Singly Linked List - tidskompleksitet

Skemaer – til sammenligning

Skriv noget klogt her!

Singly Linked List er objekter der har en next reference til andre objekter. Den eneste måde vi kan gå vores liste igennem er at starte med det først node objekt, 'head'. Fra 'head' kan vi se dens 'next' og osv osv. Det gør at i værste scenarier at tiden til at iterere igennem listen vil tage O(n), vi kan ikke slå op på et index, med mindre vi kender noden.

Singly Linked List

Læs et element[1]	første	sidste	midterste	i'te	næste[2]
	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$
Find element [3]	eksisterer *usorteret liste*	eksisterer *sorteret liste*	eksisterer ikke *usorteret liste*	eksisterer ikke *sorteret liste*	
	$O(n)$	$O(n)$	$O(n)$	$O(n)$	
Indsæt nyt element	i starten	i slutningen	i midten	efter node	før node
	$O(n)$	$O(n)$	$O(n)$	$O(n)$ $O(1)^*$	$O(n)$ $O(1)^*$
Fjern element	første	sidste	i'te	efter node	før node
	$O(1)$	$O(n)$	$O(n)$	$O(n)$ $O(1)^*$	$O(n)$ $O(1)^*$
Byt om på to elementer	første og sidste	første og i'te	sidste og i'te	i'te og j'te	nodes
	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$ $O(1)^*$

Disse scenarier går ud fra worst case.

*, hvis vi kender node(s)

[1] At læse et element er som regel det samme som at skrive nyt indhold i et eksisterende element
[2] Hvis vi allerede har fat i ét element i en datastruktur, kan vi måske læse det "næste" hurtigere end i+1'te
[3] Find et element med en bestemt værdi – alt efter om vi ved at listen er sorteret eller ej, og om elementet findes eller ej.