Queue- tidskomplexitet

Skemaer – til sammenligning

Nedenstående skema beskriver tidskompleksiteten Big O for en QUEUE.

Men den kan implementeres med en linked liste!

Operationerne kan oversættes direkte:

enqueue → add/addLast

dequeuue → removeFirst

peek → get/first

Det er ej heller et array, men kan implementeres med et array, med enqueue → push og dequeue med shift, eller enqueue → unshift og dequeue → pop

Datastrukturnavn

Læs et element[1]	første	sidste	midterste	i'te	næste[2]
	O(1)	*N/A*	*N/A*	*O(i)**	*N/A*
Find element [3]	eksisterer *usorteret liste*	eksisterer *sorteret liste*	eksisterer ikke *usorteret liste*	eksisterer ikke *sorteret liste*	
	N/A	*N/A*	*N/A*	*N/A*	
Indsæt nyt element	i starten	i slutningen	i midten	Efter node	Før node
	N/A	*O(1)*	*N/A*	*N/A*	*N/A*
Fjern element	Første	sidste	i'te	Efter node	Før node
	O(1)	*N/A*	*N/A*	*N/A*	*N/A*
Byt om på to elementer	første og sidste	første og i'te	sidste og i'te	i'te og j'te	Nodes
	N/A	*N/A*	*N/A*	*N/A*	

[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.