

Tree – tidskompleksitet

Skemaer – til sammenligning

Tree-struktur

Pros: Fleksibel, simpel API, god til hierarkiske data.

Cons: Ingen balancering, langsommere søgning på store træer, memory-overhead.

Tree

| | første | sidste | midterste | i'te | næste ² |
|-----------------------------|-------------------------------|------------------------------|------------------------------------|-----------------------------------|--------------------|
| Læs et element ¹ | $O(1)$ | $O(n)$ | $O(n)$ | $O(n)$ | $O(n)$ |
| 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 | | |
| | NA | $O(1)$ | ?? | | |
| Fjern element | første | Sidste | i'te | | |
| | $O(n)$ | $O(n)$ | $O(n)$ | | |
| Byt om på to elementer | første og sidste | første og i'te | sidste og i'te | i'te og j'te | |
| | $O(n)$ | $O(n)$ | $O(n)$ | $O(n)$ | |

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.