Datastrukturer – tidskompleksitet

Tree

[1]root node, [2]leaf node, [3]vilkårlig node, [4]child node

Læs et element[1]	[1]Første	[2]sidste	midterste	[3]i'te	[4]næste
	O(1)	*n/a*	*n/a*	*O(n)*	*O(1)*
Find element [2]	eksisterer		Eksisterer ikke		
	O(n)		*O(n)*		
Indsæt nyt element	i starten	i slutningen[2]	i midten	efter node[4]	før node
	O(1)	*O(n)*	*n/a*	*O(1)*	*n/a*
Fjern element	første[1]	Sidste[2]	i'te[3]	efter node[4]	før node
	*O(n)**	*O(1)**	*O(n)*	*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
	*O(n)**	*O(n)**	*O(n)**	*O(n)**	*O(n)**

**hvor "n" er antallet af childNodes på element-et/erne*

*** alle nodes længere nede i træet vil også blive fjernet. Ellers skal man flytte alle child nodes fra den slettede node til at være childnodes på den fjernede nodes parent node, og sætte deres parent til denne, og ville så være O(n).*

[1] At læse et element er som regel det samme som at skrive nyt indhold i et eksisterende element
[2] Find et element med en bestemt værdi – alt efter om vi ved at listen er sorteret eller ej, og om elementet findes eller ej.