Version v04

October 10, 2021

Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Gopalganj
SHIICT

Assignment CSE201

Data Structure

Md. Kazi Iqbal Hossen
ID: 18ICTCSE065
Dept. CSE, 18-19

Sections & Sub-Sections

List of Figures

1 What is a data structure? What are the types of data structures? What are the notations of measurement of performance of an algorithm?

Data structure is a special way of organizing data in a computer so that we can use those efficiently later. Types of data structures means how the data structure implemented and how it looks like when comparing with the real world, how structured the skeleton that stayed beyond of that data structure, is it linear or non-linear? is it stack type or hierarchical tree type as well.
The two basic types of data-structure:

1. Linear
2. Non-linear

To solve a problem we builds some different logic and algorithm but we don't know actually which algorithm is more efficient for our program in terms of holding less memory and shortest time. That's why we compare algorithms in a manner to find out best algorithmic approach that takes less memory and shortest time to make our program efficient. Due to respect of finding best algorithm there are so many approaches and they are denoted by some kind of mathematical notations. By those notations we measure the performance of an algorithm and scalability.
Some notations are:

1. Big Theta notation $\theta(\ )$
2. Big Ow notation $O(\ )$
3. Big Omega notation $\omega(\ )$

Mostly used notation here is big Ow notation $O(\ )$. Here is a complexity analysis graph via big Ow notation $O(\ )$.

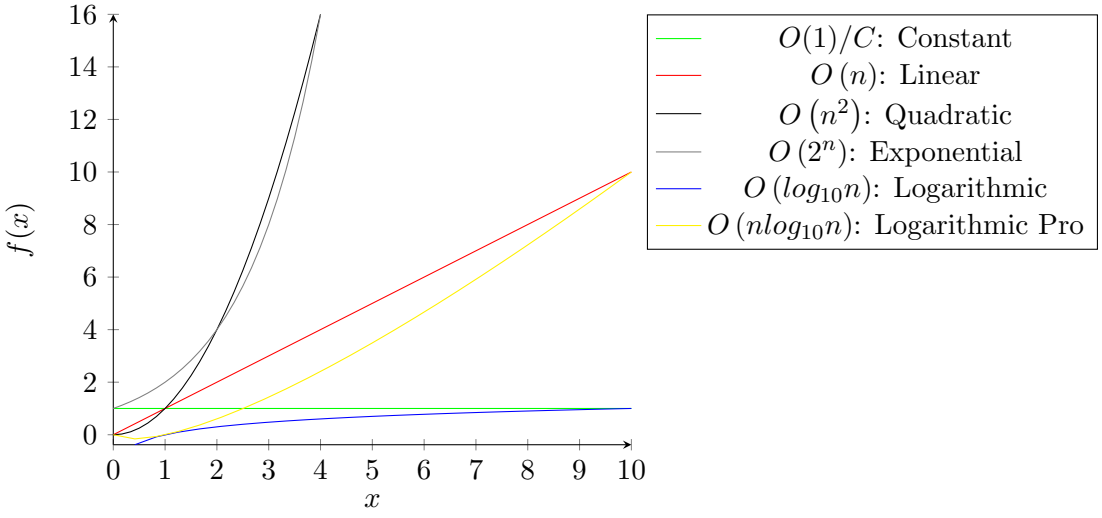


Figure 1: Complexity analysis graph due to respect of BIG O

2 What is bubble sort? Write an algorithm for sorting an array of N elements using bubble sort. Sort this array using bubble sort: [15, 18, 4, 5, 2]

Bubble sort is a very simple sorting algorithm who sort elements by just swapping two adjacent element is they are not in the correct order. Bubble sort has $O(n^2)$ as it's worst-case and average-case time complexity. Here is the algorithm of bubble sort:

Algorithm 1: Bubble Sort: optimized for boolean supported language.

```
Data: An array of N elements.
Result: Sorted list.
begin
    Declare & initialize variable i=0, j=0 and n = length(array).
    Set boolean swapped to false.
    for i from 0 to n-1, increment by 1 do
        for j from 0 to n-i-1, increment by 1 do
            if array[j] is greater than array[j + 1] then
                Swap(array[j], array[j + 1])
                Set swapped value to true.
        if not swapped then
            exit from the loop.
        else
            Set swapped value to false again.
end
```

Sorting the array [15, 18, 4, 5, 2] using bubble sort:

1. After iteration 1: [15, 18, 4, 5, 2]
2. After iteration 2: [15, 4, 18, 5, 2]
3. After iteration 3: [15, 4, 5, 18, 2]
4. After iteration 4: [15, 4, 5, 2, 18]
5. After iteration 5: [4, 15, 5, 2, 18]
6. After iteration 6: [4, 5, 15, 2, 18]
7. After iteration 7: [4, 5, 2, 15, 18]
8. After iteration 8: [4, 5, 2, 15, 18]
9. After iteration 9: [4, 5, 2, 15, 18]
10. After iteration 10: [4, 2, 5, 15, 18]
11. After iteration 11: [4, 2, 5, 15, 18]
12. After iteration 12: [4, 2, 5, 15, 18]
13. After iteration 13: [2, 4, 5, 15, 18]

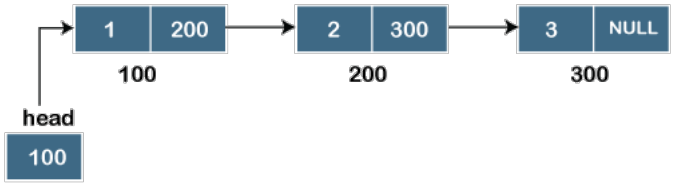
Yep! Sorted using bubble sort.

3 What are the types of linked list? Explain with diagram. Write and algorithm to delete element(node) from "pos" position in linked list

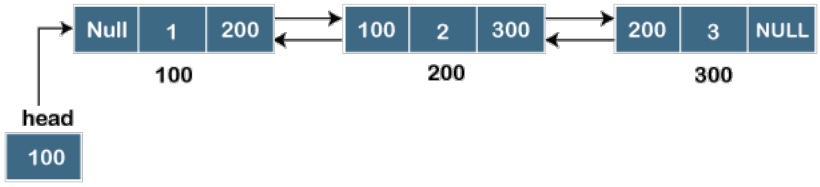
3.1 Type definition with diagram

Types of linked list means looking structure of a linked list.
Types of linked list with diagram:

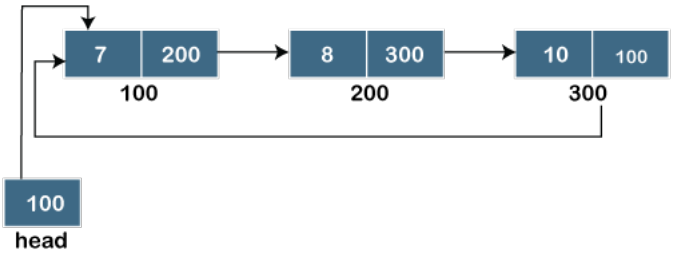
1. Singly linked list, forward iteration only



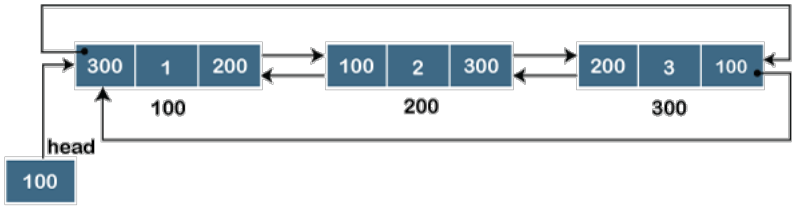
2. Double linked list, can iterate both forward and backward



3. Circular linked list, tail is linked with head, forward iteration only



4. Double circular linked list, tail is linked with head, forward and backward both iteration supported



3.2 Algorithm to delete a node from "pos" index

Algorithm 2: Delete a node from "pos"

Data: Head of the linked list and "pos" of that node which to delete.
Result: Modified list.
begin
  if *head is null or next node of head is null and "pos"* $\geq$ *1* **then**
    Return null with error message "NullPointerException" and terminate the program.
  Declare and initialize prev = head and curr = head.next node respectively
  Now declare and initialize idx = 0 for loop over the linked list
  while *idx+1*$\neq$*pos and curr*$\neq$*null* **do**
    Forward prev node one step, $prev = prev.next$
    Forward curr node one step, $curr = curr.next$
    Increment idx by 1.
  Set prev.next node value to curr.next, $prev.next = curr.next$
  De-allocate space for *curr* node and free the memoey.
  Exit with success code 0.
end

4 Define the properties of complete binary tree. Create a tree from given orders of traversals- Preorder: {11, 5, 3, 8, 16, 14, 18, 17, 20}, Inorder: {3, 5, 8, 11, 14, 16, 17, 18 , 20}

4.1 Properties of complete binary tree

A binary tree is a hierarchical tree type data structure where every node has at most 2 children and in a proper\complete binary tree every internal nodes has exactly 2 child nodes.
Properties of binary tree:

- Special type of binary tree which every level, except possibly the last, is completely filled and all nodes are as far left as possible.
- At each level, there are exactly 2^l child nodes at level 'l' where root is assumed as level 1.
- Height of that tree is $\log(n+1)$ where n is the number of nodes in the tree.

4.2 Constructed binary tree

Preorder: {11, 5, 3, 8, 16, 14, 18, 17, 20}
Inorder: {3, 5, 8, 11, 14, 16, 17, 18 , 20}

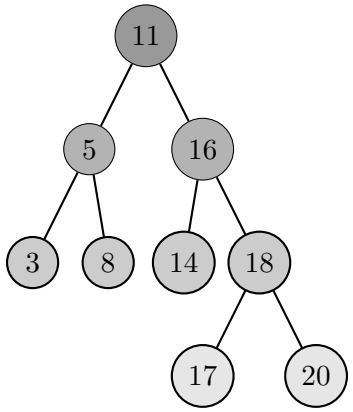


Figure 2: Binary tree with given traversals

5 Convert the following infix equation to postfix equation using stack:

$(A + B \hat{} D)/(E - F) + G$

Serial	Char	Stack	Equation
1	(	(	
2	A	(	A
3	+	(+	A
4	B	(+	AB
5	^	(+^	AB
6	D	(+^	ABD
7	)		ABD^+
8	/	/	ABD^+
9	(	/(	ABD^+
10	E	/(	ABD^+E
11	-	/(−	ABD^+E
12	F	/(−	ABD^+EF
13	)	/	ABD^+EF−
14	+	+	ABD^+EF−/
16	G	+	ABD^+EF−/G

Final postfix equation: $ABD\hat{}+EF-/G+$

6 What is the difference between sequential(linear) search and binary search? Show the steps of searching element '5' using binary search technique in [2, 5, 8, 15, 20]

Linear search is a searching approach that tries to find out an element in a given list\array sequentially from start to end where a binary search approach always grab the middle element in the list and try to match it with the given key, if founds then return its index or recursively do the same approach until left index is not greater than of right index. In binary search, list\array must be sorted.
Searching '5' in [2, 5, 8, 15, 20], steps:

1. Set $key = 5$.
2. First iteration:
   (a) left = 0
   (b) right = length(list) = 5
   (c) $mid = floor\_int\left(\frac{left+right}{2}\right) = 2$
   (d) list[mid] is greater than key. So, $right = mid - 1 = 1$
3. Second iteration:
   (a) left = 0
   (b) right = 1
   (c) $mid = floor\_int\left(\frac{left+right}{2}\right) = 0$
   (d) list[mid] is less than key. So, $left = mid + 1 = 1$
4. Third iteration:
   (a) left = 1
   (b) right = 1
   (c) $mid = floor\_int\left(\frac{left+right}{2}\right) = 1$
   (d) list[mid] is equal to the key. Return mid as index with success code 0.

7 Explain merge sort using an example.

Merge sort is an efficient sorting algorithm that follows Divide and Conquer algorithm to sort the given list of elements. Technique is recursively divide the given list in two halves until we obtain a certain stage where the length of the list becomes only 1. After that, merge those individual arrays in sorted manner till we combines the total list. Here is a pictorial demonstration available that gives a better understanding with an example of list: [38, 27, 43, 3, 9, 82, 10]

Here we take the total list of integers and recursively divide it in two halves until the divided list

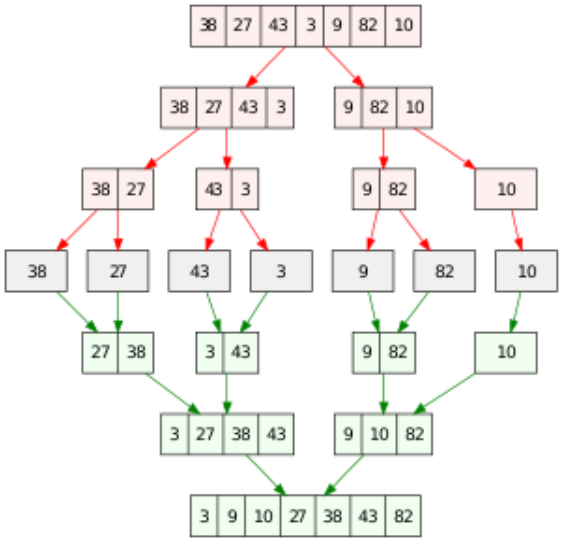


Figure 3: Merge sort illustration

size becomes 1 then contineously merge them in sorted manner till we reach the exact length as before.

Step 1: [38, 27, 43, 3, 9, 82, 10]

Step 2: [38, 27, 43, 3] [9, 82, 10]

Step 3: [38, 27] [43, 3] [9, 82] [10]

Step 4: [38] [27] [43] [3] [9] [82] [10]

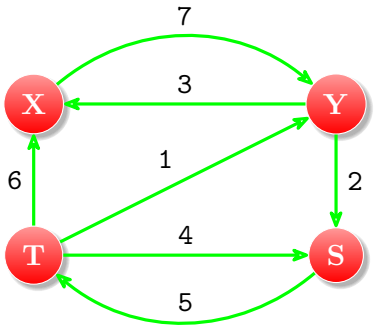
Step 5: [27, 38] [3, 43] [9, 82] [10]

Step 6: [3, 27, 38, 43] [9, 10, 82]

Step 7: [3, 9, 10, 27, 38, 43, 82]

Yep! the list is sorted.

8 Consider the weighted graph G in Fig-1. Suppose the nodes are stored in an array data as follows: [X, Y, S, T]. (a) Find the weight matrix W of G. (b) Find the matrix Q of shortest paths using Warshall's algorithm.



Find out the shortest path of this graph using Warshall's algorithm.

8.1 Weight matrix

$$\text{Weight matrix of G, W} = \begin{Vmatrix} & X & Y & S & T \\ X & 0 & 7 & 0 & 0 \\ Y & 3 & 0 & 2 & 0 \\ S & 0 & 0 & 0 & 5 \\ T & 6 & 1 & 4 & 0 \end{Vmatrix}$$

8.2 Finding shortest path using Warshall's algorithm

$$Q_0 = \begin{bmatrix} 0 & 7 & \infty & \infty \\ 3 & 0 & 2 & \infty \\ \infty & \infty & 0 & 5 \\ 6 & 1 & 4 & 0 \end{bmatrix}$$

$$Q_X = \begin{bmatrix} 0 & 7 & \infty & \infty \\ 3 & 0 & 2 & \infty \\ \infty & \infty & 0 & 5 \\ 6 & 1 & 4 & 0 \end{bmatrix} \qquad Q_Y = \begin{bmatrix} 0 & 7 & 9 & \infty \\ 3 & 0 & 2 & \infty \\ \infty & \infty & 0 & 5 \\ 4 & 1 & 3 & 0 \end{bmatrix}$$

$$Q_S = \begin{bmatrix} 0 & 7 & 9 & \infty \\ 3 & 0 & 2 & 7 \\ \infty & 0 & 0 & 5 \\ 4 & 1 & 3 & 0 \end{bmatrix} \qquad Q_T = \begin{bmatrix} 0 & 7 & 9 & \infty \\ 3 & 0 & 2 & 7 \\ 11 & 6 & 0 & 5 \\ 4 & 1 & 3 & 0 \end{bmatrix}$$

$$\text{Ultimately the shortest path: Q} = \begin{Vmatrix} 0 & 7 & 9 & \infty \\ 3 & 0 & 2 & 7 \\ 11 & 6 & 0 & 5 \\ 4 & 1 & 3 & 0 \end{Vmatrix}$$

Index