DynamicArray – tidskomplexitet

Skemaer – til sammenligning

DynamicArray er egentlig en implementering af en array-lignende datastruktur med dynamisk vækst.

Til forskel fra en linked list: direkte adgang via index er O(1), men indsættelse/fjernelse midt i array kræver skub af elementer → O(n).

Tilføjelse til slutningen (add) er amortiseret O(1), fordi grow() sker sjældent (hver gang size = capacity, kopieres hele arrayet).

DynamicArray

Læs et element[1]	første	sidste	midterste	i'te	tidligere[2]
	O(1)	*O(1)*	*O(1)*	*O(1)*	*O(1)- hvis index kan vi tilgå forrige*
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(1)*	*O(n)*	*O(n)*	*O(n)*
Fjern element	første	sidste	i'te	efter node	før node
	O(n)	*O(1)*	*O(n)*	*O(n)*	*O(n)*
Byt om på to elementer	første og sidste	første og i'te	sidste og i'te	i'te og j'te	nodes
	O(1)	*O(1)*	*O(1)*	*O(1)*	*O(1)*

[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.