LINFO 1121

DATA STRUCTURES AND ALGORITHMS

UCLouvain

TP1: Complexités, piles, Files et listes

Question 1.1.1: Qu’est-ce qu’un type abstrait de données?

En informatique, un **type abstrait** est une spécification mathématique d'un ensemble de **données** et de l'ensemble des opérations qu'on peut effectuer sur elles. On qualifie d'**abstrait** ce **type** de **données** car il correspond à un cahier des charges qu'une structure de **données** doit ensuite implémenter.

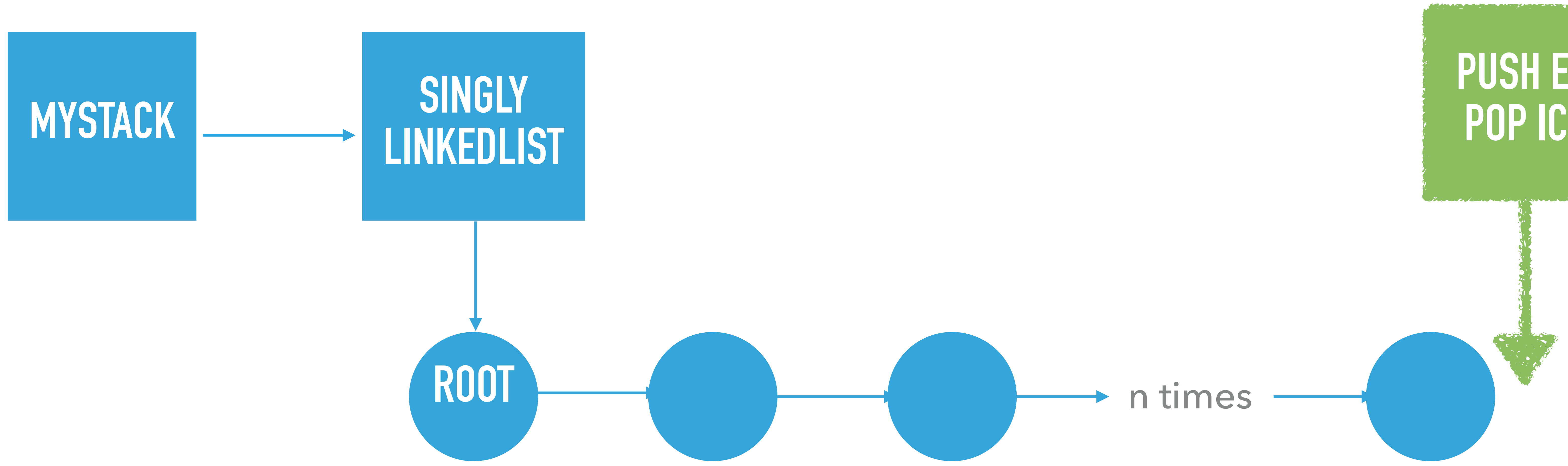
Question 1.1.1: Qu’est-ce qu’un type abstrait de données?

DE QUOI SE RAPPROCHE LE PLUS UN TYPE ABSTRAIT DE DONNÉE EN JAVA?

- Une classe
- Un objet
- Une interface
- Une structure
- Un type primitif

Question 1.1.2

Comment implémenter une pile (stack) avec une liste simplement chaînée, où les opérations se font en fin de liste?



Question 1.1.2

QUELLE EST LA COMPLEXITÉ DE CETTE IMPLÉMENTATION?

$\mathcal{O}(1)$	~ 1	$\Omega(1)$	$\Theta(1)$
$\mathcal{O}(n)$	$\sim n$	$\Omega(n)$	$\Theta(n)$

QUESTION 1.1.3: QUELLES SONT LES IMPLÉMENTATIONS POSSIBLES POUR UNE PILE?

Question 1.1.3 (suite)

Pourquoi java.util.Stack est-il implémenté via un tableau?

	Liste chainée	Tableau
Push	O(1)	O(n) ?
Pop	O(1)	O(n) ?
n push	O(n)	O(n^2) ?
n pop	O(n)	O(n^2) ?

Question 1.1.3 (suite)

Pourquoi java.util.Stack est-il implémenté via un tableau?

	Liste chainée	Tableau
Push	O(1)	O(1) Sauf redim. O(n)
Pop	O(1)	O(1) Sauf redim. O(n)
n push	O(n)	O(n^2) ?
n pop	O(n)	O(n^2) ?

On double la taille du tableau quand on atteint la limite.

Sur n push, on fait donc un resize à ces positions:

- 1
- 2
- 4
- 8
- 16
- …
- n

Combien de redim.?

$$\sum_{i=0}^{\log_2 n} \frac{n}{2^i}$$

Question 1.1.3 (suite)

Pourquoi java.util.Stack est-il implémenté via un tableau?

	Liste chainée	Tableau
Push	O(1)	O(1) Sauf redim. O(n)
Pop	O(1)	O(1) Sauf redim. O(n)
n push	O(n)	O(n^2) ?
n pop	O(n)	O(n^2) ?

On double la taille du tableau quand on atteint la limite.

Sur n push, on fait donc un resize à ces positions:

- 1
- 2
- 4
- 8
- 16
- …
- n

Combien de redim.?

$$\sum_{i=0}^{\log_2 n} \frac{n}{2^i} < 2n \in \mathcal{O}(n)$$

Question 1.1.3 (suite)

Pourquoi java.util.Stack est-il implémenté via un tableau?

	Liste chainée	Tableau
Push	O(1)	O(1) Sauf redim. O(n)
Pop	O(1)	O(1) Sauf redim. O(n)
n push	O(n)	O(n)
n pop	O(n)	O(n)

On double la taille du tableau quand on atteint la limite.

Sur n push, on fait donc un resize à ces positions:

- 1
- 2
- 4
- 8
- 16
- ...
- n

COMPLEXITÉS AMORTIES

Combien de redim.?

$$\sum_{i=0}^{\log_2 n} \frac{n}{2^i} < 2n \in \mathcal{O}(n)$$

Question 1.1.3 (suite)

Pourquoi java.util.Stack est-il implémenté via un tableau?

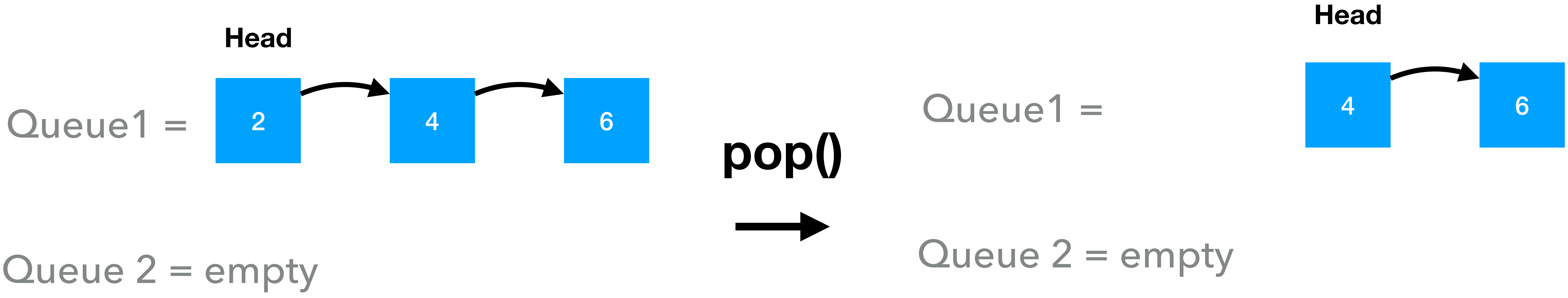
	Liste chainée	Tableau
Push	O(1)	O(1) Sauf redim. O(n)
Pop	O(1)	O(1) Sauf redim. O(n)
n push	O(n)	O(n) (amorti)
n pop	O(n)	O(n) (amorti)

Même complexités, pourquoi ce choix d'implémentation?

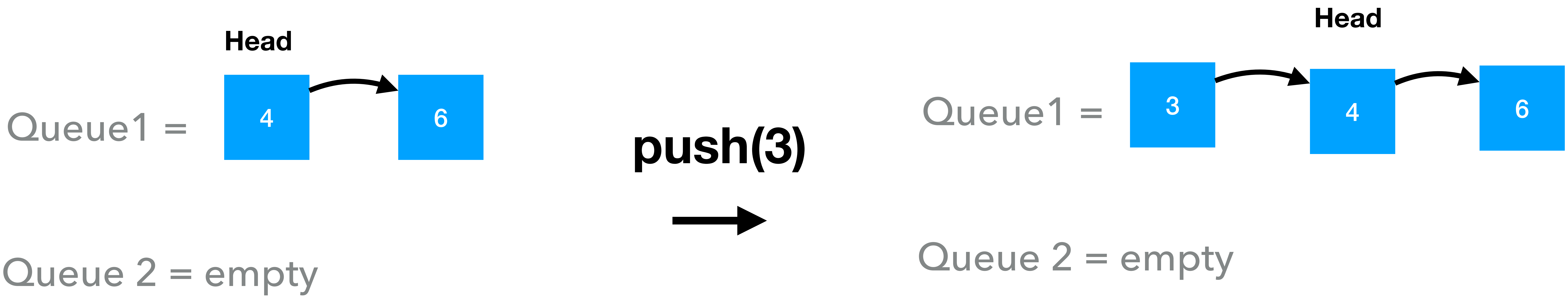
Question 1.1.4: Pile avec deux files

Queue API (FIFO)	Stack API (LIFO)
enqueue(e)	*push(e)*
E dequeue/remove()	*E pop()*

Question 1.1.4: Pile avec deux files



Question 1.1.4: Pile avec deux files



Question 1.1.4: Pile avec deux files

```java
Queue<E> queue1;
Queue<E> queue2; // always empty

public E pop() throws EmptyStackException {
    if (empty()) throw new EmptyStackException();
    return queue1.remove();
}

public void push(E item) {
    queue2.add(item);
    while (!queue1.isEmpty()) {
        queue2.add(queue1.remove());
    }
    Queue<E> buffer = queue1;
    queue1 = queue2;
    queue2 = buffer;
}
```

Question 1.1.5: Est-ce équivalent en terme de complexité ?

```
LinkedList<Integer> list = new LinkedList<>();

// assume I insert n elements in the list here

for (Integer val: list) {
    System.out.println(val);
}

Iterator<Integer> itr = list.iterator();
while (itr.hasNext()) {
    Integer val = itr.next();
    System.out.println(val);
}

for (int i = 0; i < list.size(); i++) {
    Integer val = list.get(i);
    System.out.println(val);
}
```

Question 1.1.6: TYPES de complexités

Le livre utilise la notation ~ (tilde). On utilise dans d'autres cours les notations O, Omega et Theta.

Quelle est la différence entre ces différentes classes de complexités?

Question 1.1.6: types de complexités

$$f(n) \in \mathcal{O}(g(n)) \quad \Longleftrightarrow \quad \exists k \in \mathbb{R}^+, n_0 \in \mathbb{N} \quad \text{t.q.} \quad f(n) \leq k \cdot g(n) \quad \forall n \geq n_0$$

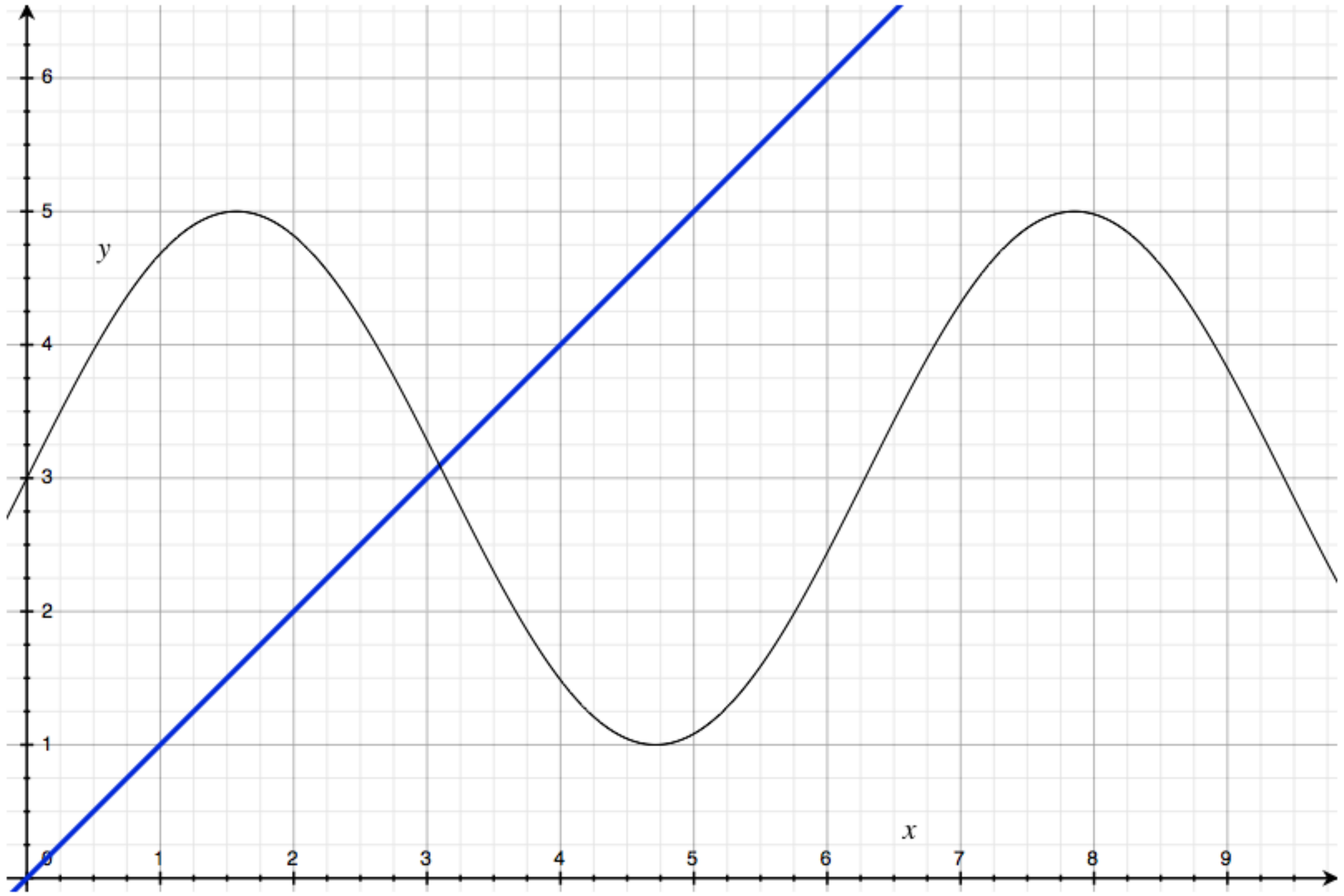
« Appartient »
O() est donc un ensemble

Il existe un k tel que k*g(n) est plus grand que f(n) quand n est grand

Ceci est une fonction qui compte le nombre d'opérations en fonction de n

Question 1.1.6: types de complexités

$$f(n) \in \mathcal{O}(g(n)) \quad \Longleftrightarrow \quad \exists k \in \mathbb{R}^+, n_0 \in \mathbb{N} \quad \textbf{t.q.} \quad f(n) \leq k \cdot g(n) \quad \forall n \geq n_0$$



$$f(x) = 2\sin(x) + 2$$

$$g(x) = x$$

$$2\sin(x) + 2 \in \mathcal{O}(x)$$

Question 1.1.6: types de complexités

$$f(n) \in \mathcal{O}(g(n)) \quad \Longleftrightarrow \quad \exists k \in \mathbb{R}^+, n_0 \in \mathbb{N} \quad \textbf{t.q.} \quad f(n) \leq k \cdot g(n) \quad \forall n \geq n_0$$

$$f(n) \in \boldsymbol{\Omega}(g(n)) \quad \Longleftrightarrow \quad \exists k \in \mathbb{R}^+, n_0 \in \mathbb{N} \quad \textbf{t.q.} \quad \mathbf{k \cdot f(n) \geq g(n)} \quad \forall n \geq n_0$$

$$f(n) \in \boldsymbol{\Theta}(g(n)) \quad \Longleftrightarrow \quad \exists k_0, k_1 \in \mathbb{R}^+, n_0 \in \mathbb{N} \quad \textbf{t.q.} \quad \mathbf{k_0 \cdot g(n) \leq f(n) \leq k_1 \cdot g(n)} \quad \forall n \geq n_0$$

Question 1.1.6: types de complexités

$$f(n) \in \mathcal{O}(g(n)) \quad \Longleftrightarrow \quad \exists k \in \mathbb{R}^+, n_0 \in \mathbb{N} \quad \textbf{t.q.} \quad f(n) \leq k \cdot g(n) \quad \forall n \geq n_0$$

$$f(n) \in \boldsymbol{\Omega}(g(n)) \quad \Longleftrightarrow \quad \exists k \in \mathbb{R}^+, n_0 \in \mathbb{N} \quad \textbf{t.q.} \quad \mathbf{k} \cdot \mathbf{f}(\mathbf{n}) \geq \mathbf{g}(\mathbf{n}) \quad \forall n \geq n_0$$

$$f(n) \in \boldsymbol{\Theta}(g(n)) \quad \Longleftrightarrow \quad \exists k_0, k_1 \in \mathbb{R}^+, n_0 \in \mathbb{N} \quad \textbf{t.q.} \quad \mathbf{k_0} \cdot \mathbf{g}(\mathbf{n}) \leq \mathbf{f}(\mathbf{n}) \leq \mathbf{k_1} \cdot \mathbf{g}(\mathbf{n}) \quad \forall n \geq n_0$$

$$f(n) \sim g(n) \quad \Longleftrightarrow \quad \lim_{n \to \infty} \frac{f(n)}{g(n)} = 1$$

Question 1.1.7: Doubling RATIO TEST

Si $f(n) \sim an^b \log n$ alors $\dfrac{f(2n)}{f(n)} \sim 2^b$

Question 1.1.7: Doubling RATIO TEST

n	1000	2000	4000	8000	16000	32000	64000
T(n)	0	0	0.1	0.3	1.3	5.1	20.5
Ratio précédent		~1	~1	~3	4.3	3.92	4.01