2. Structures de données

Comparaison des structures de données

Tableau : pré-affecté en mémoire, n cases statiques (`ArrayList`)

Liste chaînée : structure chaînée, dynamique (`LinkedList`)

Table de hachage : tableau de listes chaînées (`HashSet`)

Arbre binaire de recherche : structure d'arbre explicite (`TreeSet`)

Tas : structure d'arbre dans un tableau (`PriorityQueue`)

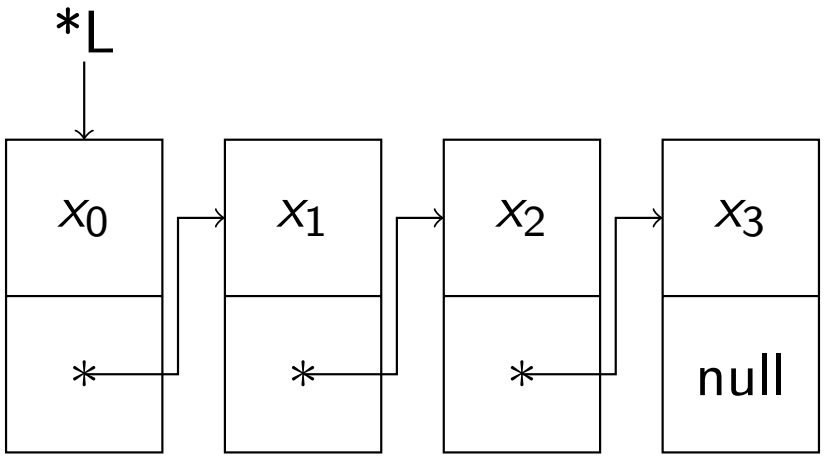
2.1 - Liste Chainée

Liste chaînée

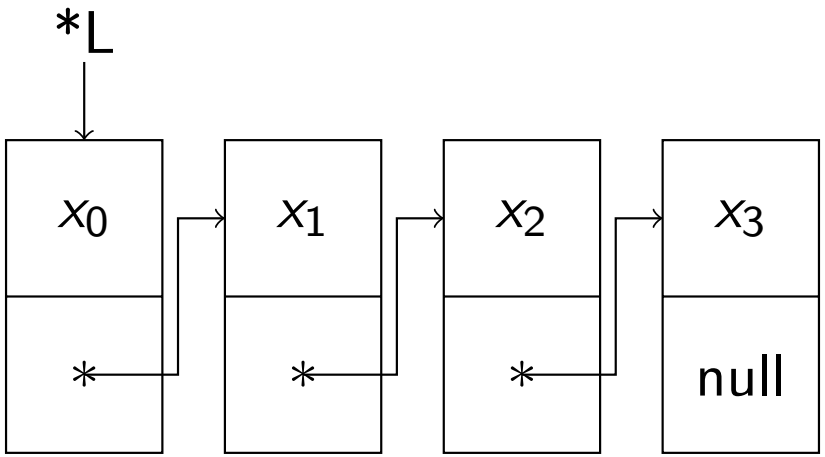
Principe

Une liste chaînée est une suite de maillons. Chacun connaît seulement le maillon suivant.

- simple structure avec une charge et un pointeur vers le maillon suivant
- permet de stocker un ensemble d'éléments, comme un tableau



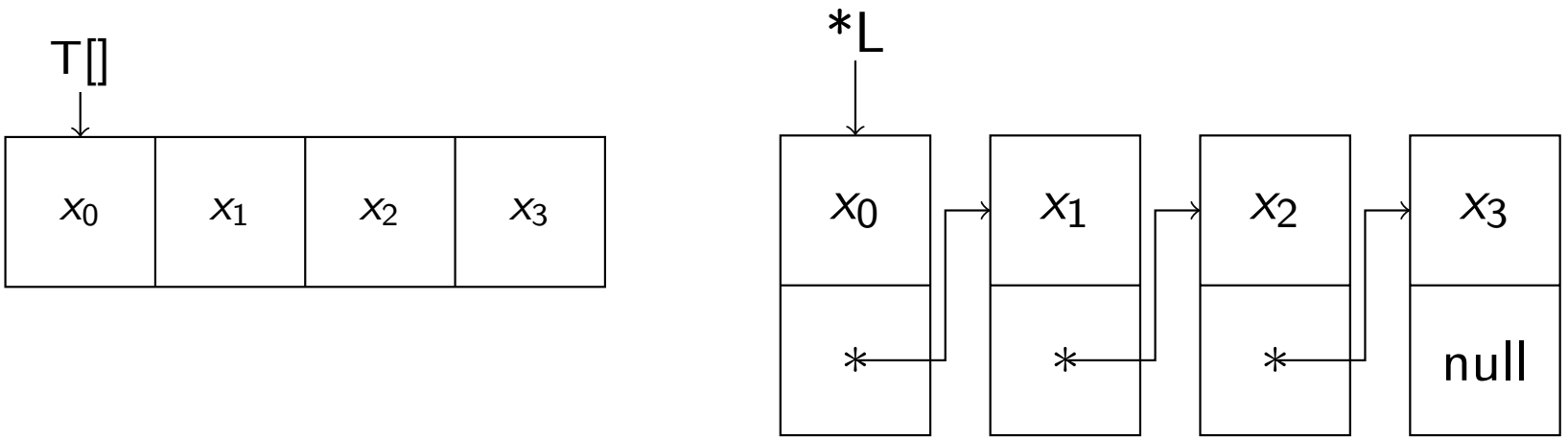
Pour/contre



- très souple, redimensionnée facilement
- permet une insertion/suppression en milieu de chaîne sans tout décaler à la main
- impossible (sans parcours) d'accéder à un élément par sa position, de connaître la taille de la liste
- LinkedList en java / patron itérateur

Comparaison des structures de données

Tableau VS Liste Chainée



Opération	accès	recherche	insertion, suppression	extraction du plus petit
Tableau	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Liste chaînée	$O(n)$	$O(n)$	$O(1)$	$O(n)$

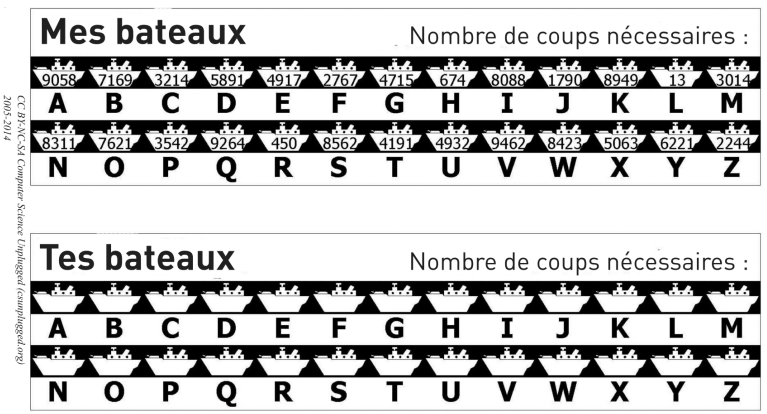
2.2. Structures associatives

Un petit jeu pour les enfants.

from Computer Science unplugged

Règles du jeu

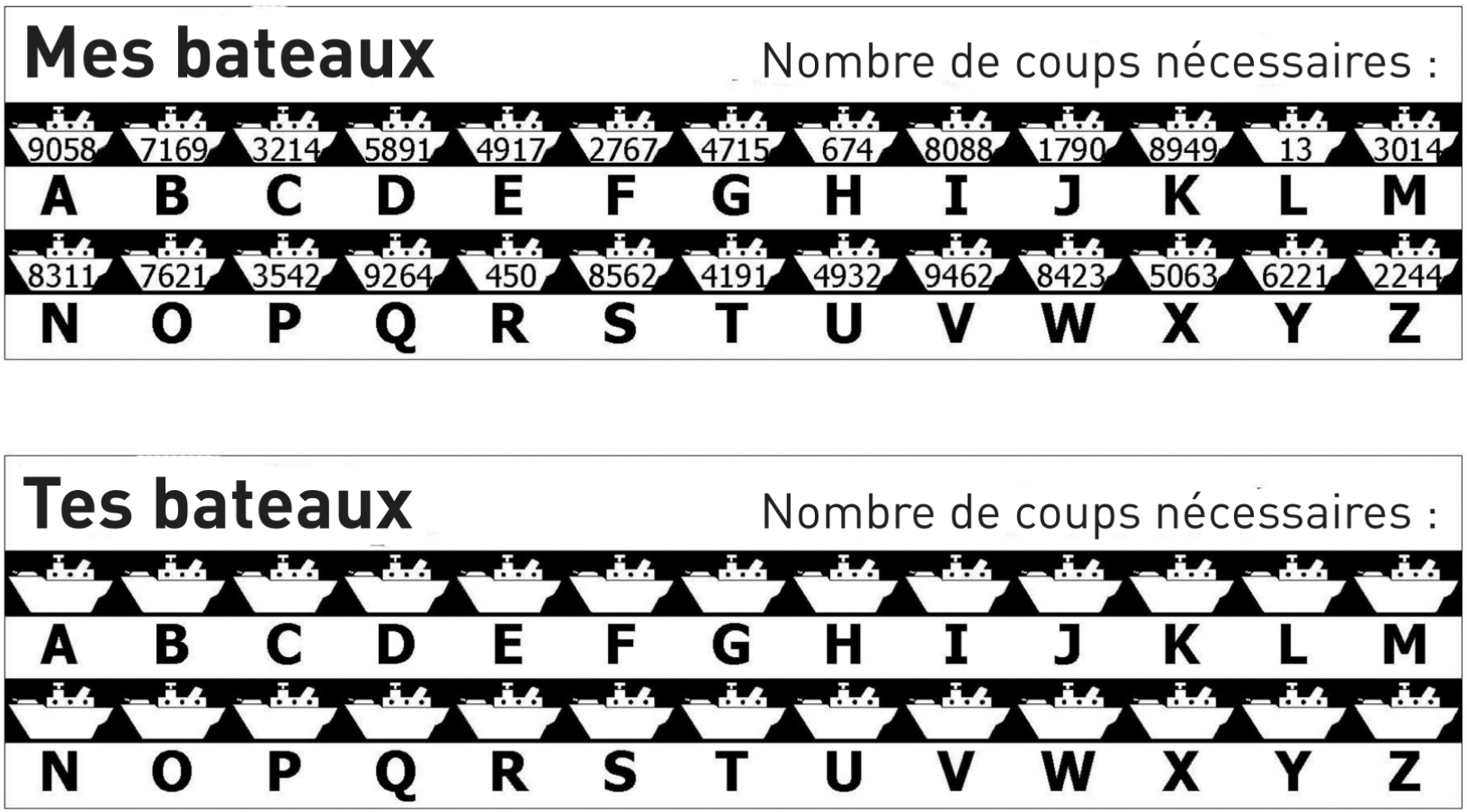
1. Choisir un bateau
2. Indiquer son numéro à l'adversaire
3. On cible un bateau adverse, l'adversaire nous indique son numéro.
4. Si c'est le numéro qu'il a choisi, c'est gagné.



L'adversaire à une carte similaire, avec des numéros différents

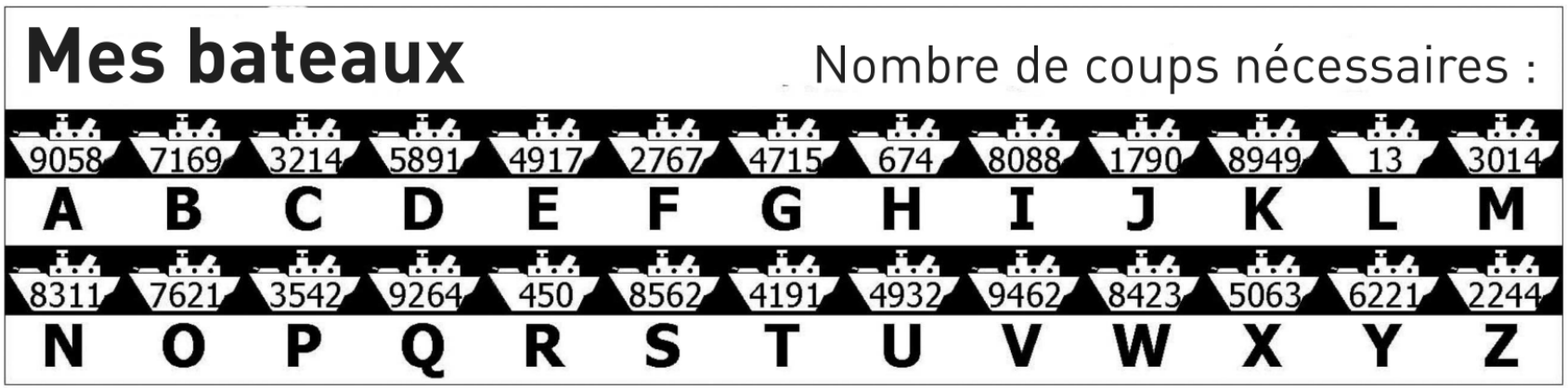
Une partie

Tableau normal



Nombre de tirs

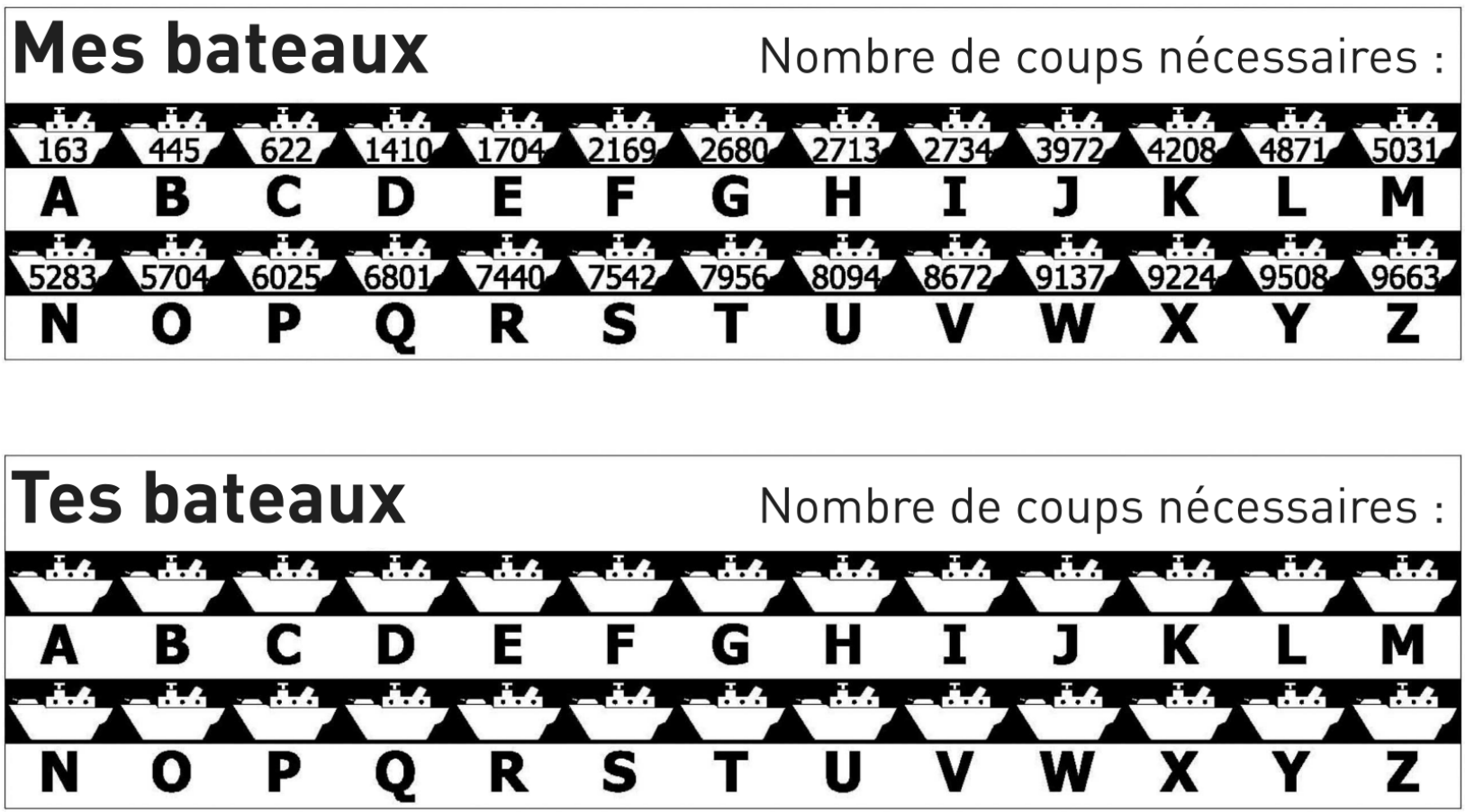
Tableau Normal



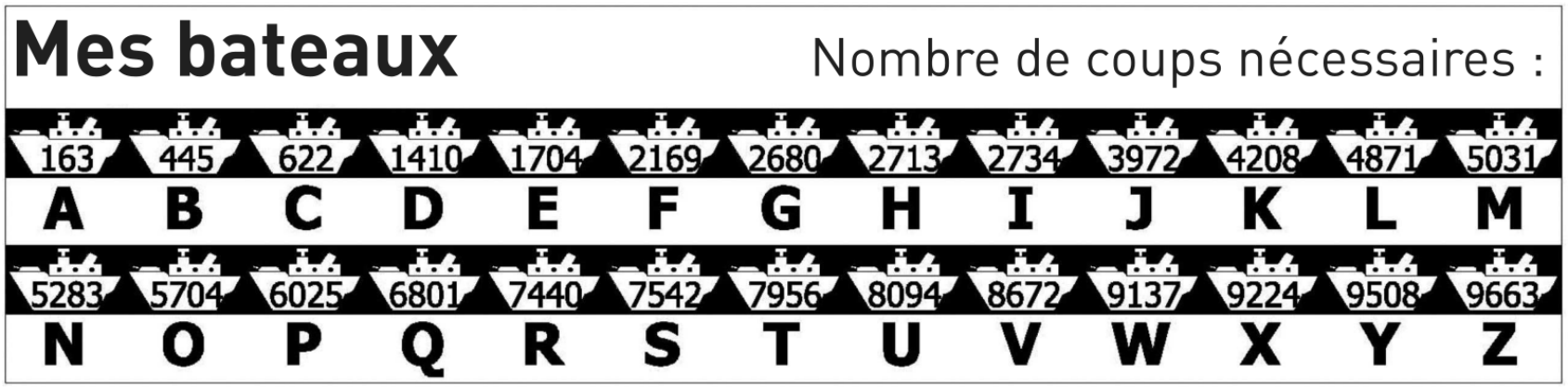
Structure non organisée

- n appels dans le pire cas
- $\frac{n}{2}$ en moyenne
- Recherche en $O(n)$

Une autre partie ?



Dichotomie

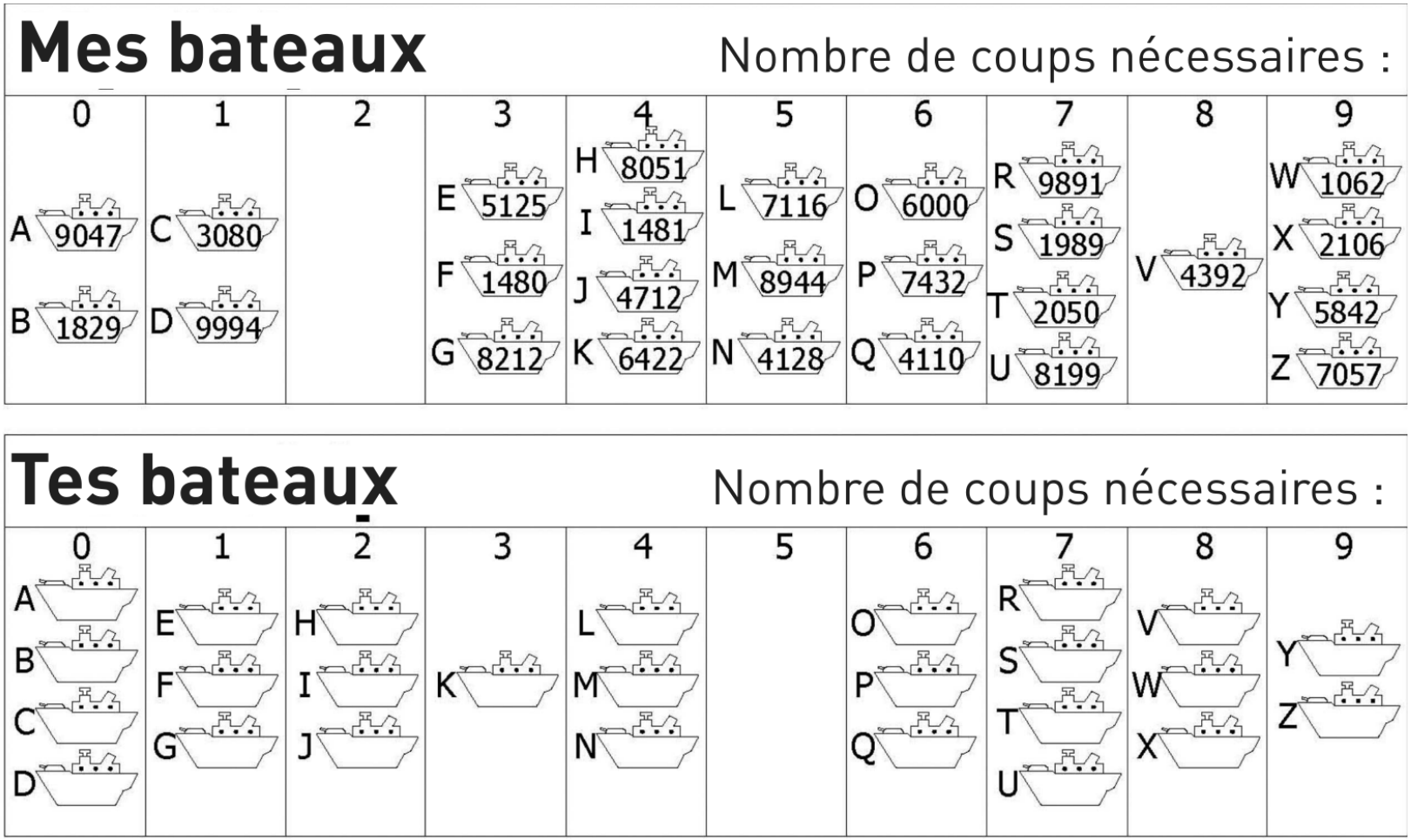


- Structure indexée.
- Éléments dans l'ordre.

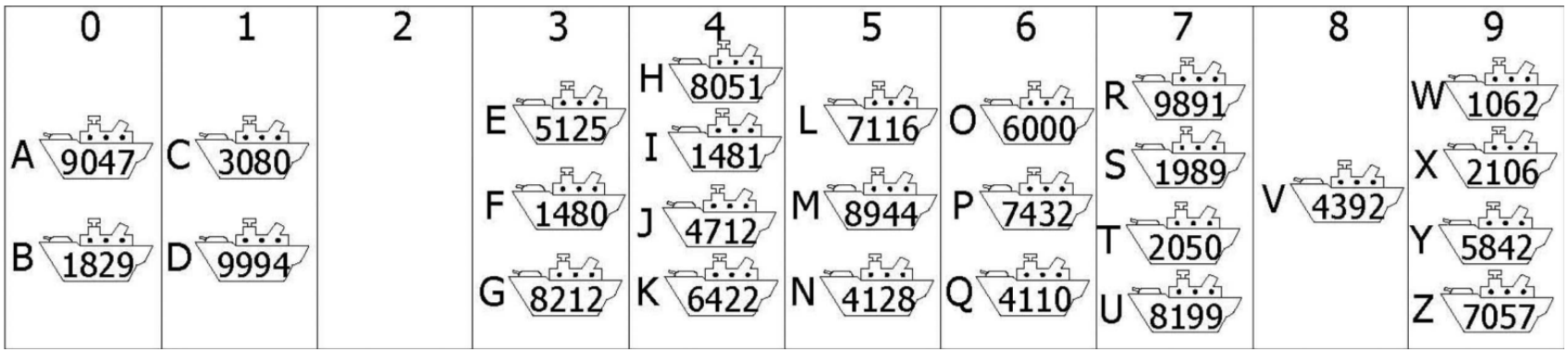
Structure triée

- Recherche en $O(log_2(n))$ (dans le pire cas et en moyenne).

Encore une partie ?



Recherche par clé



- Pour chaque valeur, une clé est calculée (appelé *hash*).
- Pour chaque clé, on regroupe l'ensemble des éléments qui correspondent (par exemple liste chaînée).

Table de hashage

- Fort rôle de la taille de l'espace des clés disons k clés.
- Recherche en $O(\frac{n}{k})$ en moyenne.

Fonction de Hashage

Attributs recherchés

- clé décorélée de la valeur initiale (ensemble de valeurs homogènes)
- Peu de collisions (éléments ayant la même image)

Usages

- HashSet
- HashMap, dictionnaire (hash de la clé).
- Cryptographie (Signature, stockage de mot de passe)

Comparaison des structures de données

Tableau : pré-affecté en mémoire, n cases statiques (e.g. `ArrayList`)

Liste chaînée : structure chaînée, dynamique (e.g. `LinkedList`)

Table de hachage : Tableau de listes chaînées (e.g. `HashSet`)

Comparaison des structures de données

Complexité des opérations

Opération	accès	recherche	insertion, suppression	extraction du plus petit
Tableau	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Liste chaînée	$O(n)$	$O(n)$	$O(1)$	$O(n)$
Tableau trié	NA	$O(\log_2(n))$	$O(n)$	$O(1)$
Hashset	NA	$O(1)$	$O(1)$	$O(n)$

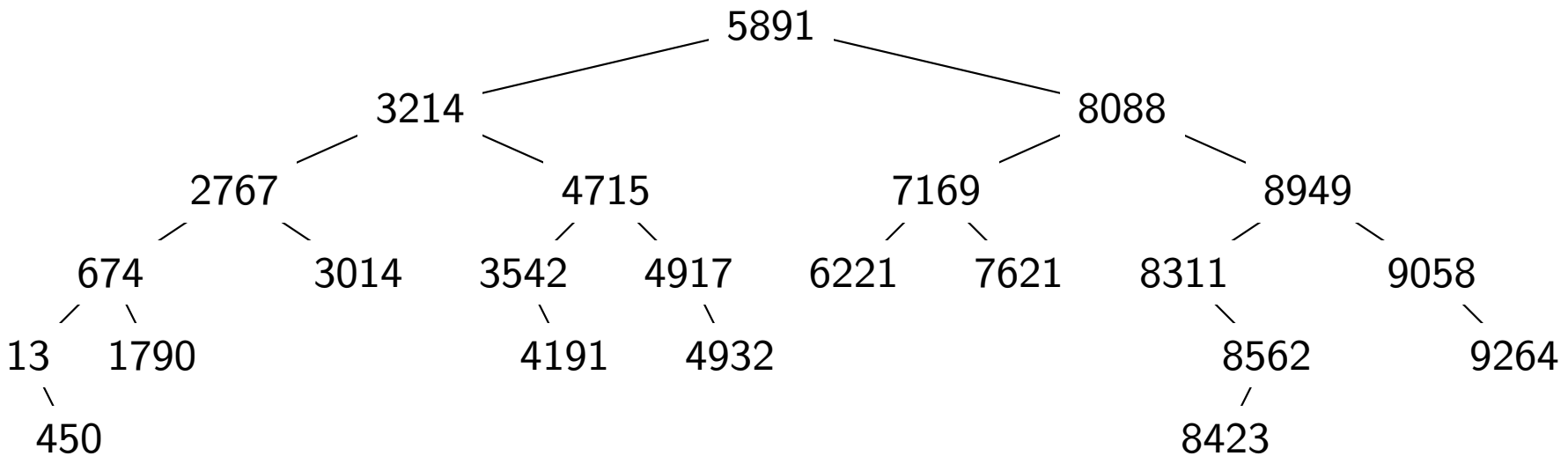
Comparaison des structures de données

Complexité des opérations

Opération	accès	recherche	insertion, suppression	extraction du plus petit
Tableau	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Liste chaînée	$O(n)$	$O(n)$	$O(1)$	$O(n)$
Tableau trié	NA	$O(\log_2(n))$	$O(n)$	$O(1)$
Hashset	NA	$O(1)$	$O(1)$	$O(n)$

En pratique, des techniques d'accélération.

Arbre binaire de recherche



- Profondeur dans le pire cas : n (nombre de valeurs)
- Profondeur en moyenne : $\log_2(n)$

Arbre binaire de recherche

Principes

- Structure d'arbre
- sous arbre gauche des valeurs inférieures, sous arbre droit des valeurs supérieures.

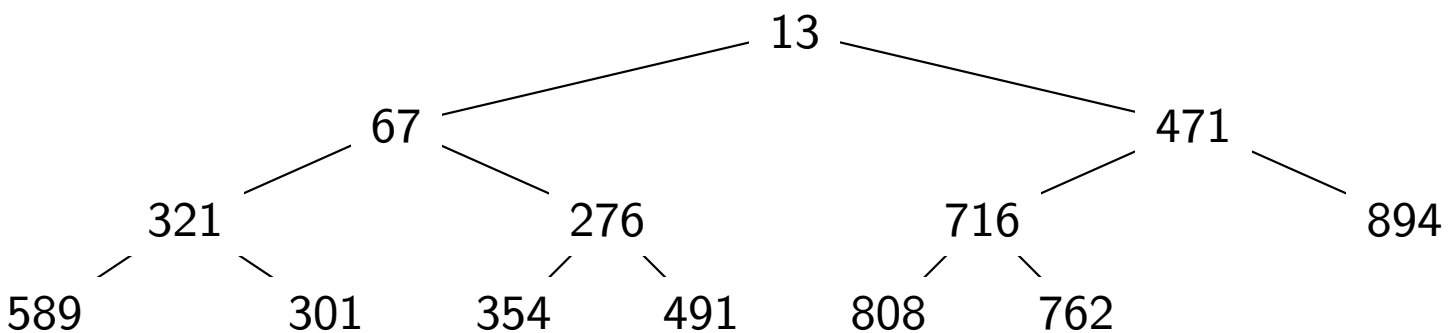
Usages

- insertion et recherche coûtent la profondeur de l'arbre
  (n dans le pire cas, $log_2(n)$ en moyenne)
- Suppression délicate avec rotation
- possiblité d'arbres rouge-noir qui utilisent des rotations et garantissent une profondeur de $\log_2(n)$

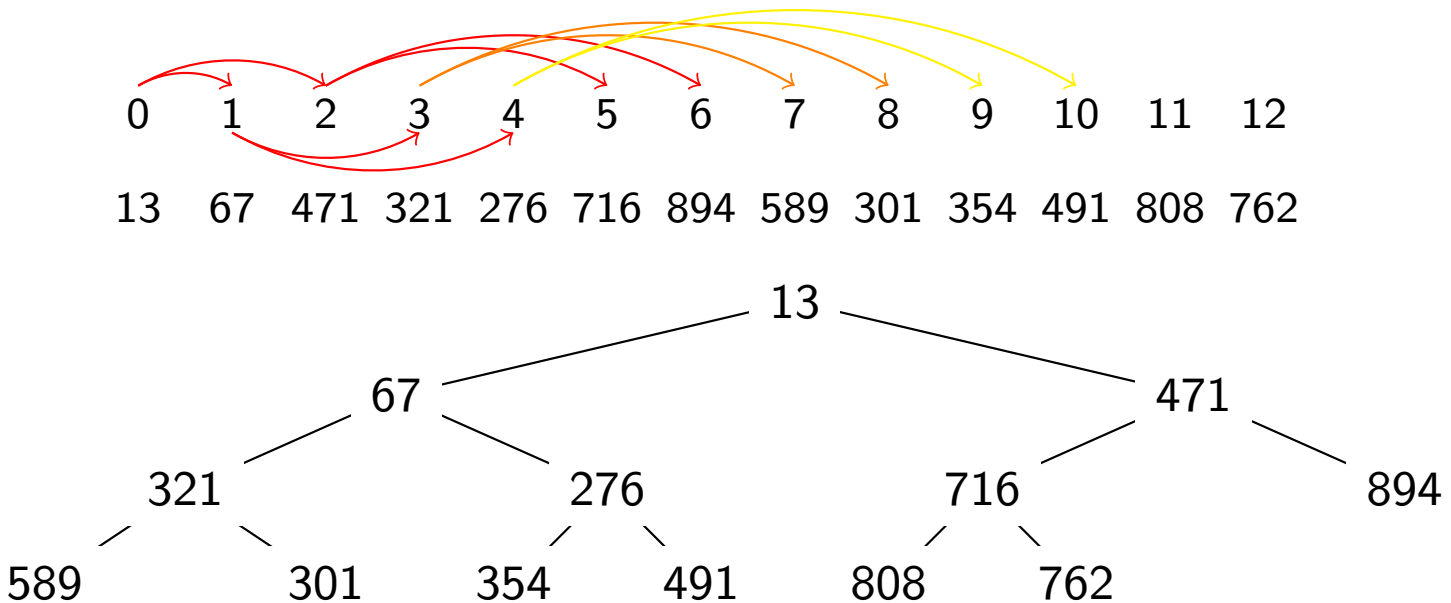
Les tas

Définition :

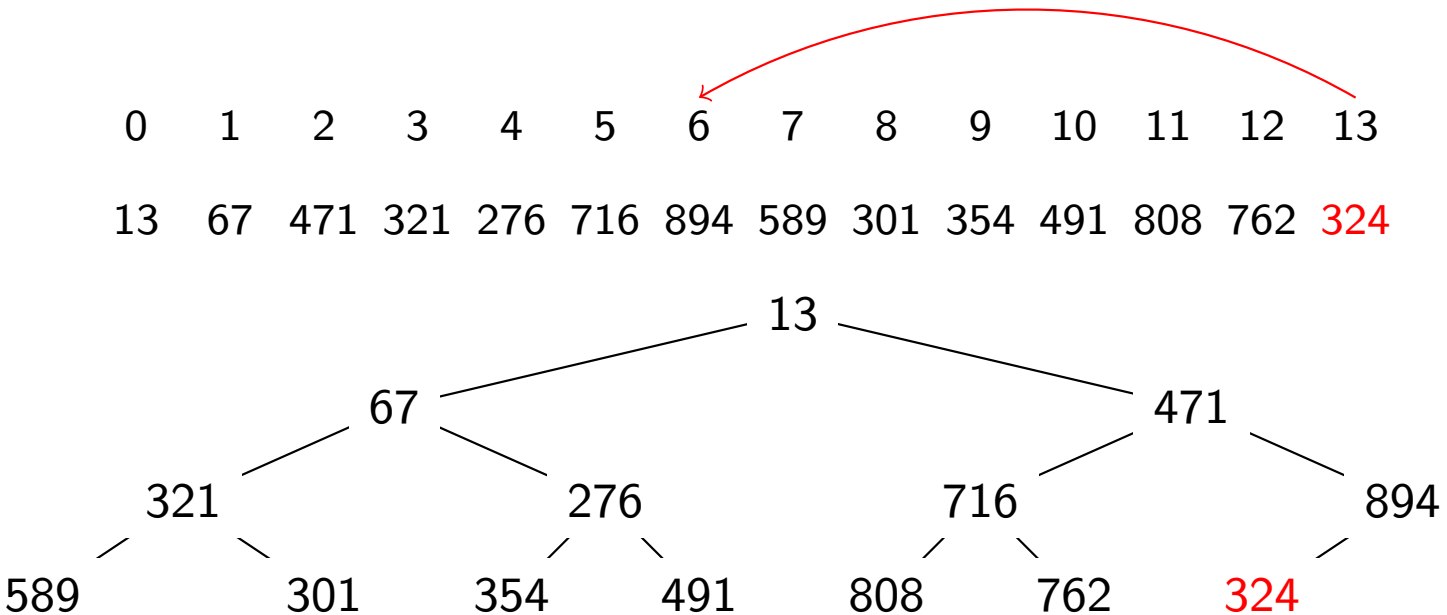
- Une structure semi-triée rangée dans un tableau
- Règle du jeu : chaque nœud est plus petit que ses enfants.
- en Java, implémentée dans les `PriorityQueue`.



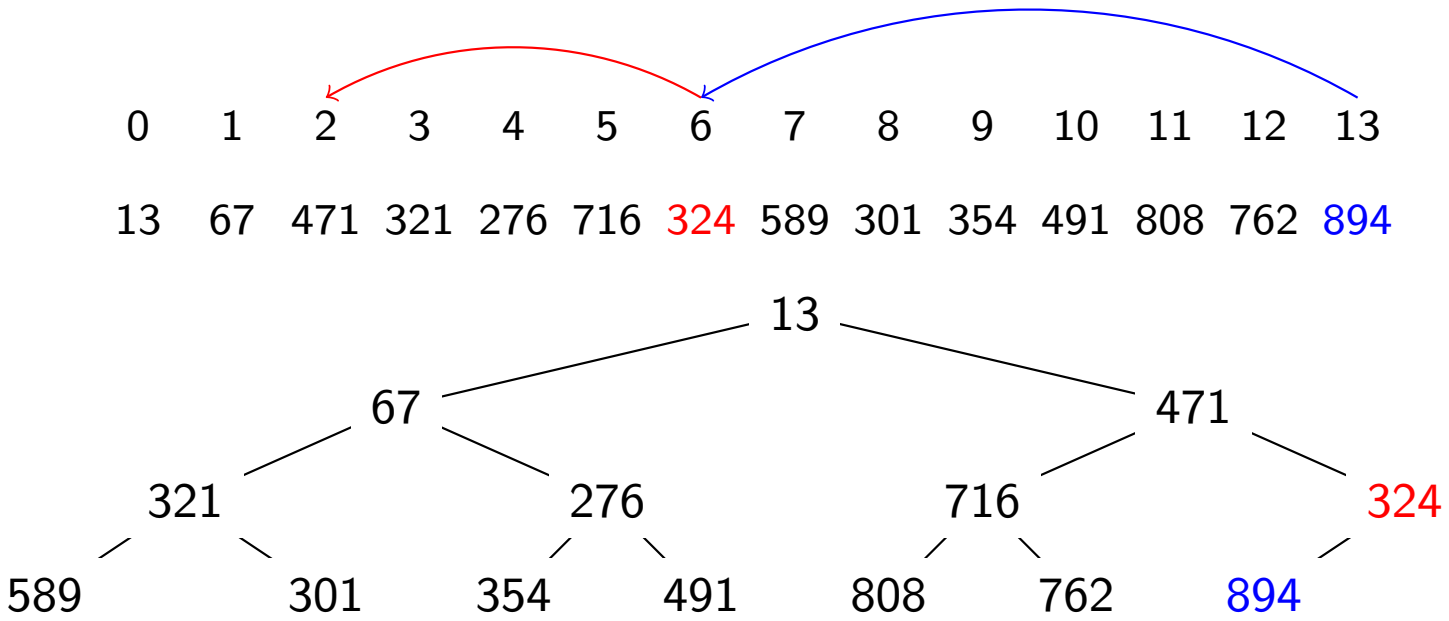
Insertion dans un tas



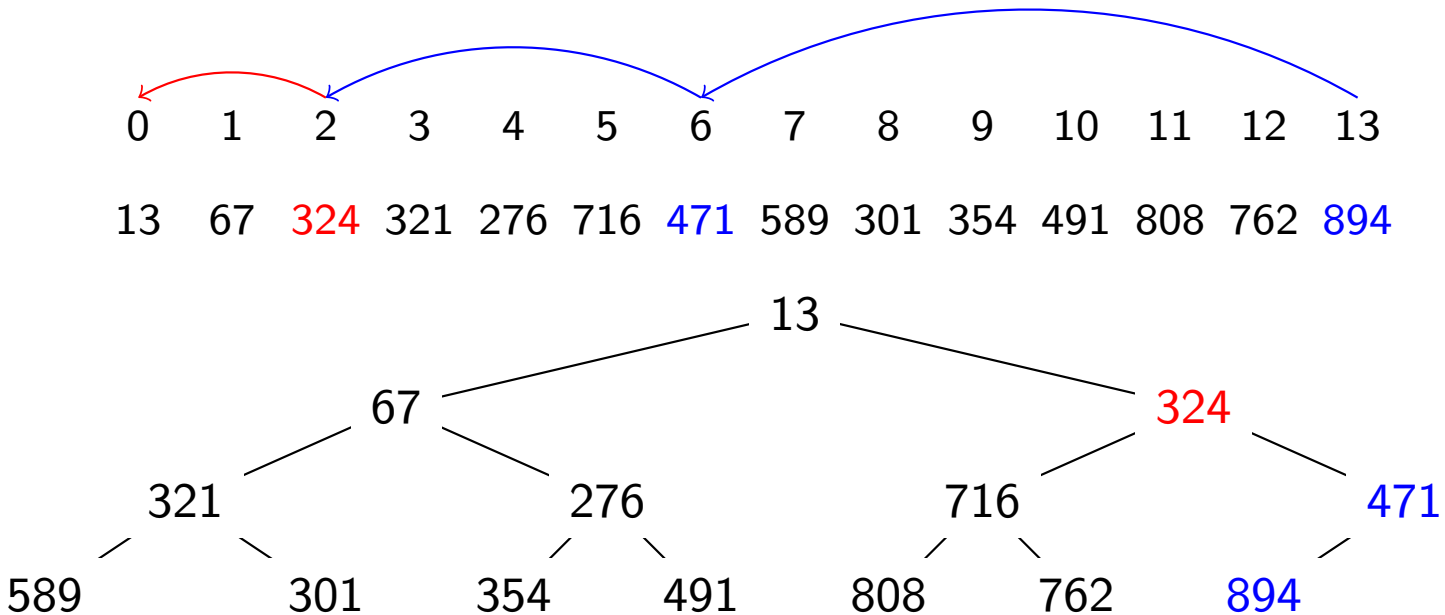
Insertion dans un tas



Insertion dans un tas



Insertion dans un tas



Comparaison des structures de données

Tableau : pré-affecté en mémoire, n cases statiques (`ArrayList`)

Liste chaînée : structure chaînée, dynamique (`LinkedList`)

Table de hachage : tableau de listes chaînées (`HashSet`)

Arbre binaire de recherche : structure d'arbre explicite (`TreeSet`)

Tas : structure d'arbre dans un tableau (`PriorityQueue`)

Comparaison des structures de données

Complexité des opérations

Opération	accès	recherche	insertion, suppression	extraction du plus petit
Tableau	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Liste chaînée	$O(n)$	$O(n)$	$O(1)$	$O(n)$
Tableau trié	NA	$O(\log_2(n))$	$O(n)$	$O(1)$
Hashset	NA	$O(1)$	$O(1)$	$O(n)$
ABR	NA	$O(\log_2(n))$	$O(\log_2(n))$	$O(\log_2(n))$
Tas	NA	$O(n)$	$O(\log_2(n))$	$O(\log_2(n))$