|  |
| --- |
| Högskolan Dalarna |
| Laboration 2 |
| GMI24H |

|  |
| --- |
| Mikael Olsson  2022-05-04 |

# Muntlig redovisning

<https://www.youtube.com/watch?v=VovYwrJO1o8>

# Syfte

Det skall utvecklas två datastrukturer med de mest grundliga delarna i ett programmeringsspråk för att verkligen förstå och kunna utforska hur dessa fungerar.

# Diskussion

Två datastrukturer har utvecklats, en som baserar sig på en array och en som är en linked list. Tidskomplexiteten visas i följande tabell.

|  |  |  |
| --- | --- | --- |
|  | Datastruktur baserad på array | Linked list |
| AddFirst | O(n) | O(1) |
| Iterate | O(n) | O(n) |
| AddLast | O(1) | O(n) |
| Count | O(1) | O(1) |
| IndexOf | O(n) | O(n) |
| Insert | O(n) | O(n) |
| RemoveAt | O(n) | O(n) |
| Get | O(1) | O(n) |
| Set | O(1) | O(n) |

Båda datastrukturerna använder sig av en count variabel som håller reda på hur lång de är. Det märktes att om man ville ränka varje element i en forech så räknades allt i arrayen (även tomma) och alltid en extra i linkedlist. Då ansågs det vara bättre att lösa det på det sättet som gjordes i laborationen. Något som var märkvärdigt var att arraylist var snabbare på att lägga till något i slutet men seg på att lägga till något i början. Linkedlist var snabbare på att lägga till i början men segare på att lägga till i slutet. En förbättring skulle alternativt kunna vara att hålla reda på slutet och början för att kunna nå dem snabbare.

Arraylist skulle fungera perfekt om man visste hur stor listan var innan man skapar arrayen då det är ganska intensivt att göra arrayen längre. Arraylist är dock oftast mycket bättre än linkedlist då linkedlist kräver mycket minne. Jag läste att skaparen av linkedlist aldrig använde linkedlist då han ansåg att arraylist var bättre i 99% av fallen.

# Kommentarer

Intressant labb.