**Exercise Medelfallstid**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Sökning | Insättning | Borttagning |
| Osorterad vektor | O(n) | O(1) | O(n) |
| Sorterad vektor | O(log n) | O(n) | O(n) |
| Osorteda enkellänkad lista | O(n) | O(1) | O(n) |
| Sorterad enkellänkad lista | O(n) | O(n) | O(n) |
| Hashtabell | O(1) | O(1) | O(1) |

Motivering:

I hashtabellen är storleken lika med antalet element. Då får vi platsen till objektet

direkt, och kan lägga till och ta bort hur som helst direkt. Därför är alla O(1).

Borttagning blir O(n) i alla andra fall eftersom den först måste hitta elementet och det är O(n) förutom i sorterad vektor. Men I en sorterad vektor måste vi också flytta alla element till rätt plats efter borttagning vilket gör det till O(n).

Insättning i listor är O(1) eftersom det spelar ingen roll var vi sätter in så kan vi sätta in först eller sist direkt.

Sökning för sorterad vektor är log n för att vi kan göra binärsökning. Halvera listan och se i vilken halva elementet ligger ger O(log n). I övriga måste vi gå igenom varje element för att hitta rätt vilket ger linjär tidskomplexitet. Eftersom bästa fallet är 1 och sämsta n har vi T(n)=(n+1)/2 =O(n).