Laboratory practice No. 3: Linked List and Array List

Ana Sofía Gutiérrez
Universidad Eafit
Medellín, Colombia
asgutierrt@eafit.edu.co

Santiago Hidalgo
Universidad Eafit
Medellín, Colombia
shidalgoo1@eafit.edu.co

3. **Practice for final project defense presentation**
   3.1. Complexity values comparison table

	ArrayList	LinkedList
Exercise 1.1	O(n^2)	O(n)

   3.2.
      Exercise 2.1:
         One line is read at a time from the file. The characters of the read line will be added to a linked list according to the given condition. They will be added first or last. The characters of the read line will be added to a linked list according to the given condition
      Exercise 2.2:
         The implementation of this algorithm is especially based on using auxiliary batteries to comply with the specified commands
   3.3.
      Exercise 2.1.
         the complexity is O(m*n)
      Exercise 2.2:
         The complexity is O(n)

   3.4.
      Exercise 2.1:
         n is the number of lines read and m is the length of the String
      Exercise 2.2:
         n is the number of blocks
4. ***Practice for midterms***
   4.1. c) Both have the same asymptotic complexity
   4.2. c) O(n)
   4.3.
      4.3.1. !(q.size()==1
      4.3.2. <=

- 4.3.3. q.remove()
- 4.3.4. q.get(0);

4.4.
- 4.4.1. Lista.size>0
- 4.4.2. Lista.add(auxiliar.pop());

4.5.
- 4.5.1. line12: auxiliar1.size() >0
  line16: auxiliar2.size() >0
- 4.5.2. personas.offer(edad);

4.6. a) O(n3)
4.7. c) O(n)
4.8.
- 4.8.1. c) O(nlogk)
- 4.8.2. c) 12
- 4.8.3. c) O(1)

4.9.
- 4.9.1. a) O(logn)
- 4.9.2. a) 6
- 4.9.3. b) O(n)

4.10.
- 4.10.1. c) O(max(list)×n2)
- 4.10.2. b) O(n)

4.11.
- 4.11.1. !S1.isEmpty()
- 4.11.2. S1.pop()
- 4.11.3. S1.pop()

4.12.
- 4.12.1. iv) 0, 2, 4, 6, 8, 10
- 4.12.2. i) O(1)

4.13.
- 4.13.1. iii) O(n^2)
- 4.13.2. i) O(n)

4.14. iii) 2, 3, 4, 5

5) Data Structures and Algorithms made easy in Java: Linked Lists by Narasimha Karumanchi

Linked lists

Use for storing data where elements are connected by pointers, the last one (tail) points to null and only takes extra memory for pointers. The main operations are insert, delete, count and find a given element One of its advantages is that to expand the linked list we only need to connect another element and not make a copy of the complete array in another with the double size and for disadvantages we need to be really careful when adding or removing elements and make sure the pointer in the previous element it correctly positioned. Another disadvantage is that it takes O(n) time to find an item in a specific index in the worst case compare to the O(1) it takes on an array

Basic operations are reduce to the use and understanding of the pointers system, if you want to insert or delete elements you have to set the pointers carefully so that you not skip any element, besides, when adding items make sure not to delete previous items by removing the pointer connecting the elements you are putting the new one between and when adding/deleting things in the last position (tail) make sure to set its pointer to null

Doubly Linked lists

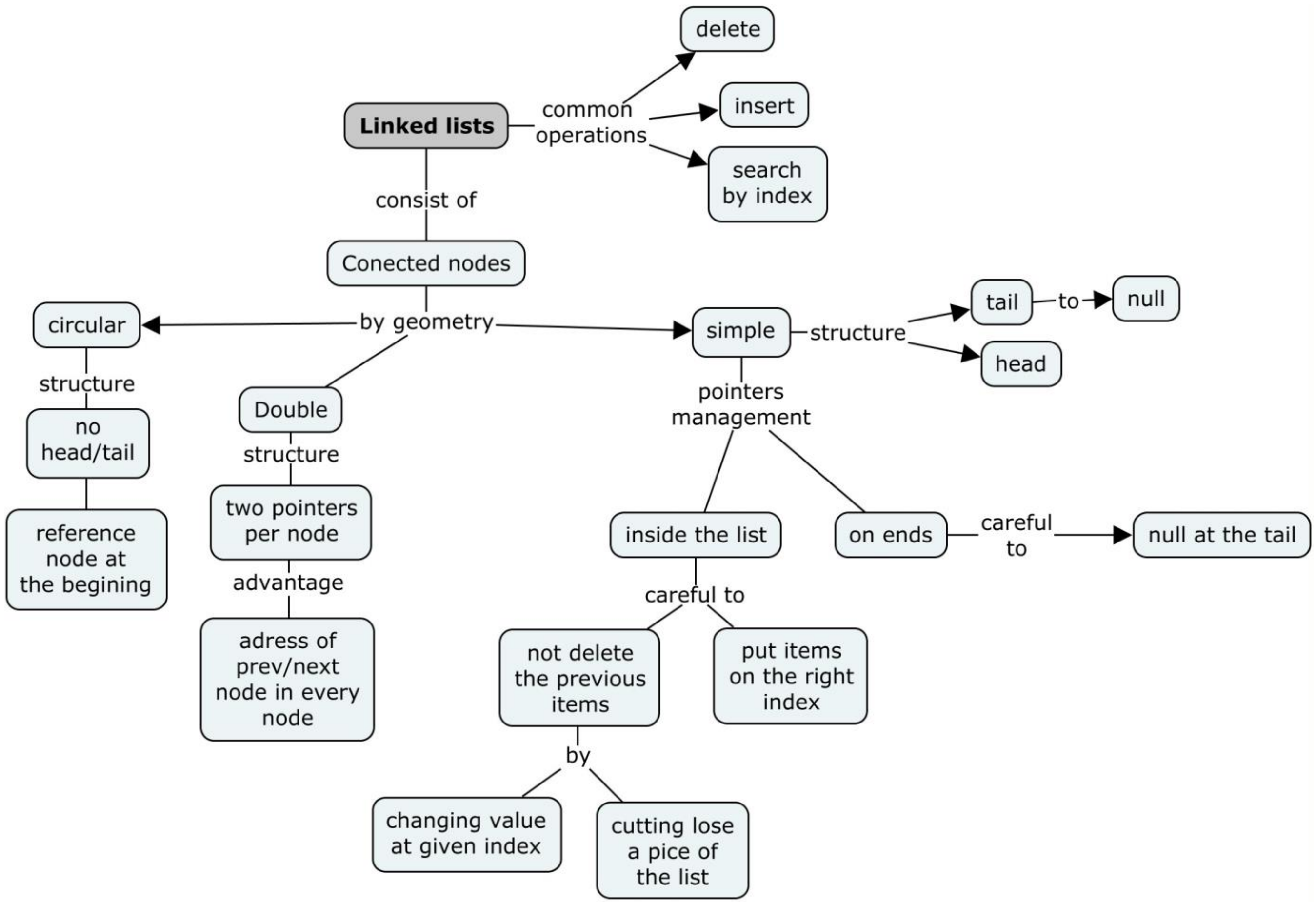
The main difference with simple linked lists is that they have two pointers per element which allows navigate and make operations between them more easily. For example, you don't need he previous node address to delete a node because the item you want to delete already has the information of its previous and next one addresses.

Operations work basically the same, one thing to have in mind is that the first pointer in the first element and the last pointer in the last element must be null, this indicates the head/tail of the list

Circular Linked Lists

This lists are accessed by a node called head and the last item- points towards the head, so it kind of follows a circular geometry which can be useful for certain algorithms but it can also lead to an infinite loop when running along the list so it is important to know the size of the list. Finally, perform any operation in the circular linked lists as if doing it on a simple linked list but never consider to be on the end of the list, which means no pointers will be set to null

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473



PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473