Science informatique

CM1

Antonio E. Porreca
aeporreca.org/scienceinfo

Équipe enseignante

- Responsable de l'UE : c'est moi
  - Antonio E. Porreca 👉 antonio.porreca@univ-amu.fr
    - L'accent drôle est italien 💪
- Chargé de TD :
  - Antonio E. Porreca, c'est toujours moi

⚠️ Les TD sont intégrés aux CM ! ⚠️

Deux UE complémentaires

- Science informatique
  - Remettre à plat et approfondir vos connaissances informatiques au travers d'exemples et de mises en situation théoriques et pratiques, en « mode débranché »
- Projet informatique
  - Développer des compétences pratiques en algorithmiques et programmation, sur ordinateur en langage Python, avec un travail en groupe à visée applicative

On va apprendre 😊

- Concevoir le traitement informatisé d'informations de différentes natures : texte, nombres, images

- Modéliser un problème concret sous la forme d'un problème algorithmique connu

- Évaluer l'efficacité et la correction d'une solution algorithmique

- Être familiarisé avec les concepts fondamentaux de complexité et de calculabilité

On ne va pas apprendre ☹️

- Utiliser des logiciels bureautiques
- Installer Linux sur l’ordinateur
- Faire du web design
- Jouer aux ou réaliser des jeux vidéos
- Sécurité informatique

Format de l'UE

- 30h de cours et travaux dirigés
  - 10 semaines à 3h
- Travail personnel
  - prise de notes pendant les cours
  - travail entre les séances pour préparer/finir les TD
  - lecture des notes de cours
  - courts quiz sur AMeTICE

Evaluation

- Évaluation en 1e session :

$$\max(0{,}3 \times \text{partiel} + 0{,}7 \times \text{ex-terminal}, \text{ex-terminal})$$

  - Partiel de 1h30, sans documents ni calculatrice
  - Examen terminal de 2h sans document ni calculatrice
- Évaluation en 2e session (rattrapage, en juin) :

$$1{,}0 \times \text{examen-terminal}$$

Algorithmes !

C’est quoi un algorithme ?

- La description non ambiguë d’une séquence finie d’instructions permettant de résoudre un problème

- Finitude = termine après un nombre fini d’étapes

- Non ambigu = précis (en termes d’opérations élémentaires)

- Entrées = données

- Sorties = résultat attendu

محمد بن موسى الخوارزمي

Muḥammad ibn Mūsā al-Khwārizmī

Résoudre un problème

- On cherche un algorithme
- On le décrit précisément, de manière non ambigüe
- On prouve qu'il est correct
- On vérifie qu'il est efficace (idéalement, on choisit l'algorithme optimal)
- On le met en œuvre (pas dans cette UE)
- On le teste (pas dans cette UE)

Décrire des algorithmes

- En langage naturel (par exemple, en français avec un accent italien)

- En pseudocode (semi-formel)

- En langage de programmation (formel)

    - Par exemple, en Python comme ici et dans l'UE Projet informatique

Recherche dans une séquence en langage naturel

- Pour chaque élément de la séquence à partir du premier :
    - Si cet élément est l'élément cherché, on a terminé
    - Sinon, on continue avec l'élément suivant
- S'il n'y a plus d'éléments et on n'a pas trouvé ce qu'on cherchait, alors il n'est pas là

Recherche dans une séquence en pseudocode

```
fonction chercher(élément, séquence)
    n ≔ longueur(séquence)
    i ≔ 0
    tant que i < n faire
        si séquence[i] = élément alors
            retourner i
        fin si
        i ≔ i + 1
    fin tant que
    retourner –1
fin fonction
```

Recherche dans une séquence en Python 🐍

```
def chercher(element, sequence):
    n = len(sequence)
    i = 0
    while i < n:
        if sequence[i] == element:
            return i
        i = i + 1
    return -1
```

Structures de contrôle

```
instruction1
instruction2
…
instructionn
```

```
if condition:
    instructions
else:
    d'autres instructions
```

```
while condition:
    instructions
```

Exécution d’un algorithme et affectations des variables

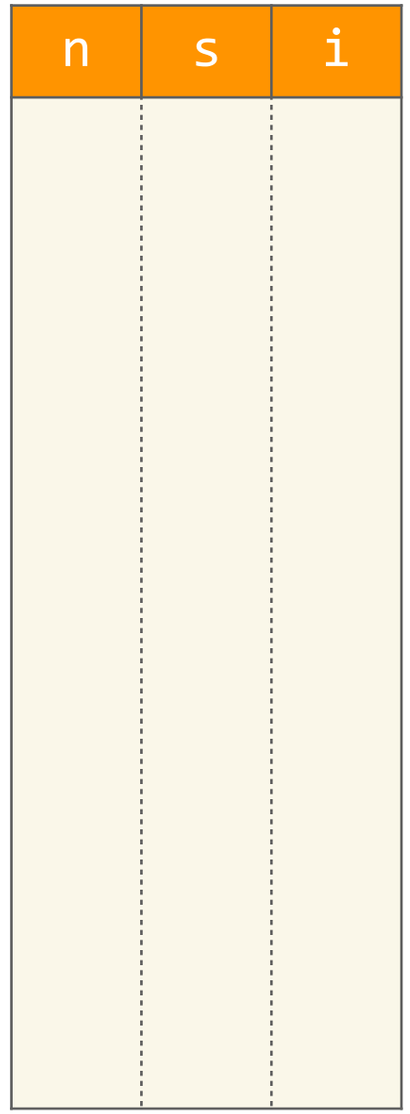
```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s
```

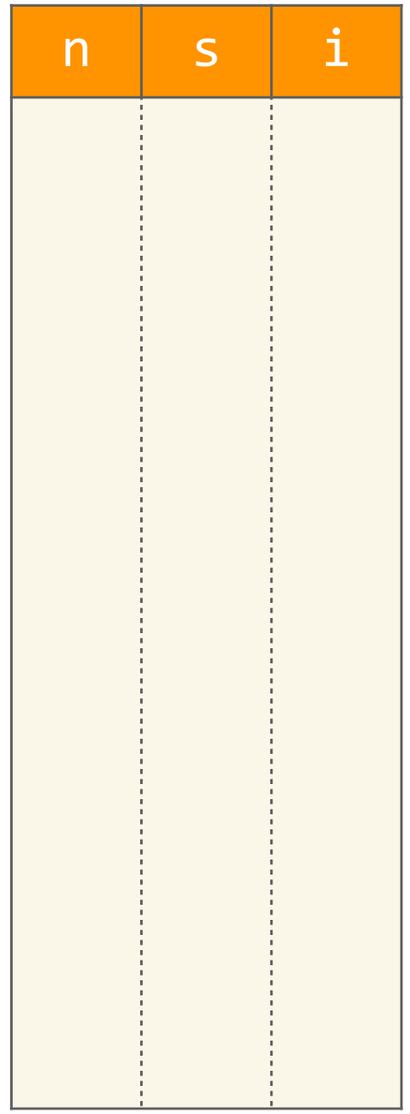
👉 `>>> somme_multiples_3_ou_5(10)`



Exécution d'un algorithme et affectations des variables

```
👉 def somme_multiples_3_ou_5(n):
       s = 0
       for i in range(1, n + 1):
           if i % 3 == 0 or i % 5 == 0:
               s = s + i
       return s

>>> somme_multiples_3_ou_5(10)
```



Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10		

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
👉      if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1 2

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
		3

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

👉

n	s	i
10	0	1
		2
		3

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
		5

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

👉

n	s	i
10	0	1
		2
	3	3
		4
		5

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
		6

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
👉          s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
		6

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
👉      if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
👉      if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
👉      if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8
		9

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
👉          s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8
		9

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8
	23	9

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
👉      if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8
	23	9
		10

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
👉          s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8
	23	9
		10

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
👉  for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8
	23	9
	33	10

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
👉      return s

>>> somme_multiples_3_ou_5(10)
```

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8
	23	9
	33	10

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s
```

👉 `>>> somme_multiples_3_ou_5(10)`

n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8
	23	9
	33	10

Exécution d'un algorithme et affectations des variables

```
def somme_multiples_3_ou_5(n):
    s = 0
    for i in range(1, n + 1):
        if i % 3 == 0 or i % 5 == 0:
            s = s + i
    return s

>>> somme_multiples_3_ou_5(10)
👉 33
```

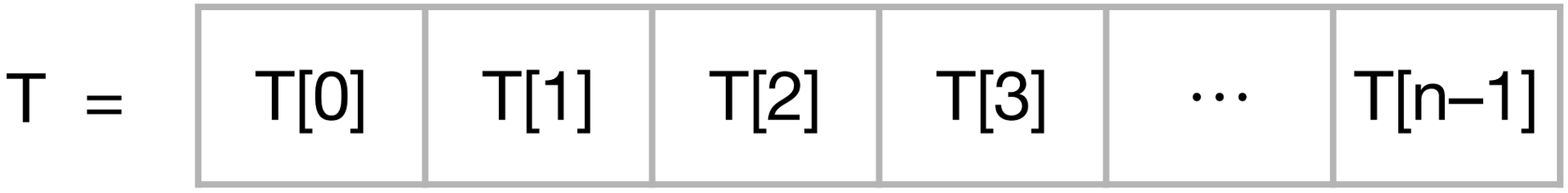
n	s	i
10	0	1
		2
	3	3
		4
	8	5
	14	6
		7
		8
	23	9
	33	10

Exercice 1 du TD1

Structures de données

- Variables entières
- Tableaux
- Matrices
- Arbres

Structures de données : les tableaux



Structures de données : les tableaux

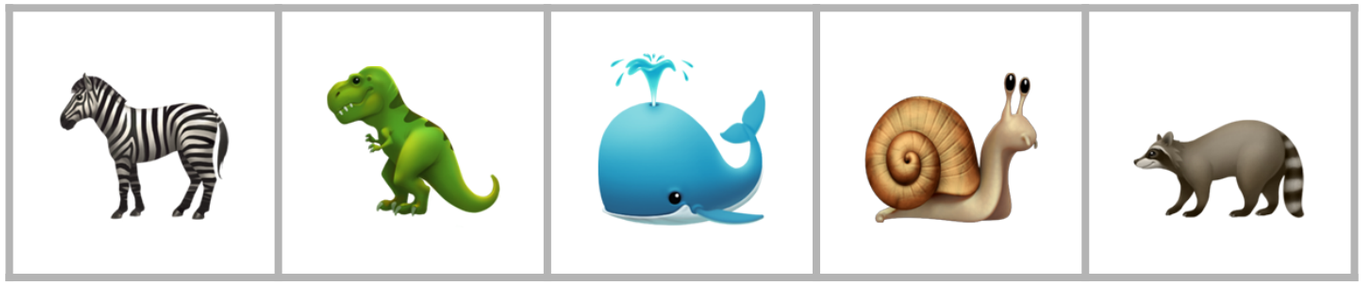
T =

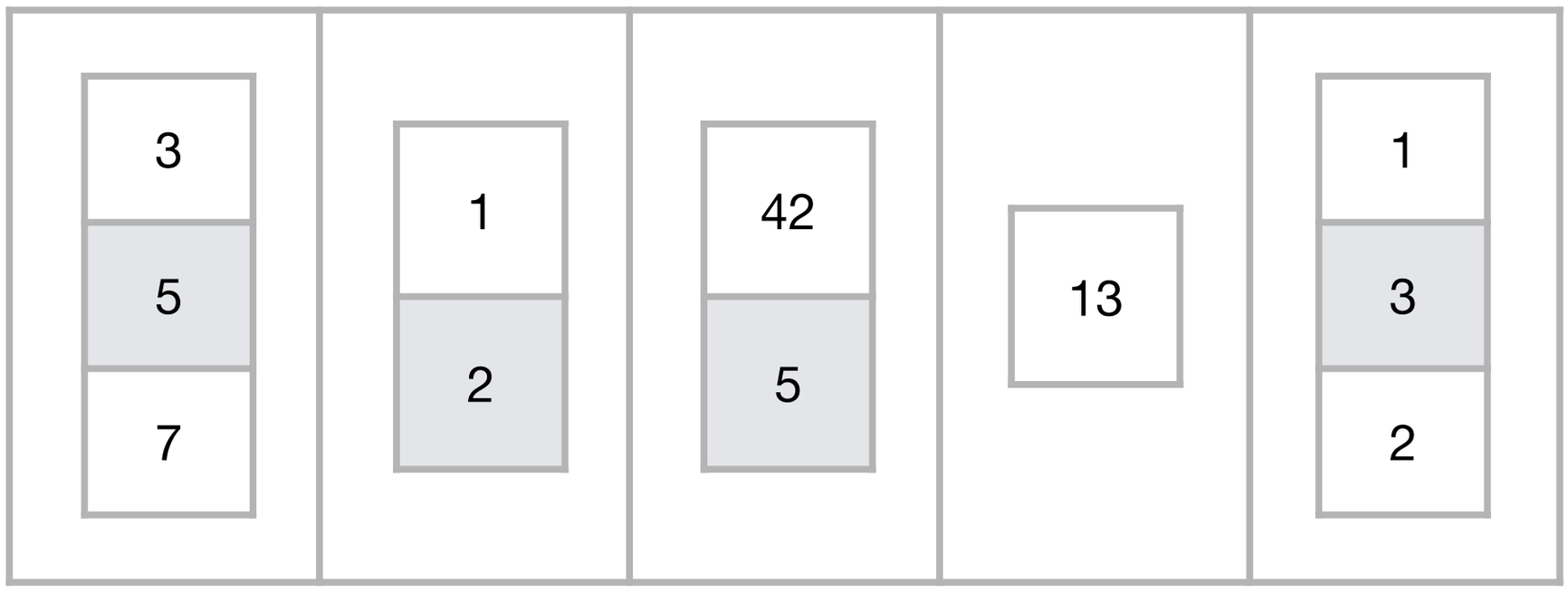
T[0]	T[1]	T[2]	T[3]	⋯	T[n–1]

3	4	2	3	1

I	N	F	O	R	M	A	T	I	Q	U	E

Tableaux





Exercice 2 du TD1

Comment faire une recherche dans un dictionnaire ? 📖

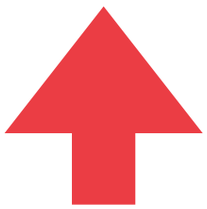
Recherche dans un tableau

```
def rechercher_séquentiel(t, e):
    n = len(t)
    for i in range(n):
        if t[i] == e:
            return i
    return -1
```

Recherche linéaire

Recherche de 33

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11



i

Recherche linéaire

Recherche de 33

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11

i

Recherche linéaire

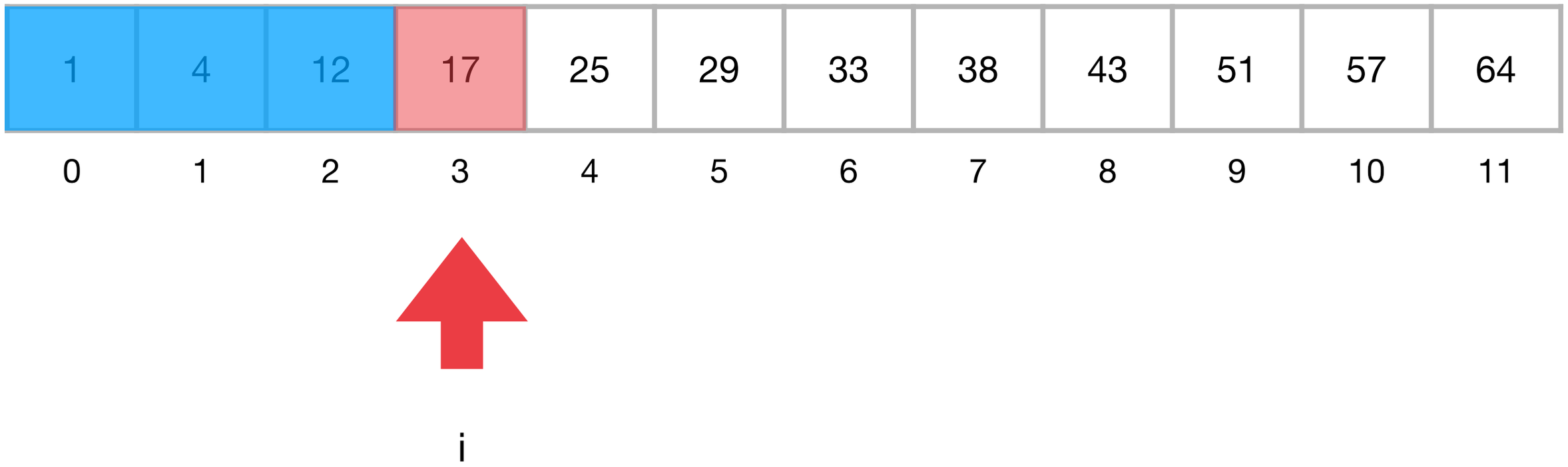
Recherche de 33

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11

i

Recherche linéaire

Recherche de 33



Recherche linéaire

Recherche de 33

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11

i

Recherche linéaire

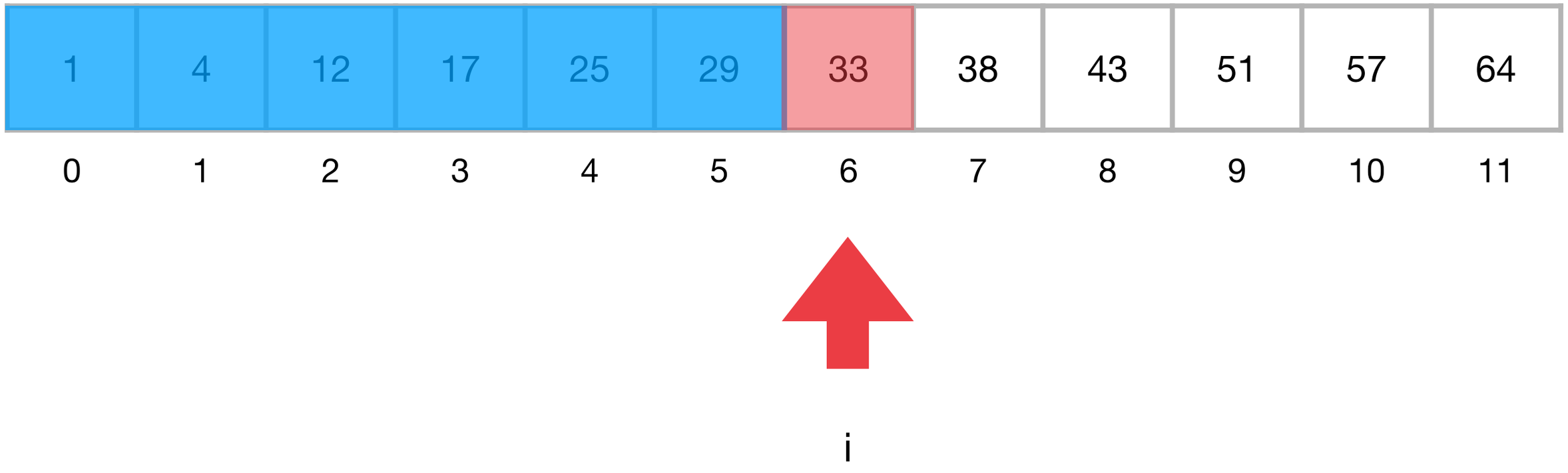
Recherche de 33

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11

i

Recherche linéaire

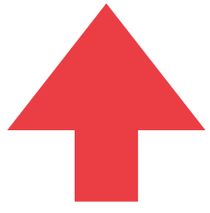
Recherche de 33



Recherche linéaire

Recherche de 33

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11

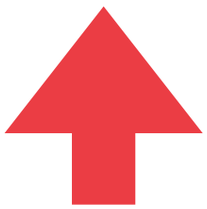


i

Recherche linéaire

Recherche de 3

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11



i

Recherche linéaire

Recherche de 3

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11

i

Recherche linéaire

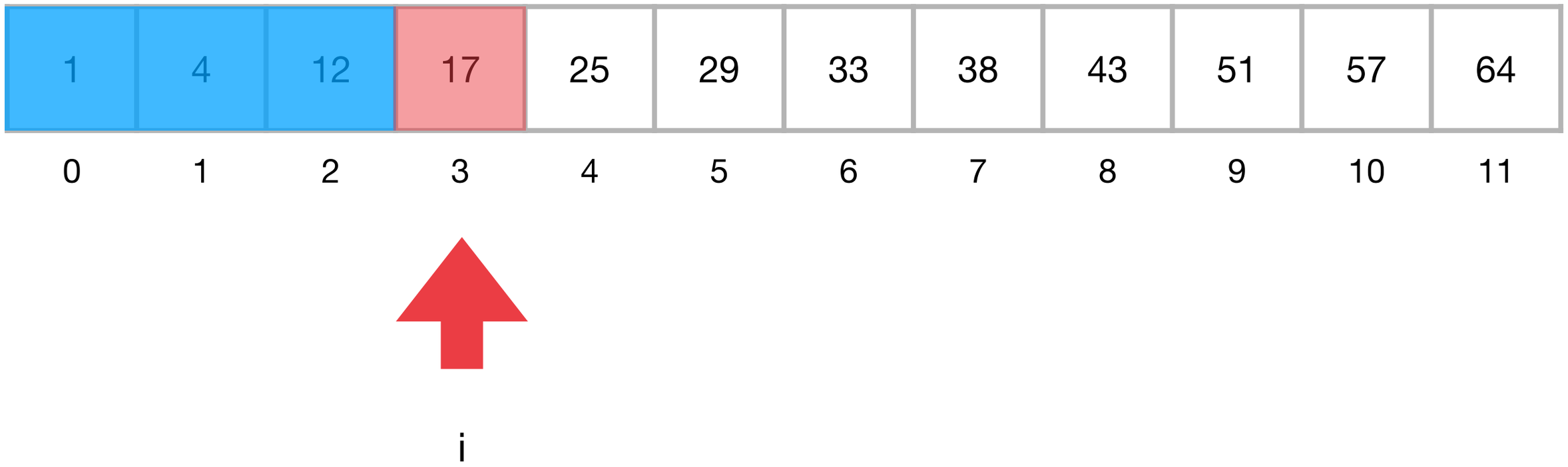
Recherche de 3

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11

i

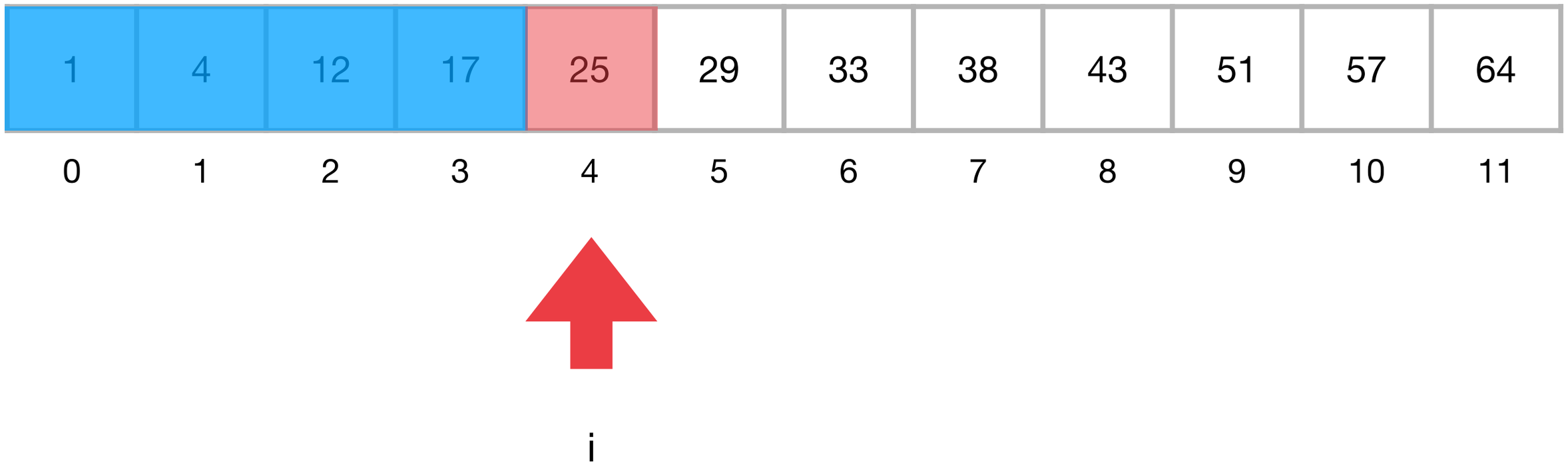
Recherche linéaire

Recherche de 3



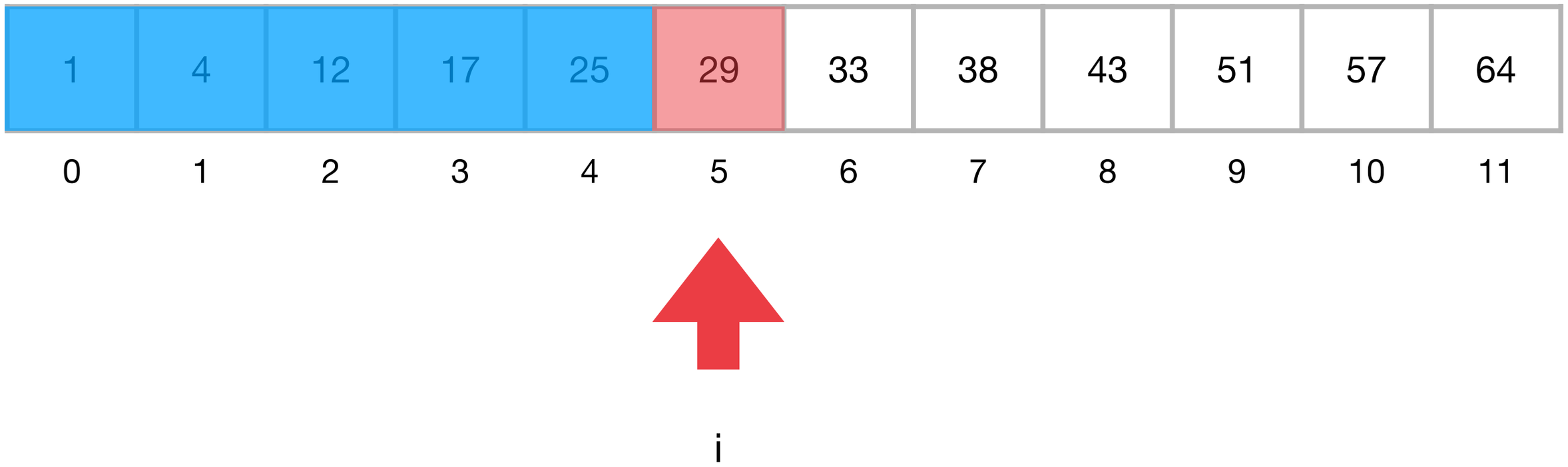
Recherche linéaire

Recherche de 3



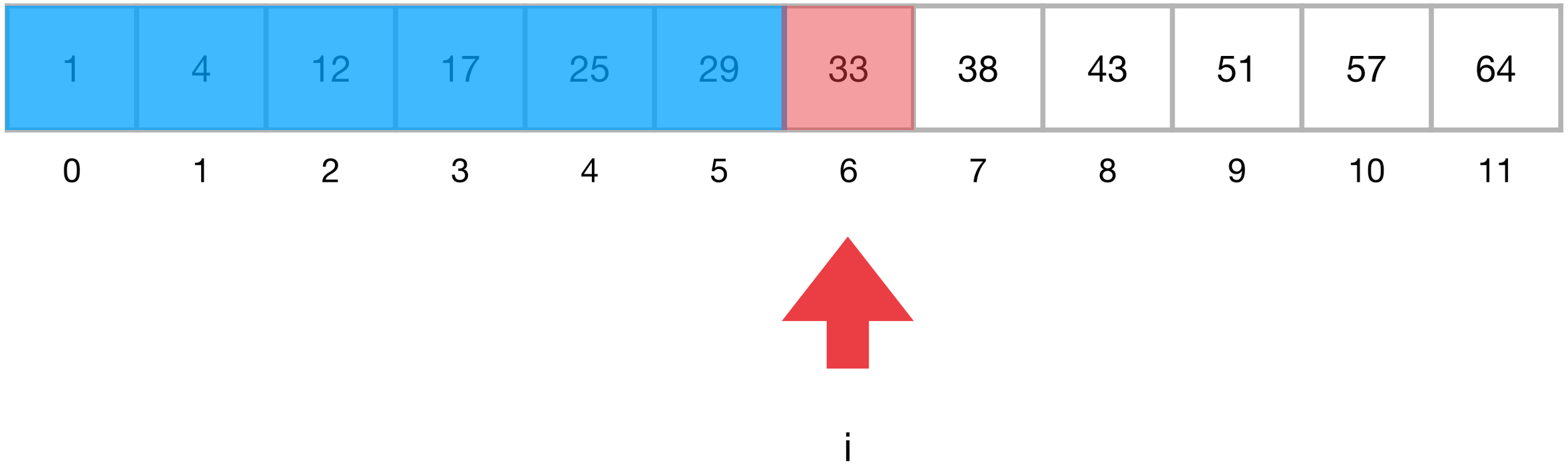
Recherche linéaire

Recherche de 3



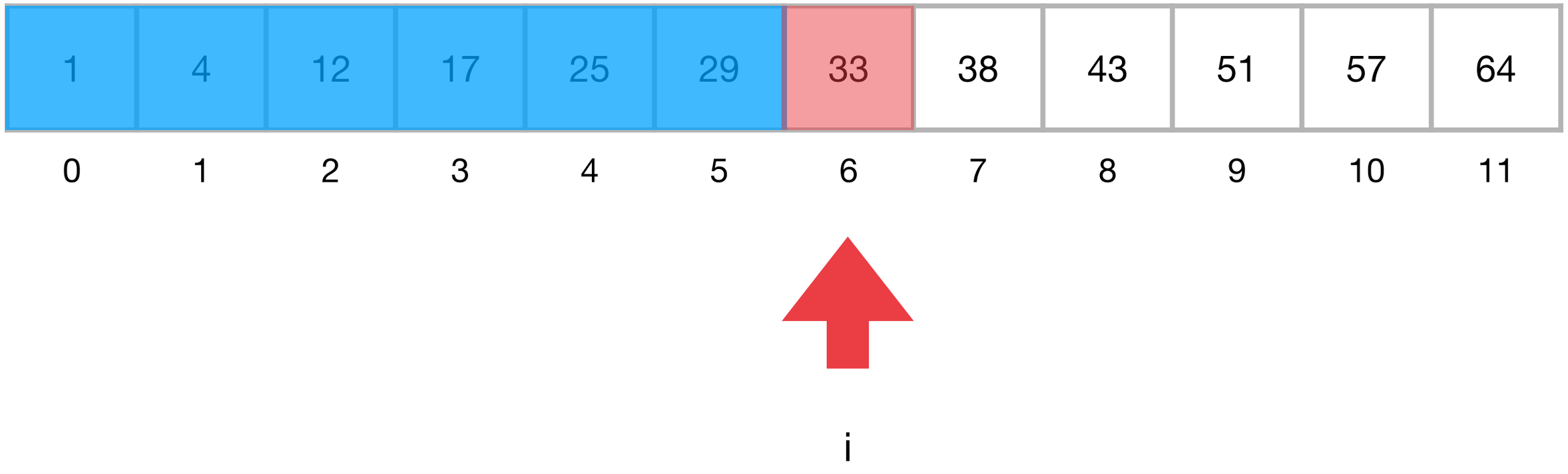
Recherche linéaire

Recherche de 3



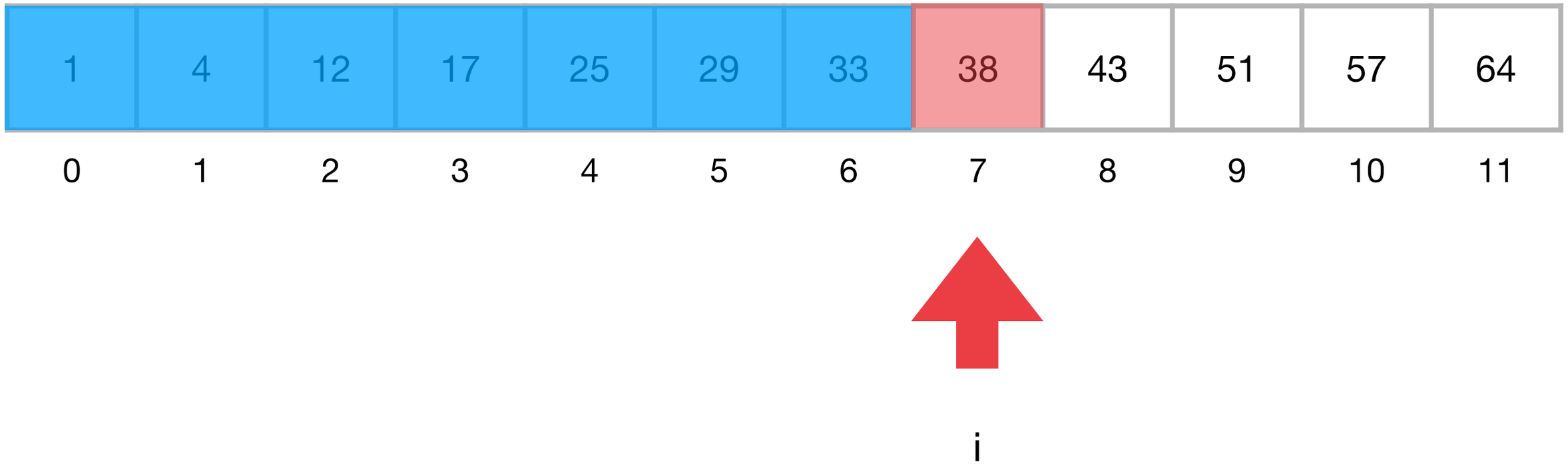
Recherche linéaire

Recherche de 3



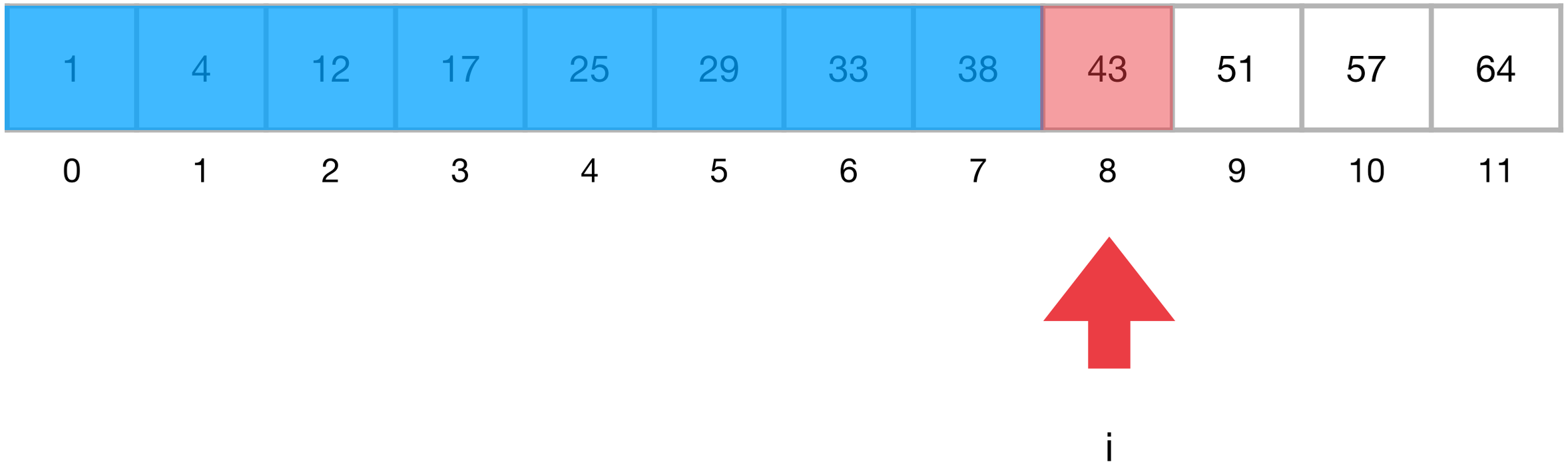
Recherche linéaire

Recherche de 3



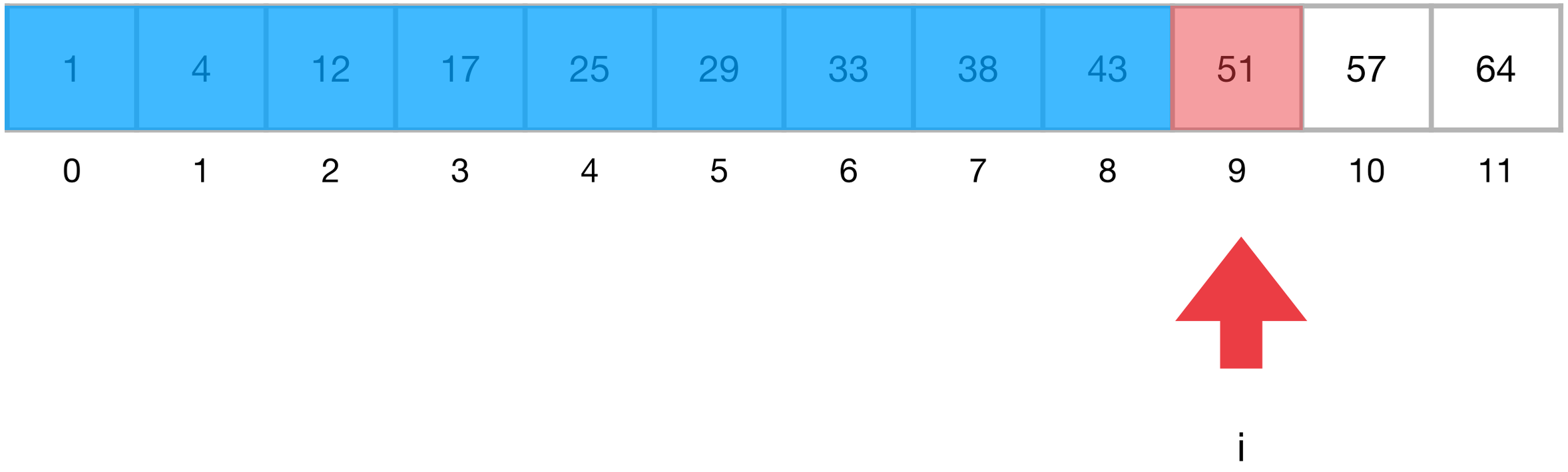
Recherche linéaire

Recherche de 3



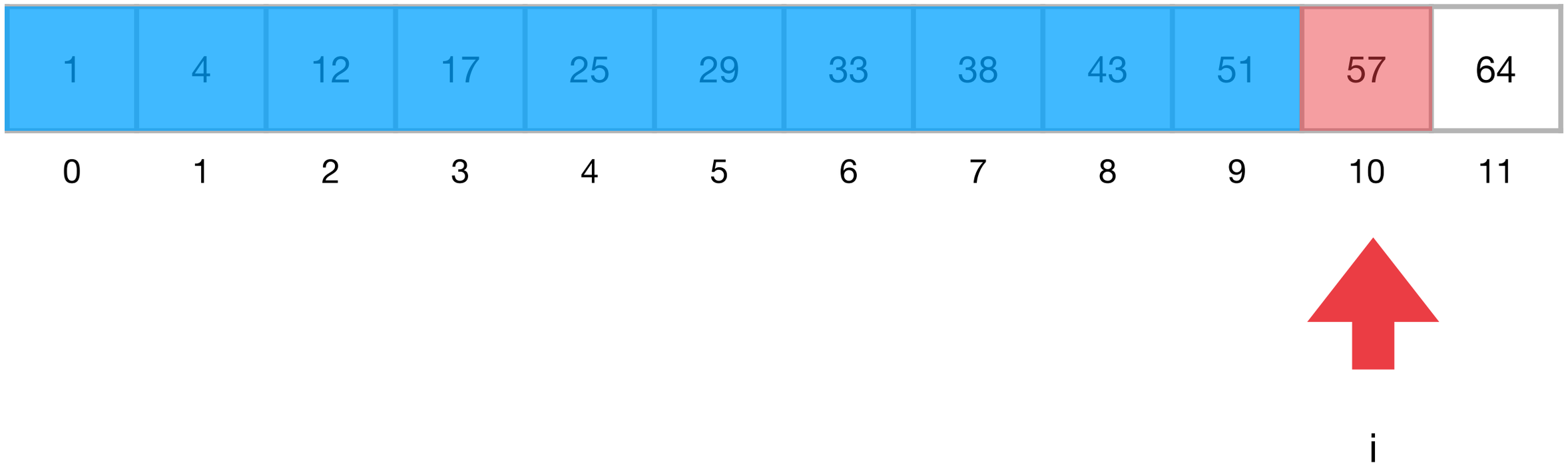
Recherche linéaire

Recherche de 3



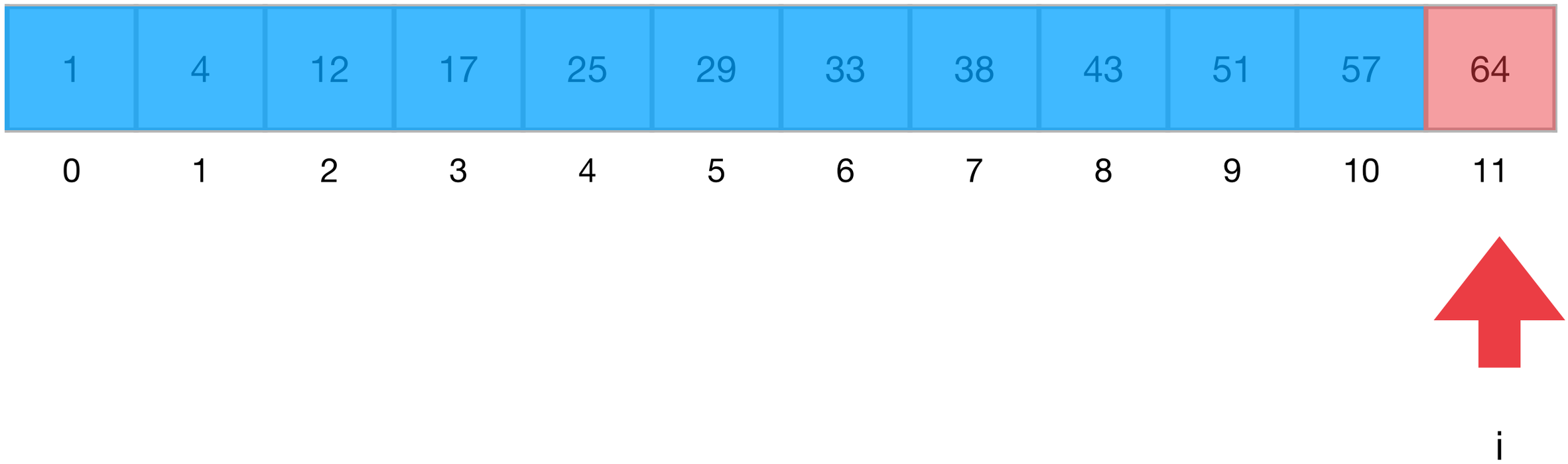
Recherche linéaire

Recherche de 3



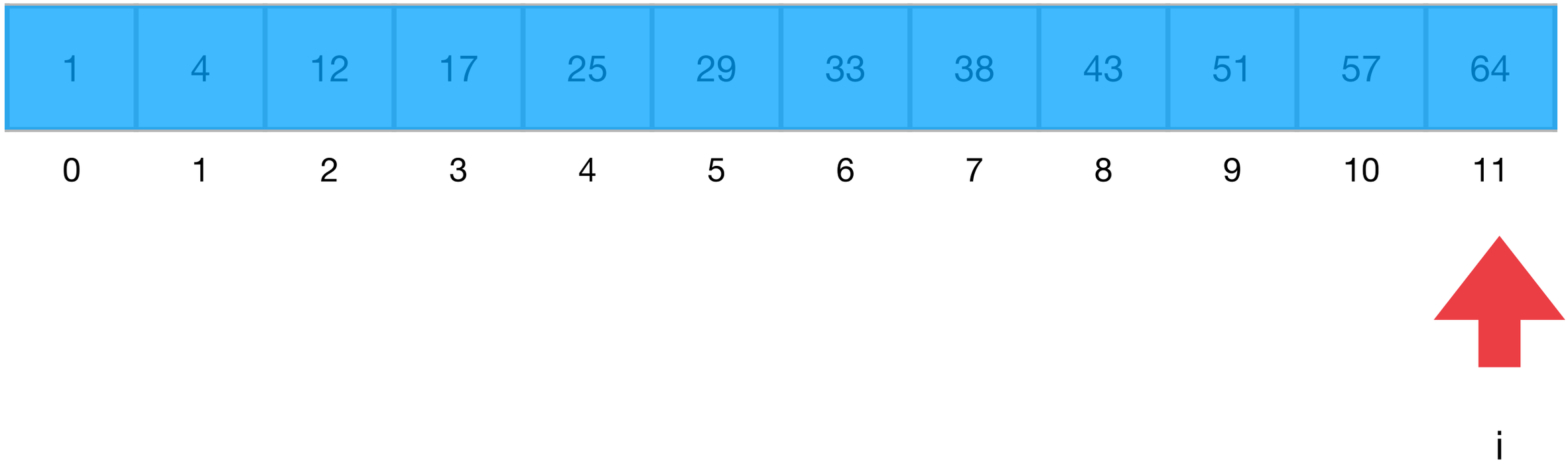
Recherche linéaire

Recherche de 3



Recherche linéaire

Recherche de 3



Recherche linéaire

Recherche de 3

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11

Comptage des opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)
    for i in range(n):
        if t[i] == e:
            return i
    return -1
```

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)
    for i in range(n):
        if t[i] == e:
            return i
    return -1
```

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
n = len(t)                                  1
for i in range(n):
    if t[i] == e:
        return i
return -1
```

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1
        if t[i] == e:
            return i
    return -1
```

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1
        if t[i] == e:                   1
            return i
    return -1
```

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1
        if t[i] == e:                   1
            return i                    1
    return -1
```

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1
        if t[i] == e:                   1
            return i                    1
    return -1                           1
```

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
n = len(t)                          1
for i in range(n):                  1
    if t[i] == e:                   1
        return i                    1
return -1                           1
```

si on a t[k] == e

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1
        if t[i] == e:                   1   } k + 1 fois
            return i                    1
    return -1                           1
```

si on a t[k] == e

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1
        if t[i] == e:                   1   } k + 1 fois
            return i                    1
    return -1                           1   0 fois
```

si on a t[k] == e

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1
        if t[i] == e:                   1   } k + 1 fois
            return i                    1
    return -1                           1   0 fois
```

si on a t[k] == e

total :

2k + 4

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
n = len(t)                          1
for i in range(n):                  1
    if t[i] == e:                   1
        return i                    1
return -1                           1
```

si e n’est pas là

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1
        if t[i] == e:                   1    n fois
            return i                    1
    return -1                           1
```

si e n’est pas là

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1
        if t[i] == e:                   1     n fois
            return i                    1     0 fois
    return -1                           1
```

si e n’est pas là

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1     n + 1 fois
        if t[i] == e:                   1     n fois
            return i                    1     0 fois
    return -1                           1
```

si e n’est pas là

Comptage des opérations

#opérations

```
def rechercher_séquentiel(t, e):
    n = len(t)                          1
    for i in range(n):                  1    n + 1 fois
        if t[i] == e:                   1    n fois
            return i                    1    0 fois
    return -1                           1
```

si e n’est pas là

total :

2n + 3

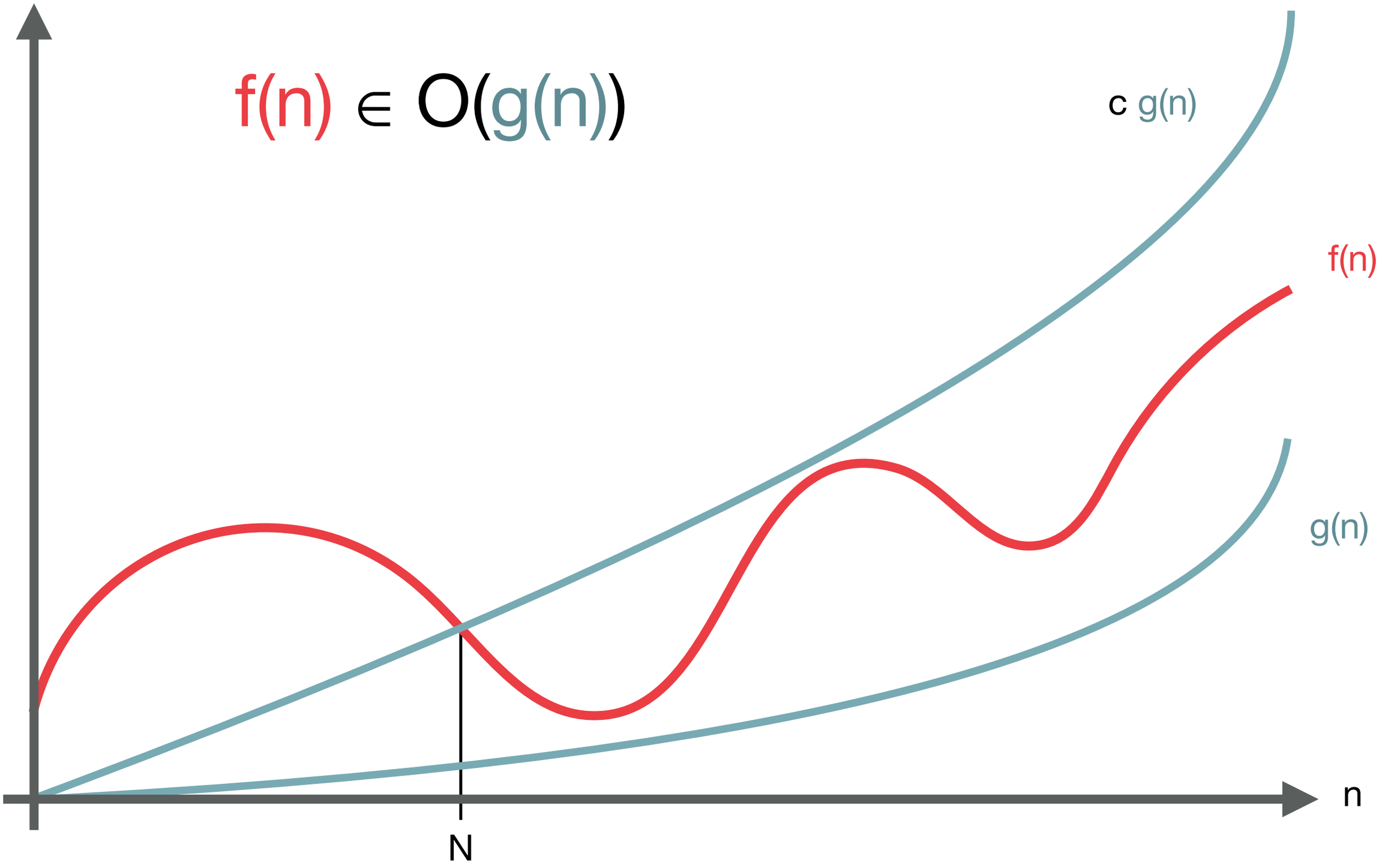
Efficacité

```
def rechercher_séquentiel(t, e):
    n = len(t)
    for i in range(n):
        if t[i] == e:
            return i
    return -1
```

- Si on a de la chance, on a `t[0] == e` et on termine tout de suite en 4 opérations
- Si `t[k] == e` on fait $2(k + 1) + 2 = 2k + 4$ opérations
- Si e n'est pas là on fait $2n + 3$ opérations

Notation « grand O »

$f(n) \in O(g(n))$

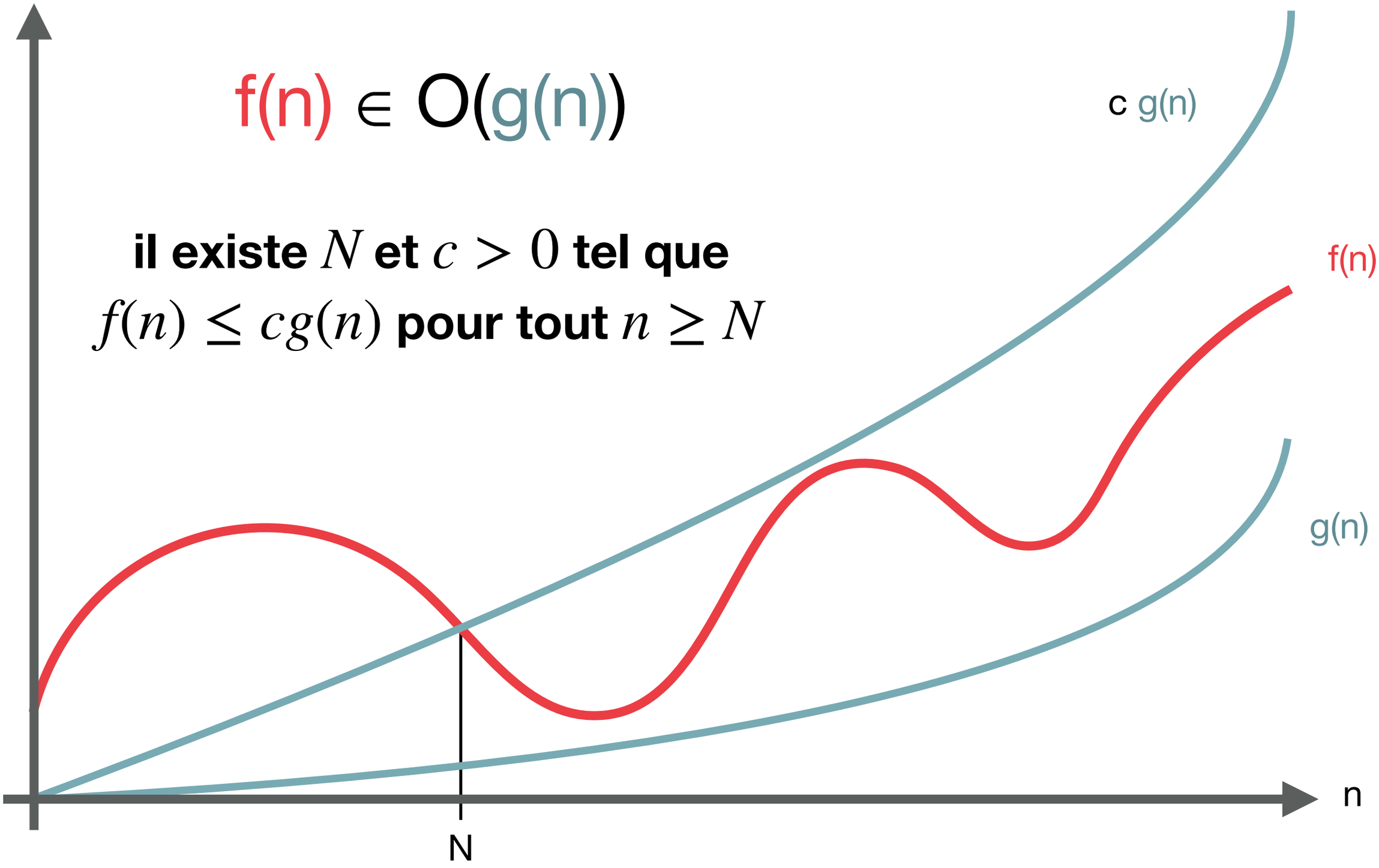


Notation « grand O »

$f(n) \in O(g(n))$

il existe N et $c > 0$ tel que

$f(n) \leq cg(n)$ **pour tout** $n \geq N$



Ordres de grandeur

Il existe N et $c > 0$ tel que $f(n) \leq cg(n)$ pour tout $n \geq N$

- $n \in O(n)$
- $n + 5 \in O(n)$
- $2n + 5 \in O(n)$
- $n^2 + 2 \in O(n^2)$
- $n^2 \notin O(n)$

Peut-on faire mieux que $O(n)$ pour la recherche dans un tableau ?

Exercices 3 et 4 du TD1