Dynamic Array (Arraylist)– tidskomplexitet

Skemaer – til sammenligning

ArrayList – kort beskrivelse:
En **ArrayList** er en dynamisk liste, der internt bruger et **array** til at gemme elementer. Den vokser automatisk, når der tilføjes flere elementer end der er plads til, og giver hurtig adgang til elementer via indeks.
Fordele (Pros):

- **Random access:** O(1) – Hurtig adgang til elementer via indeks (som i et almindeligt array).
- **Tilføjelse bagerst:** Amortized O(1) – Når listen ikke skal udvides, sker add() meget hurtigt.
- **Iteration:** O(n) – Effektiv sekventiel gennemløb.

Ulemper (Cons):

- **Indsættelse/fjernelse midt i listen:** O(n) – Alle efterfølgende elementer skal flyttes.
- **Udvidelse af array:** O(n) – Når kapaciteten nås, skal et nyt array oprettes og elementerne kopieres.
- **Mindre effektiv til mange ændringer i midten** – Brug hellere en **LinkedList** i de tilfælde.

Dynamic Array(Arraylist)

Læs et element[1]	første	sidste	midterste	i'te	næste[2]
	$O(1)$	$O(1)$	$O(1)$	$O(1)$	O(n) O(1)-hvis vi kender index
Find element [3]	eksisterer *usorteret liste*	eksisterer *sorteret liste*	eksisterer ikke *usorteret liste*	eksisterer ikke *sorteret liste*	
	$O(n)$	$O(\log n)$	$O(n)$	$O(\log n)$	
Indsæt nyt element	i starten	i slutningen	i midten		
	$O(n)$	$O(1)/O(n)$	$O(n)$		
Fjern element	første	Sidste	i'te		
	$O(n)$	$O(1)$	$O(n)$		
Byt om på to elementer	første og sidste	første og i'te	sidste og i'te	i'te og j'te	
	$O(1)$	$O(1)$	$O(1)$	$O(1)$	

[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.