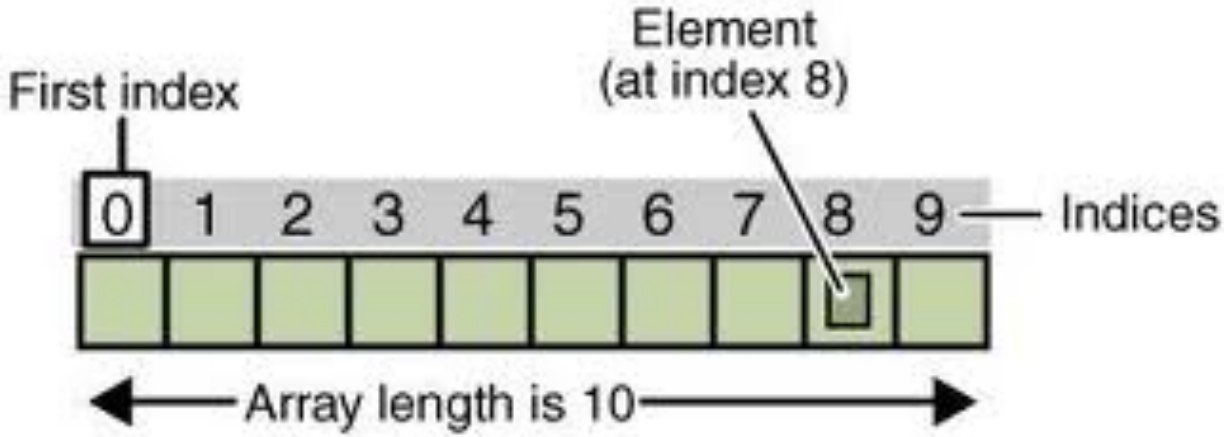
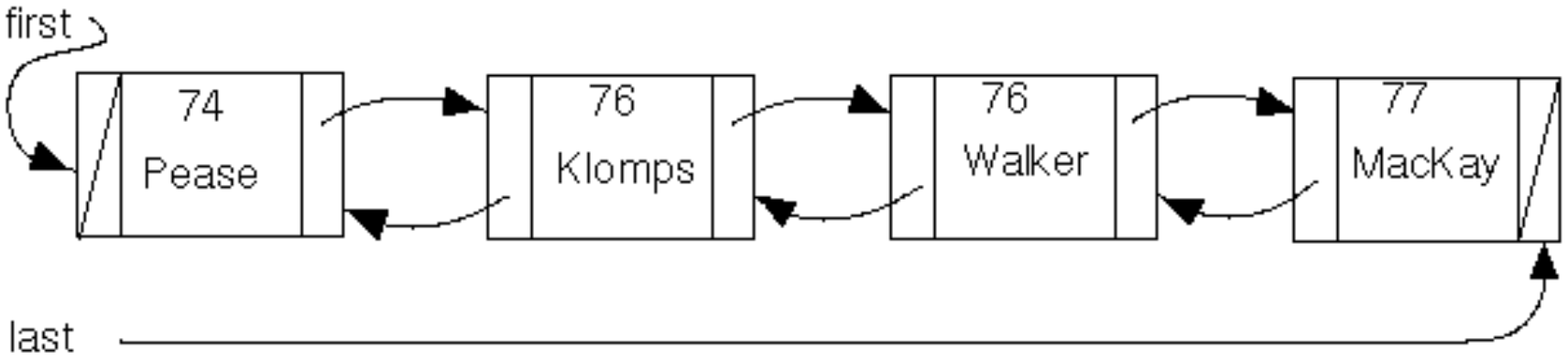
ArrayList



size, isEmpty, get, set : en temps constant (accès à une valeur ou à une case) **O**(1)

add/remove : dans le pire des cas il faut décaler des éléments, dans le pire des cas, on devra décaler tous les éléments (**O**(n))

LinkedList



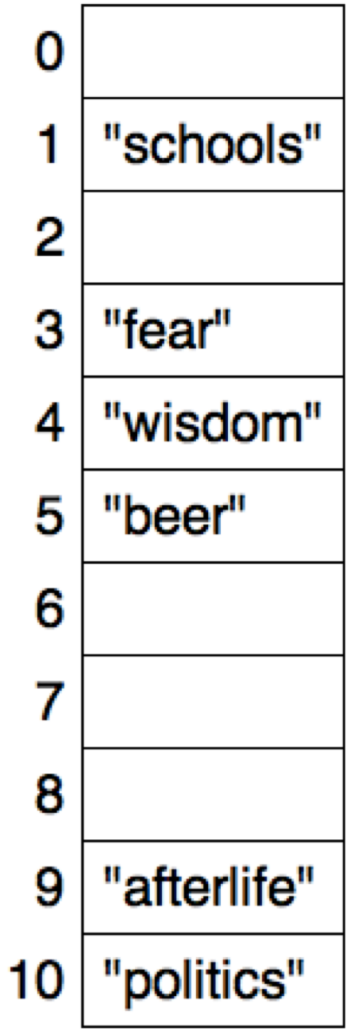
size, isEmpty, add, remove, set, get : temps constant

add(i, elt), get(i), remove(i) : parcours de la liste jusqu'à l'élément i (dans le pire des cas toute la longueur de la liste) en **O**(n)

HashSet (ou HashMap)

élément	hash(element)
"beer"	5
"afterlife"	9
"wisdom"	4
"politics"	10
"schools"	1
"fear"	3

add, remove, contains, size : **O**(1)
considérés comme en temps constant,
accès par la valeur résultat de hash

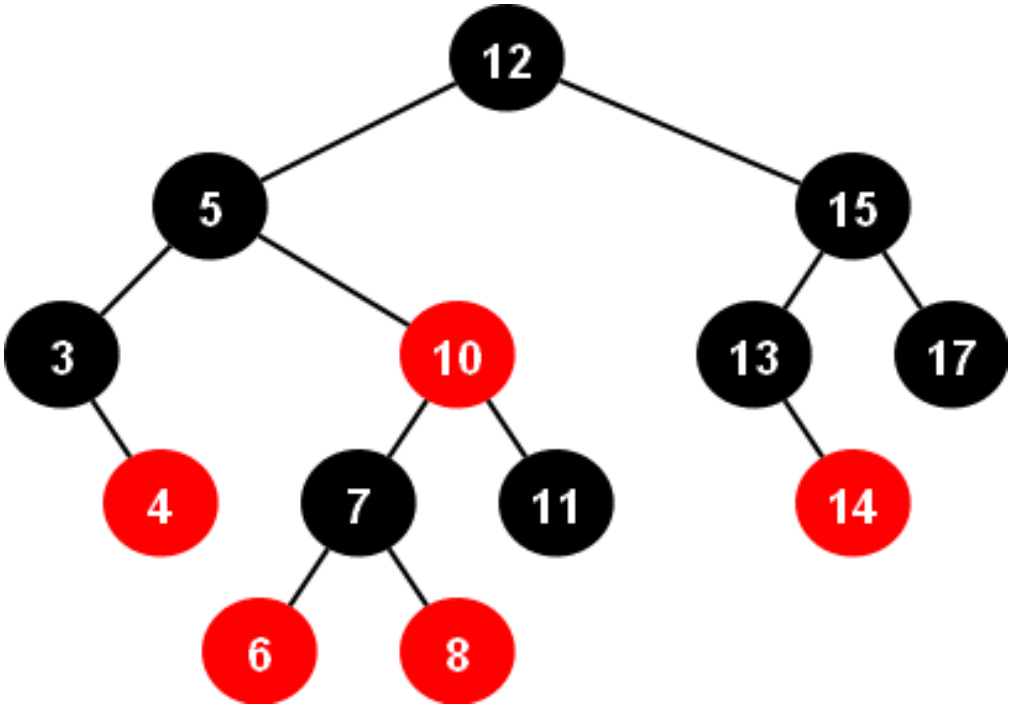


http://cs.lmu.edu/~ray/notes/redblacktrees/

TreeSet (ou TreeMap)

Arbre binaire de recherche (ordonné) équilibré

si n est le nombre de nœuds, la profondeur ne dépasse jamais 2 $\log_2(n)$



add, remove, contains , size : **O**($\log_2(n)$)

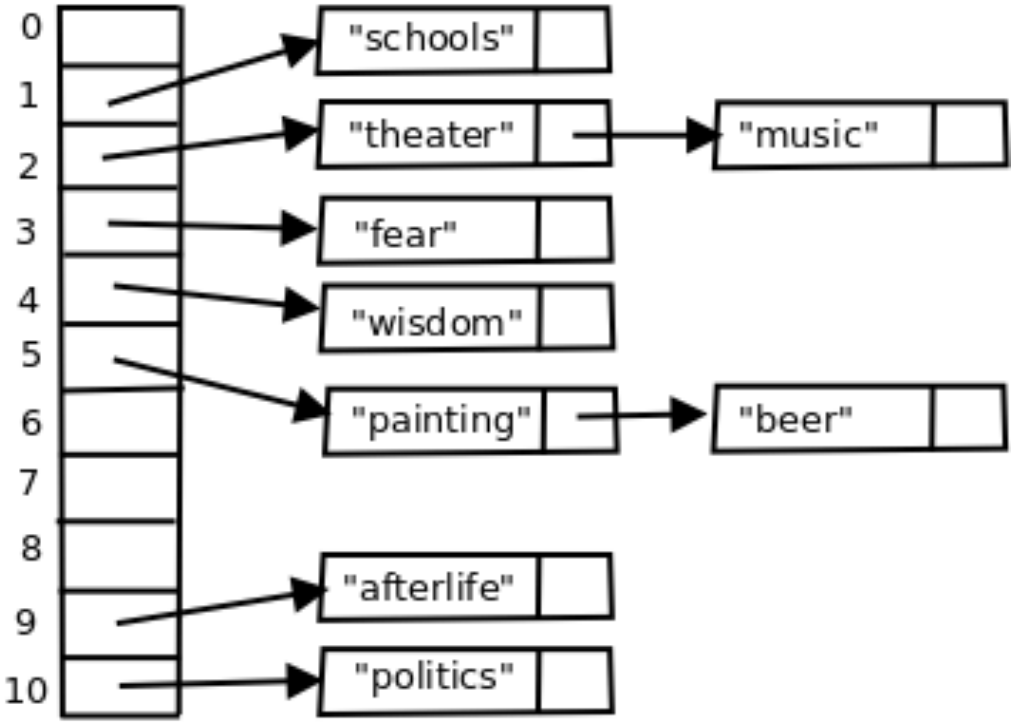
http://www.cs.wcupa.edu/rkline/ds/hash-sets.html

LinkedHashSet

"music"
hash code: 104263205
array index: 2
"beer"
hash code: 3019824
array index: 5
"afterlife"
hash code: 1019963096
array index: 9
"wisdom"
hash code: -787603007
array index: 4
"politics"
hash code: 547400545
array index: 10
"theater"
hash code: -1350043631
array index: 2

"schools"
hash code: 1917457279
array index: 1
"painting"
hash code: 925981380
array index: 5
"fear"
hash code: 3138864
array index: 3

add, remove, contains, size : **O**(1)
considéré en temps constant,
accès par la valeur résultat de hash
et parcours courte liste pour les
collisions



Complexité des opérations en Java

	O(1)	$O(\log_2(n))$	O(n)
ArrayList (tableau)	size, isEmpty, get, set		add/remove (**O**(1) en complexité amortie)
LinkedList (cellules chaînes)	size, isEmpty, add, remove, set, get		méthodes faisant référence à un indice get(i), set(i)
HashSet/HashMap (table de hachage)	add, remove, contains size		
TreeSet/TreeMap red-black tree (arbre binaire de recherche équilibré)		add, remove, contains , size	
LinkedHashSet-Map (tableaux et cellules)	add, remove, contains, size		

Conclusion

- Distinguer **TDA** et **SD**
  - **TDA** : type abstrait de données (spécification représentée par une interface et des assertions)
  - **SD** : structure de données (organisation des données, représentée par des classes)
- Analyser le besoin en opérations, la complexité en place et en temps pour le choix du TDA et de la SD
- Utiliser les outils de Java pour une bonne mise en œuvre
  - généricité, assertions, exceptions, itérateurs, streams (Java 1.8)
  - intégration dans l'API, usage des interfaces et classes existantes