

## 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)

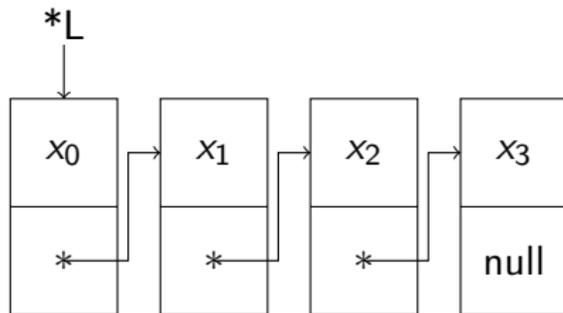
## 2.1 - Liste Chainée

## Liste chaînée

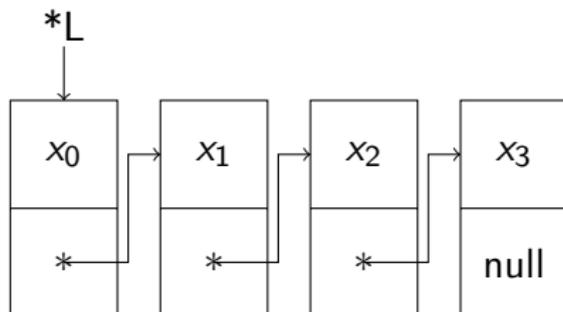
### Principe

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



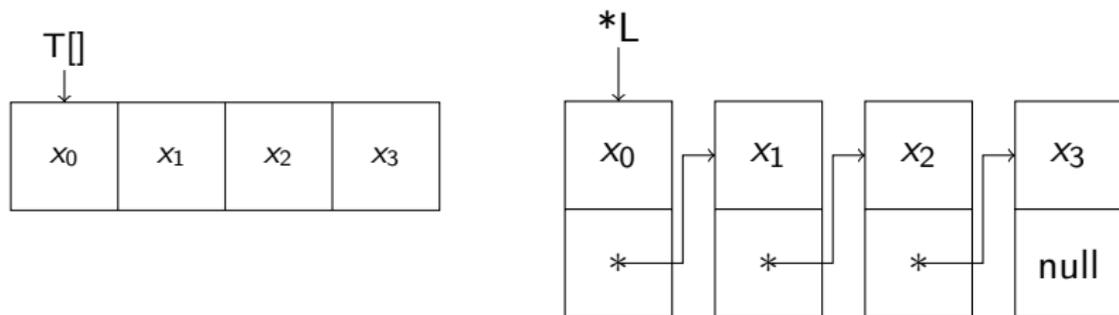
## Pour/contre



- ▶ 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	$O(n)$	$O(n)$	$O(1)$	$O(n)$

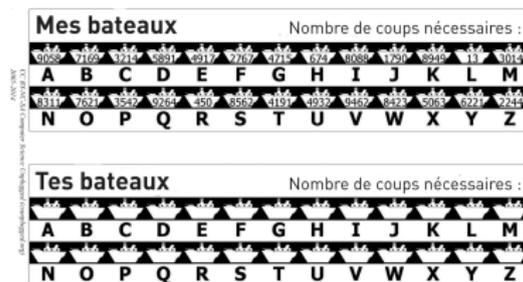
## 2.2. Structures associatives

# 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

**Mes bateaux** Nombre de coups nécessaires :

9058	7169	3214	5891	4917	2767	4715	674	8088	1790	8949	13	3014
<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>	<b>I</b>	<b>J</b>	<b>K</b>	<b>L</b>	<b>M</b>
8311	7621	3542	9264	450	8562	4191	4932	9462	8423	5063	6221	2244
<b>N</b>	<b>O</b>	<b>P</b>	<b>Q</b>	<b>R</b>	<b>S</b>	<b>T</b>	<b>U</b>	<b>V</b>	<b>W</b>	<b>X</b>	<b>Y</b>	<b>Z</b>

**Tes bateaux** Nombre de coups nécessaires :

<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>	<b>I</b>	<b>J</b>	<b>K</b>	<b>L</b>	<b>M</b>
<b>N</b>	<b>O</b>	<b>P</b>	<b>Q</b>	<b>R</b>	<b>S</b>	<b>T</b>	<b>U</b>	<b>V</b>	<b>W</b>	<b>X</b>	<b>Y</b>	<b>Z</b>

# Nombre de tirs

## Tableau Normal

Mes bateaux													Nombre de coups nécessaires :	
 9058	 7169	 3214	 5891	 4917	 2767	 4715	 674	 8088	 1790	 8949	 13	 3014		
<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>	<b>I</b>	<b>J</b>	<b>K</b>	<b>L</b>	<b>M</b>		
 8311	 7621	 3542	 9264	 450	 8562	 4191	 4932	 9462	 8423	 5063	 6221	 2244		
<b>N</b>	<b>O</b>	<b>P</b>	<b>Q</b>	<b>R</b>	<b>S</b>	<b>T</b>	<b>U</b>	<b>V</b>	<b>W</b>	<b>X</b>	<b>Y</b>	<b>Z</b>		

## Structure non organisée

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

## Une autre partie ?

**Mes bateaux** Nombre de coups nécessaires :

												
163	445	622	1410	1704	2169	2680	2713	2734	3972	4208	4871	5031
<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>	<b>I</b>	<b>J</b>	<b>K</b>	<b>L</b>	<b>M</b>
												
5283	5704	6025	6801	7440	7542	7956	8094	8672	9137	9224	9508	9663
<b>N</b>	<b>O</b>	<b>P</b>	<b>Q</b>	<b>R</b>	<b>S</b>	<b>T</b>	<b>U</b>	<b>V</b>	<b>W</b>	<b>X</b>	<b>Y</b>	<b>Z</b>

**Tes bateaux** Nombre de coups nécessaires :

												
<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>	<b>I</b>	<b>J</b>	<b>K</b>	<b>L</b>	<b>M</b>
												
<b>N</b>	<b>O</b>	<b>P</b>	<b>Q</b>	<b>R</b>	<b>S</b>	<b>T</b>	<b>U</b>	<b>V</b>	<b>W</b>	<b>X</b>	<b>Y</b>	<b>Z</b>

## Dichotomie

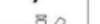
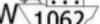
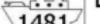
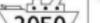
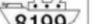
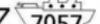
Mes bateaux													Nombre de coups nécessaires :
													
163	445	622	1410	1704	2169	2680	2713	2734	3972	4208	4871	5031	
<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>	<b>I</b>	<b>J</b>	<b>K</b>	<b>L</b>	<b>M</b>	
													
5283	5704	6025	6801	7440	7542	7956	8094	8672	9137	9224	9508	9663	
<b>N</b>	<b>O</b>	<b>P</b>	<b>Q</b>	<b>R</b>	<b>S</b>	<b>T</b>	<b>U</b>	<b>V</b>	<b>W</b>	<b>X</b>	<b>Y</b>	<b>Z</b>	

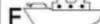
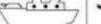
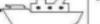
- ▶ Structure indexée.
- ▶ Éléments dans l'ordre.

### Structure triée

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

# Encore une partie ?

Mes bateaux				Nombre de coups nécessaires :					
0	1	2	3	4	5	6	7	8	9
A  9047	C  3080		E  5125	H  8051	L  7116	O  6000	R  9891		W  1062
B  1829	D  9994		F  1480	I  1481	M  8944	P  7432	S  1989	V  4392	X  2106
			G  8212	J  4712	N  4128	Q  4110	T  2050		Y  5842
				K  6422		U  8199			Z  7057

Tes bateaux				Nombre de coups nécessaires :					
0	1	2	3	4	5	6	7	8	9
A 	E 	H 		L 		O 	R 	V 	
B 	F 	I 	K 	M 		P 	S 	W 	Y 
C 	G 	J 		N 		Q 	T 	X 	Z 
D 						U 			

CC BY-NC-SA Computer Science Unplugged (csunplugged.org) 2005-2014

## Recherche par clé

0	1	2	3	4	5	6	7	8	9
			E 5125	H 8051	L 7116	O 6000	R 9891		W 1062
A 9047	C 3080		F 1480	I 1481		P 7432	S 1989	V 4392	X 2106
B 1829	D 9994		G 8212	J 4712	M 8944	Q 4110	T 2050		Y 5842
				K 6422	N 4128		U 8199		Z 7057

- ▶ 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 décorée 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	$O(1)$	$O(1)$	$O(n)$

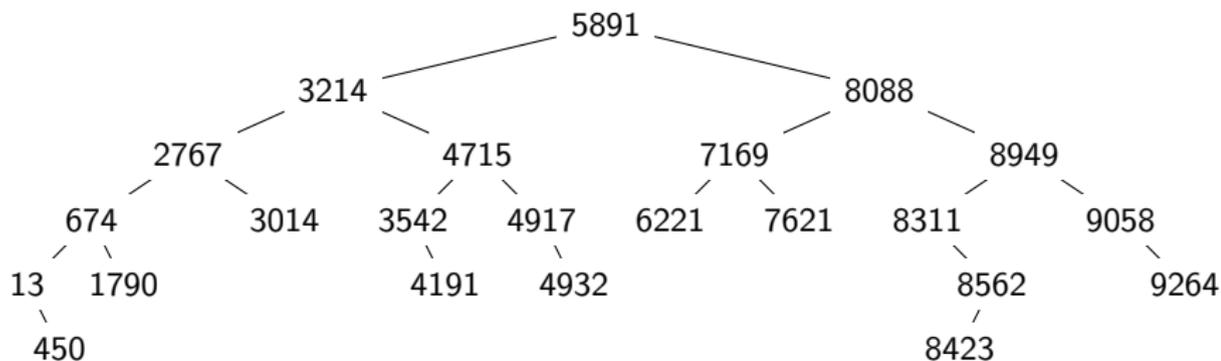
# Comparaison des structures de données

## 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.

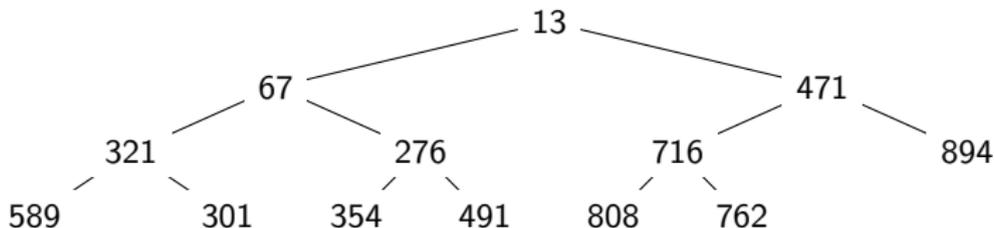
## Usages

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

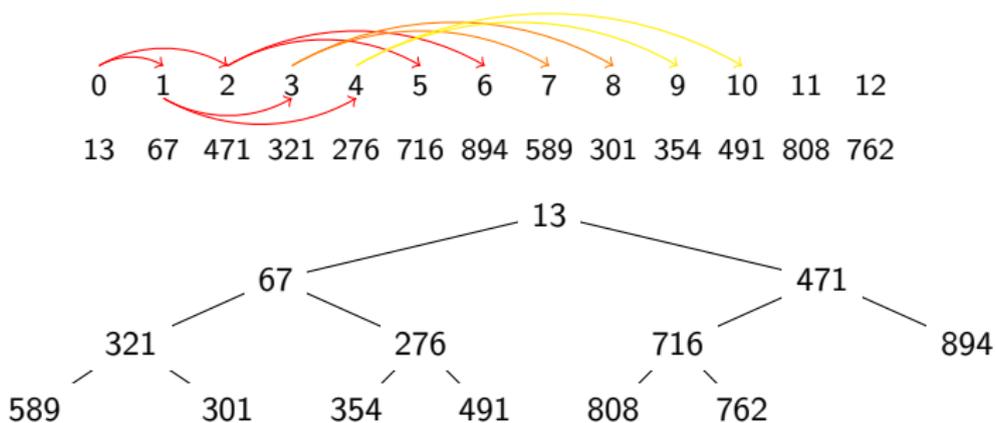
## Les tas

### Définition :

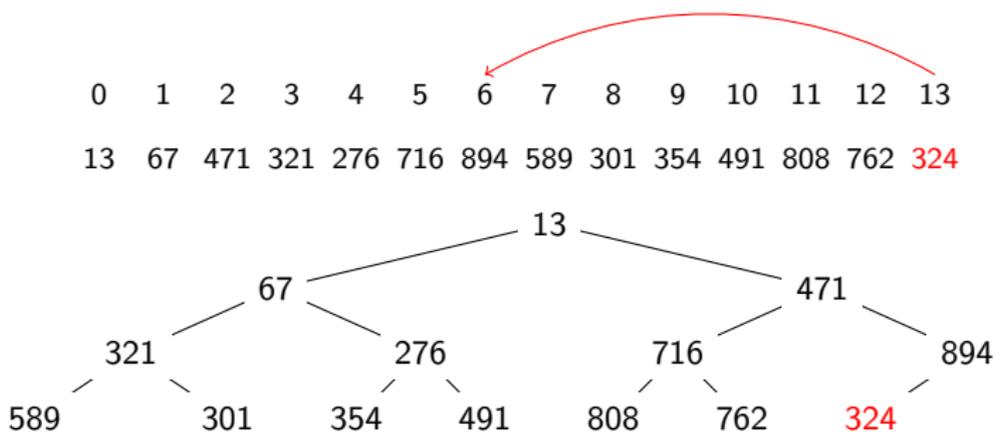
- ▶ Une structure semi-triée rangée dans un tableau
- ▶ Règle du jeu : chaque nœud est plus petit que ses enfants.
- ▶ en Java, implémentée dans les `PriorityQueue`.



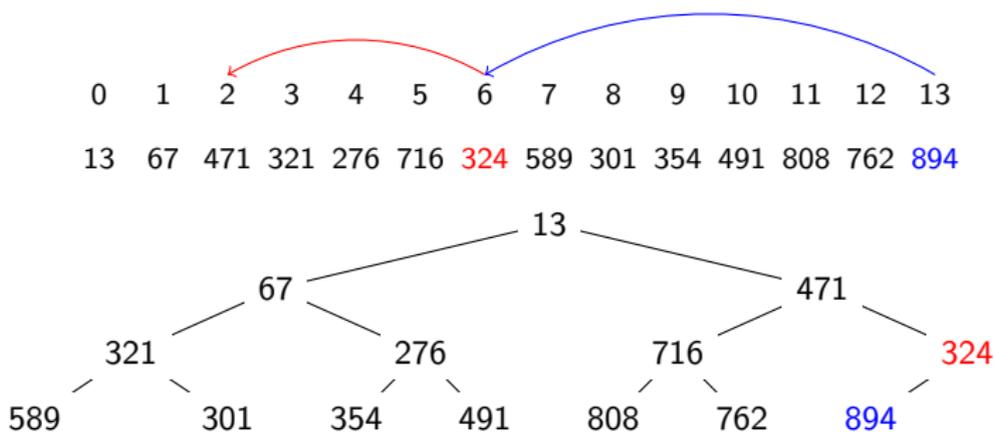
## Insertion dans un tas



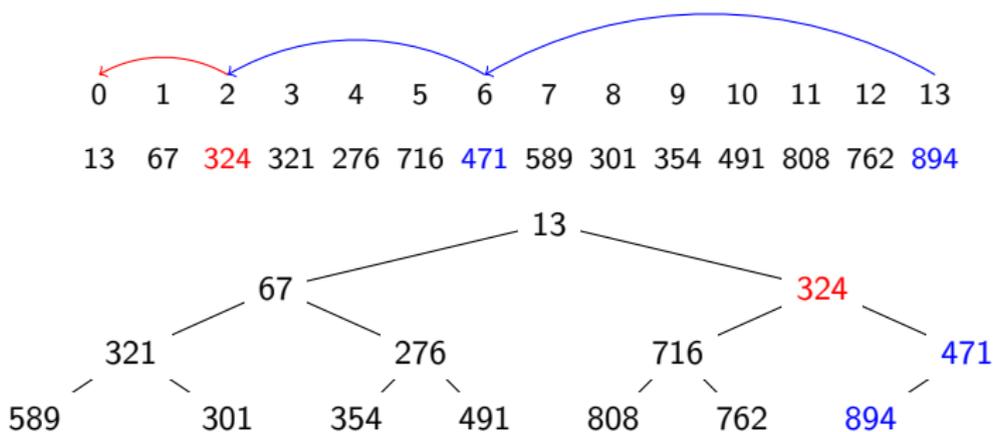
## Insertion dans un tas



## 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	$O(n)$	$O(\log_2(n))$	$O(\log_2(n))$