Datastrukturer – tidskomplexitet

Skemaer – til sammenligning

Udfyld et skema som det herunder med Big O estimater (kun i tid, ikke i rum) for de datastrukturer du lærer. Skriv også noter til dig selv om nogle af de antagelser du gør dig (for eksempel tager det $O(1)$ at fjerne det sidste element i en arraylist, hvis arrayet ikke kopieres, men $O(n)$ hvis det gør …)

Queue

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

* Det er kun muligt at kigge på det sidste element ved at bruge getIndex, da man ikke direkte har adgang til tail.
** du kan kunne finde det midterste element hvis du ved hvor lang køen er og halverer det. og burger getIndex og size().

*** det er kun muligt hvis køen er tom. Så vil det første element komme ind i starten af køen.

**** kun muligt hvis det er det eneste element i køen.

[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.