INFORMATIQUE

Complexité :
l'espace et le temps

Complexité spatiale et temporelle

On désigne par **complexité** un ordre de grandeur, exprimé en fonction de la dimension caractéristique du problème N :

Complexité spatiale : occupation en mémoire des données nécessaires pour traiter le PB.

Complexité temporelle : « durée » en nombre d'opérations pour traiter le PB

Remarques :

- tous les types de données n'occupent pas le même espace mémoire.
- toutes les opérations ne prennent pas le même temps (+*-/= ; == < != >).

Nous nous limiterons à dégager un ODG de la commensurabilité du PB avec des O()

exemples : O(1) O(Log(N)) O(N) O(N.log(N)) O(N^2) O(2^N) O(N!)

On veut en général connaître la complexité :

- dans le pire des cas.
- dans le meilleur des cas.
- en moyenne (HP) : quand les données du PB sont aléatoires.

Le rythme technologique : l'enjeu de la complexité

L'évolution des technologies sur les 50 dernières années impose le constat suivant :
Dans le temps où la mémoire disponible dans un calculateur double,
La puissance de calcul (la vitesse de calcul) double également !
Mémoire et puissance de calcul sont proportionnels !

Or en recherche si les progrès techniques permettent de disposer de 2 fois plus de mémoire, on veut automatiquement traiter un problème deux fois plus gros en mémoire !

Mais quel sera alors le temps de calcul : **tout dépend de l'algorithme !**

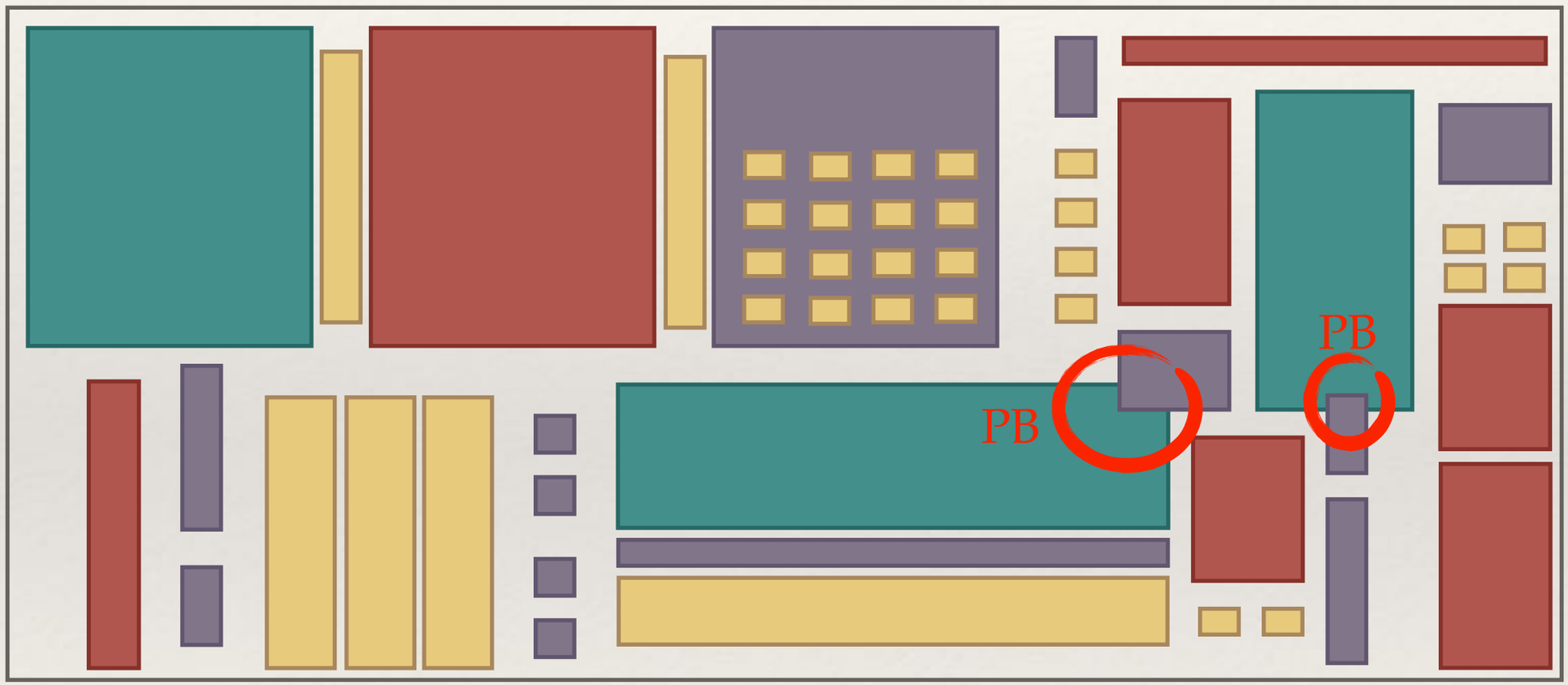
<u>Algorithme linéaire :</u>

<u>Algorithme quadratique :</u>

Ex : **La loi de Moore n'est pas seulement technologique mais aussi algorithmique !**

Idée **schématique** du problème de conception :

Soit N zones rectangulaires à graver sur le silicium, comment vérifier qu'il n'y a pas d'intersection ?



- Algorithme naïf :

- Algorithme à balayage :

<u>Conclusion :</u> La frontière technologique !

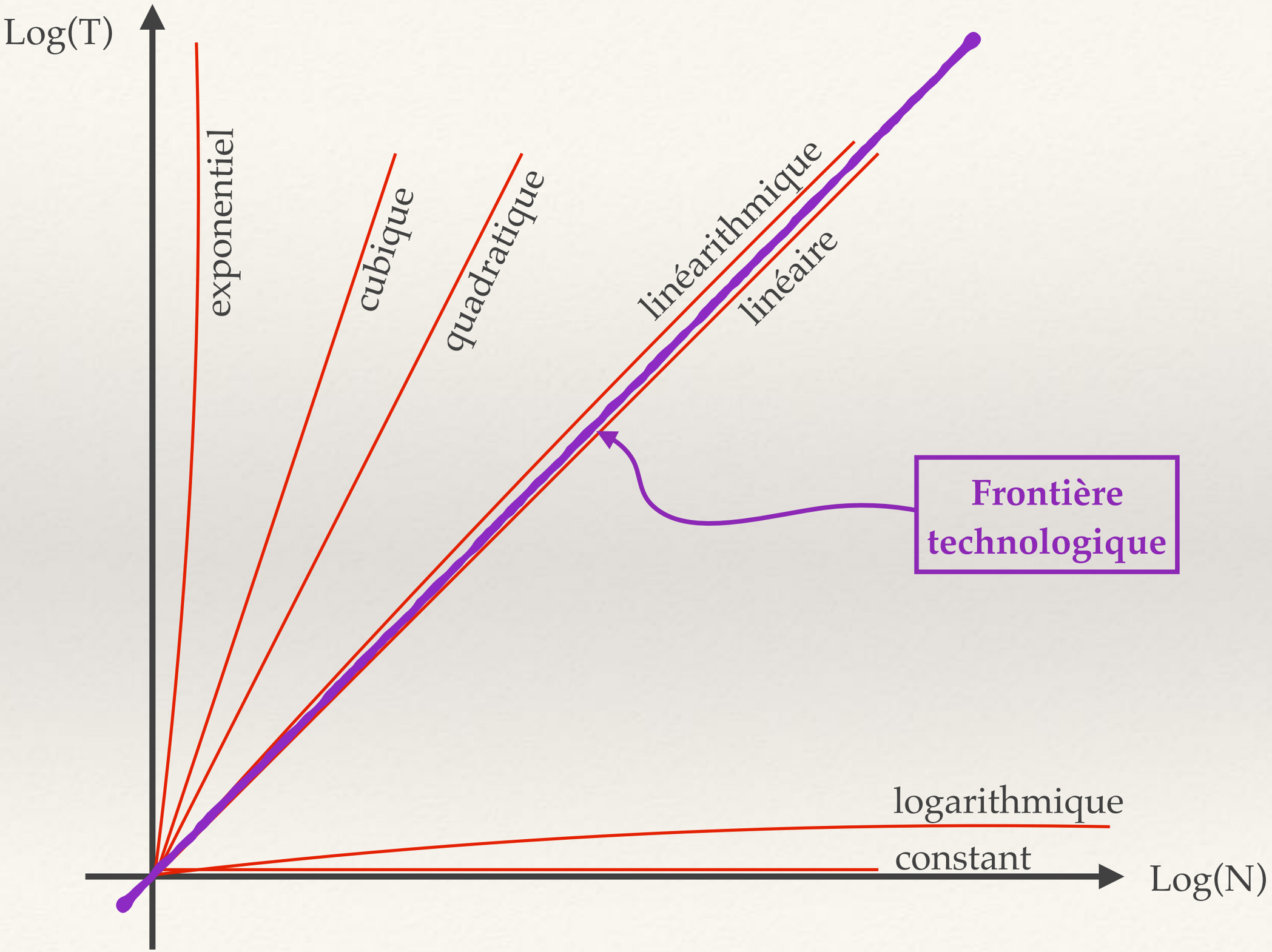
Le passage d'algorithmes quadratiques O(N^2) à des algorithmes dits «linéarithmiques» soit en O(N.Log(N)) a permis l'émergence de **nouvelles technologies** :

- **FFT Fast Fourier Transform :** obtenir un spectre en temps réel
  [oscilloscope, compression mp3, IRM, traitement d'image etc…

ainsi que la possibilité de **nouveaux champs de recherche :**

- **Problème à N-corps :** traitement des <u>systèmes complexes</u> en interaction à longue portée
  [recherche en astrophysique, physique de l'état condensé, chimie des polymères]

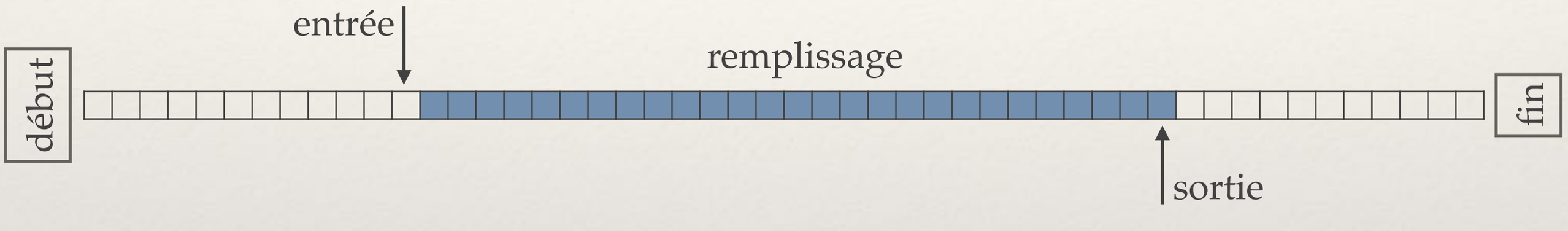
<u>Résumé :</u> **le coût de l'algorithme est un enjeu majeur**
Aussi bien sur le plan théorique que technologique.



Tableaux et stockage dynamique des données

Dans la mémoire de la machine toute structure de données est finalement stockée dans un tableau dont la mémoire est « allouée » d'une adresse de départ à une adresse de fin.

Exemple schématique : stockage d'une queue



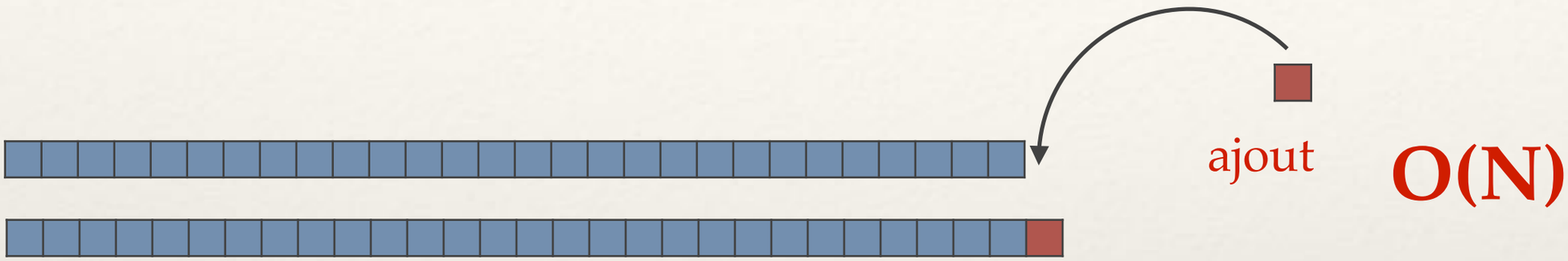
Deux contraintes techniques importantes :

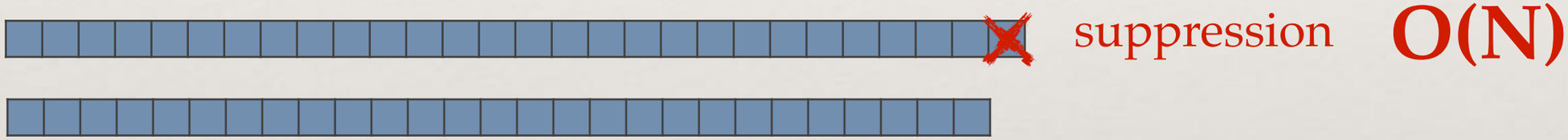
1 - Utilisée ou pas la mémoire allouée n'est plus disponible pour un autre programme.
La question se pose donc d'une **allocation dynamique** adaptée au besoin du programme qui change au cours du temps.

2 - Tout changement de la taille du tableau nécessite une seconde allocation mémoire
=> problème spatial : il faut avoir assez de mémoire O(N)
et ensuite elle requiert surtout la recopie complète des données dans le nouveau tableau
=> problème temporel : cela prend un temps en O(N)

Quelle stratégie adopter ?

Mauvaise idée : **ajouter** (**enlever)** une seule cellule à la fois
On ré-alloue un tableau de taille N+1 (ou N-1)
et on recopie tous les éléments !





PB : A chaque ajout / suppression -> **coût en O(N)**
Or très souvent, la taille d'un tableau fluctue autour d'une valeur N_0.

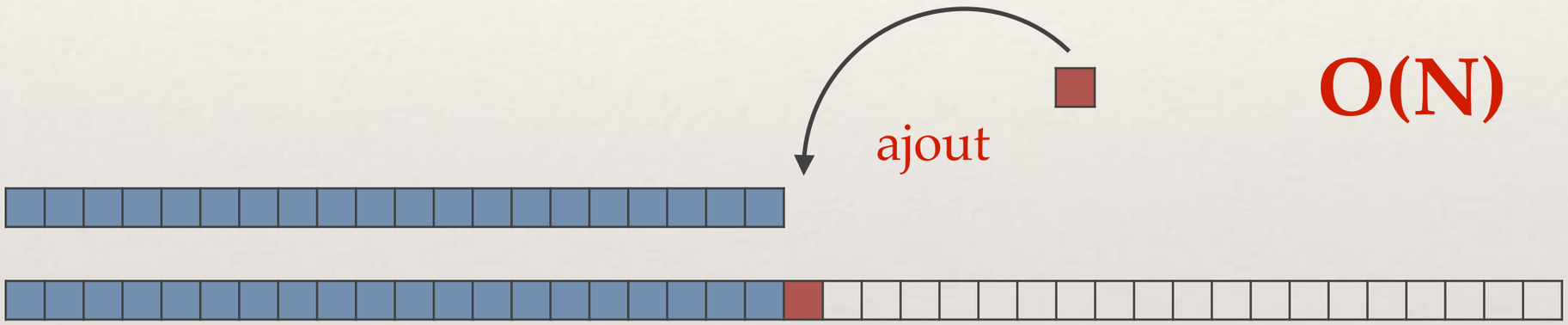
Exercice :

Quel est le coût de création d'un tableau de taille N
par ajout de N éléments successifs ?

Quelle stratégie adopter ?

<u>Meilleure idée :</u> **règle d'allocation dynamique**

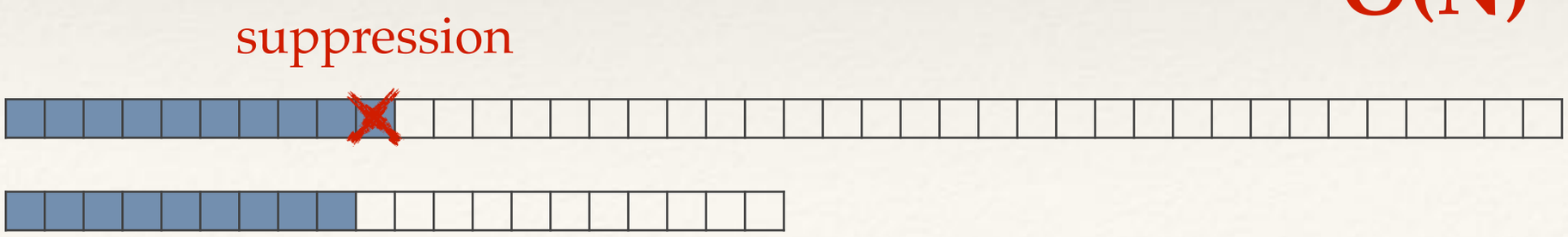
Tableau plein à 100% => On alloue un tableau de taille 2N
=> On recopie les N éléments



O(N)

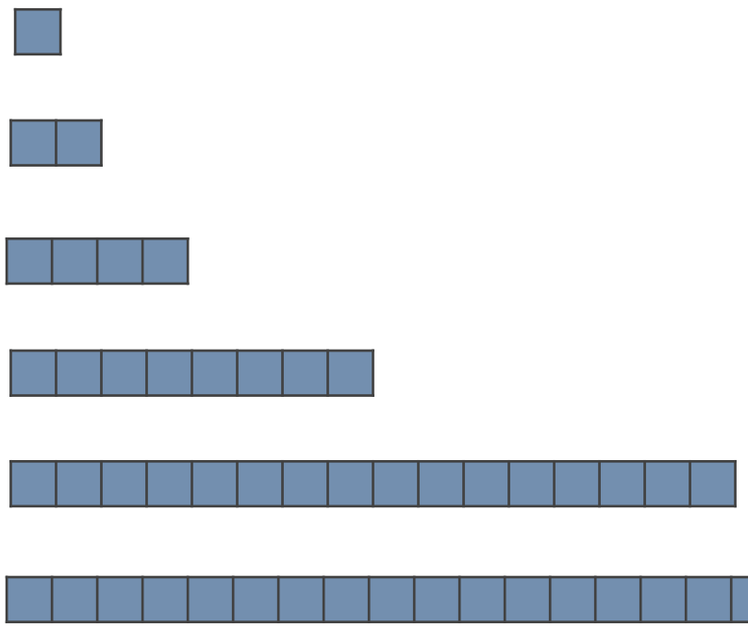
Tableau se vide à 25% => On alloue un tableau de taille N/2
=> On recopie les N/2 éléments

O(N)



Exercice :

Quel est le coût de création d'un tableau de taille N par la règle d'allocation dynamique ?



N ajouts individuels

\+ $LOG_2(N) - 1$ recopies : $1+2+4+8+16 = N - 1$

Au total : 2N-1 écritures pour N éléments

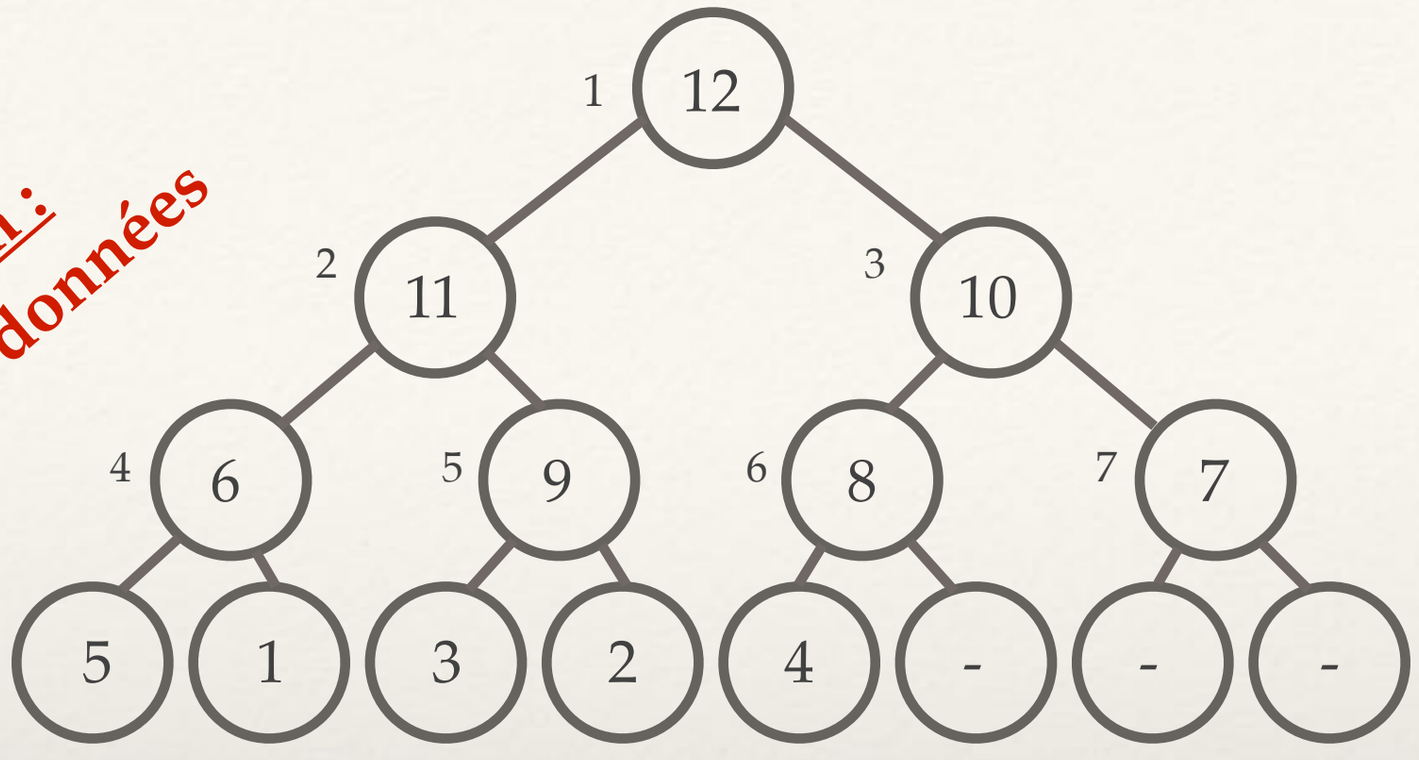
On parle alors de <u>coût amorti</u> : ajout / suppression en O(1) <u>en moyenne</u>

<u>Problème rémanent :</u>

Lors de la recopie, le processus est en attente, pour un coût temporel en O(N)

Exemple concret de stockage d'un arbre sans notion de classe

Abstraction :
Structure de données



Système de conversion

Réalité mémoire : tableau

Rq : n = i + 1

	12	11	10	6	9	8	7	5	1	3	2	4	-	-	-
n ->	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Règle de construction de la structure données :

Invariant : Le père est toujours plus grand que ses fils

Navigation dans l'arborescence

Indice du père : $n_P = n_g \,//\, 2 = n_d \,//\, 2$

Indice du fils gauche : $n_g = 2 \times n_P$

Indice du fils droit : $n_d = 2 \times n_P + 1$

Rq : n = i + 1