The results are in! See what nearly 90,000 developers picked as their most loved, dreaded, and desired coding languages and more in the 2019 Developer Survey.

Time complexity of deletion in a linked list

Ask Question



I'm having a bit of trouble understanding why time complexity of link lists are O(1) according to this website.

From what I understand if you want to delete an element surely you must traverse the list to find out where the element is located (if it even exists at all)? From what I understand shouldn't it be O(n) or am I missing something





data-structures linked-list

asked Nov 29 '15 at 20:15



2 Answers



No, you are not missing something.

If you want to delete a specific element, the time 9 complexity is o(n) (where n is the number of elements)

Home

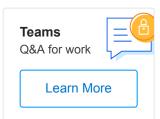
PUBLIC



Tags

Users

Jobs





because you have to find the element first.



If you want to delete an element at a specific index $\, i \,$, the time complexity is $\, o(i) \,$ because you have to follow the links from the beginning.

The time complexity of insertion is only o(1) if you already have a reference to the node you want to insert after. The time complexity for removal is only o(1) for a doubly-linked list if you already have a reference to the node you want to remove. Removal for a singly-linked list is only o(1) if you already have references to the node you want to remove and the one before. All this is in contrast to an array-based list where insertions and removal are o(n) because you have to shift elements along.

The advantage of using a linked list rather than a list based on an array is that you can efficiently insert or remove elements while iterating over it. This means for example that filtering a linked list is more efficient than filtering a list based on an array.

edited Nov 29 '15 at 20:33

answered Nov 29 '15 at 20:22



Paul Boddington

30k

6 45

But say for Singly linked list you do have a reference to a pointer, for example we know the tail, wouldn't you still have to iterate through to list to find the previous node before the tail to update the current tail pointer? — Wolf Nov 29 '15 at 20:27

Yes, that's true. You'd need a reference to the previous node too. I'll edit my answer. – Paul Boddington Nov 29 '15 at 20:28

What you say make sense, but surely it would be at least worst case would be O(n) over O(1) - Wolf Nov 29 '15 at

20:33

Are you talking about removal or insertion? – Paul Boddington Nov 29 '15 at 20:33

1 Ah I see, thank you so much for your help. – Wolf Nov 29 '15 at 20:47



I think he actually means delete head O(1). Otherwise, deleting a specific node would be O(n).

-1

answered Nov 29 '15 at 20:24



user4080725