(a,b) & = pgcd(b,r) \quad \text{r reste de la div. de a par b} \\ pgcd(a,0) & = a \end{cases}$

Algorithme : ? ? ?

Pgcd : écriture d' un algorithme

Observation : $\begin{cases} pgcd(a,b) & = pgcd(b,r) \quad \text{r reste de la div. de a par b} \\ pgcd(a,0) & = a \end{cases}$

Algorithme :

si $b = 0$ **retourner** a **fin si**

Pgcd : écriture d' un algorithme

Observation : $\begin{cases} pgcd(a,b) & = pgcd(b,r) \quad \text{r reste de la div. de a par b} \\ pgcd(a,0) & = a \end{cases}$

Algorithme :

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

$r \leftarrow$ reste de la div. euclidienne de x par y

Pgcd : écriture d' un algorithme

Observation : $\begin{cases} pgcd(a,b) & = pgcd(b,r) \quad \text{r reste de la div. de a par b} \\ pgcd(a,0) & = a \end{cases}$

Algorithme :

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

$r \leftarrow$ reste de la div. euclidienne de x par y

si $r = 0$ **alors retourner** y

Pgcd : écriture d' un algorithme

Observation : $\begin{cases} pgcd(a,b) & = pgcd(b,r) \quad \text{r reste de la div. de a par b} \\ pgcd(a,0) & = a \end{cases}$

Algorithme :

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

$r \leftarrow$ reste de la div. euclidienne de x par y

si $r = 0$ **alors retourner** y

sinon $x \leftarrow y$; $y \leftarrow r$

fin si

Pgcd : écriture d' un algorithme

Observation : $\begin{cases} pgcd(a,b) & = pgcd(b,r) \quad \text{r reste de la div. de a par b} \\ pgcd(a,0) & = a \end{cases}$

Algorithme :

si $b = 0$ **retourner** a **fin si**
$x \leftarrow a; y \leftarrow b$
répéter
    $r \leftarrow$ reste de la div. euclidienne de x par y
    si $r = 0$ **alors retourner** y
    sinon $x \leftarrow y$; $y \leftarrow r$
    fin si
fin répéter

Pgcd : écriture d' un algorithme

Observation : $\begin{cases} pgcd(a,b) & = pgcd(b,r) \quad \text{r reste de la div. de a par b} \\ pgcd(a,0) & = a \end{cases}$

Algorithme :

si $b = 0$ **retourner** a **fin si**
$x \leftarrow a; y \leftarrow b$
répéter
    $r \leftarrow$ reste de la div. euclidienne de x par y
    si $r = 0$ **alors retourner** y
    sinon $x \leftarrow y$; $y \leftarrow r$
    fin si
fin répéter

Pgcd : écriture d' un algorithme

Observation : $\begin{cases} pgcd(a,b) & = pgcd(b,r) \quad \text{r reste de la div. de a par b} \\ pgcd(a,0) & = a \end{cases}$

Algorithme :

si $b = 0$ **retourner** a **fin si**
$x \leftarrow a; y \leftarrow b$
répéter
    $r \leftarrow$ reste de la div. euclidienne de x par y
    si $r = 0$ **alors retourner** y
    sinon $x \leftarrow y$; $y \leftarrow r$
    fin si
fin répéter

Pgcd : écriture d' un algorithme

Observation : $\begin{cases} pgcd(a,b) = pgcd(b,r) & \text{r reste de la div. de a par b} \\ pgcd(a,0) = a \end{cases}$

Algorithme :

pgcd(a,b) `/* a,b entiers,` $a > b \geq 0$ `*/`

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a$; $y \leftarrow b$

répéter

$r \leftarrow$ reste de la div. euclidienne de x par y

si $r = 0$ **alors retourner** y

sinon $x \leftarrow y$; $y \leftarrow r$

fin si

fin répéter

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

$r \leftarrow x \ \% \ y$

$x \leftarrow y$

$y \leftarrow r$

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

$r \leftarrow x \ \% \ y$

$x \leftarrow y$

$y \leftarrow r$

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

r ← x % y

$x \leftarrow y$

$y \leftarrow r$

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78		

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

$r \leftarrow x \ \% \ y$

$x \leftarrow y$

$y \leftarrow r$

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

$r \leftarrow x \ \% \ y$

$x \leftarrow y$

y ← r

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78

Algorithme d'Euclide

pgcd(a,b)
    entier x, y, r
     si $b = 0$ **retourner** a **fin si**
     $x \leftarrow a; y \leftarrow b$
    tant que $y \neq 0$ **faire**
        r ← x % y
         $x \leftarrow y$
         $y \leftarrow r$
    fin tant que
    retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60		

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

$r \leftarrow x \ \% \ y$

$x \leftarrow y$

$y \leftarrow r$

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	

Algorithme d'Euclide

pgcd(a,b)
    entier x, y, r
     si $b = 0$ **retourner** a **fin si**
     $x \leftarrow a; y \leftarrow b$
    tant que $y \neq 0$ **faire**
        $r \leftarrow x \ \% \ y$
        $x \leftarrow y$
        y ← r
    fin tant que
    retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60

Algorithme d'Euclide

```
pgcd(a,b)
    entier x, y, r
     si b = 0 retourner a fin si
     x ← a; y ← b
    tant que y ≠ 0 faire
        r ← x % y
         x ← y
         y ← r
    fin tant que
    retourner(x)
```

pgcd(12345678,123456)

r	x	y
⊥	12345678	123456
78	123456	78
60	78	60
18		

Algorithme d'Euclide

pgcd(a,b)
    entier x, y, r
     si $b = 0$ **retourner** a **fin si**
     $x \leftarrow a; y \leftarrow b$
    tant que $y \neq 0$ **faire**
        $r \leftarrow x \ \% \ y$
        x ← y
        $y \leftarrow r$
    fin tant que
    retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60
18	60	

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

$r \leftarrow x \ \% \ y$

$x \leftarrow y$

y ← r

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60
18	60	18

Algorithme d'Euclide

pgcd(a,b)
    entier x, y, r
     si $b = 0$ **retourner** a **fin si**
     $x \leftarrow a; y \leftarrow b$
    tant que $y \neq 0$ **faire**
        r ← x % y
         $x \leftarrow y$
         $y \leftarrow r$
    fin tant que
    retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60
18	60	18
6		

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

$r \leftarrow x \ \% \ y$

$x \leftarrow y$

$y \leftarrow r$

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60
18	60	18
6	18	

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

$r \leftarrow x \ \% \ y$

$x \leftarrow y$

y ← r $y \leftarrow r$

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60
18	60	18
6	18	6

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

r ← x % y

$x \leftarrow y$

$y \leftarrow r$

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60
18	60	18
6	18	6
0		

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

$r \leftarrow x \ \% \ y$

x ← y

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60
18	60	18
6	18	6
0	6	

Algorithme d'Euclide

pgcd(a,b)

entier x, y, r

si $b = 0$ **retourner** a **fin si**

$x \leftarrow a; y \leftarrow b$

tant que $y \neq 0$ **faire**

$r \leftarrow x \ \% \ y$

$x \leftarrow y$

y ← r

fin tant que

retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60
18	60	18
6	18	6
0	6	0

Algorithme d'Euclide

pgcd(a,b)
    entier x, y, r
     si $b = 0$ **retourner** a **fin si**
     $x \leftarrow a; y \leftarrow b$
    tant que $y \neq 0$ **faire**
        $r \leftarrow x \ \% \ y$
        $x \leftarrow y$
        $y \leftarrow r$
    fin tant que
    retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60
18	60	18
6	18	6
0	6	0

Algorithme d'Euclide

pgcd(a,b)
    entier x, y, r
     si $b = 0$ **retourner** a **fin si**
     $x \leftarrow a; y \leftarrow b$
    tant que $y \neq 0$ **faire**
        $r \leftarrow x \ \% \ y$
        $x \leftarrow y$
        $y \leftarrow r$
    fin tant que
    retourner(x)

pgcd(12345678,123456)

r	x	y
$\perp$	12345678	123456
78	123456	78
60	78	60
18	60	18
6	18	6
0	**6**	0