**U****PPSALA UNIVERSITET CAMPUS GOTLAND**

Institutionen för informatik och media

Utbildningsprogram: Kandidatprogram i Systemvetenskap (inriktning Programvaruteknik)

Kursansvarig lärare: Thomas Ejnefjäll

Datum: 17e februari 2025

A black and white diagram

Description automatically generated

|  |
| --- |
| Laboration 3 inom Algoritmer och Datastrukter, 7,5 hp    John Sand |

Inledning

I Laboration 3 skapades metoder för att manipulera olika samlingar av heltal med både **iterativa** och **rekursiva** strategier. En metod bör returnera summan av en serie naturliga tal, från ett till den givna gränsen. Den andra metoden returnerar summan av kvadraterna av alla naturliga tal från ett till en given gräns. Det finns flera sätt att skriva metoder för att lösa problem, men olika lösningar har olika effektivitet.  
 Jag jämför och analyserar de *empiriska* tidsresultaten med mina egna tidsresultat för att avgöra vilken tidskomplexitet som är bättre för olika datastrukturer.

## Syfte

Att analysera de empiriska värdena och jämför med mina egna algoritmer från laboration 3 och bestämma hur de handla olika datastruktur.

## Metod

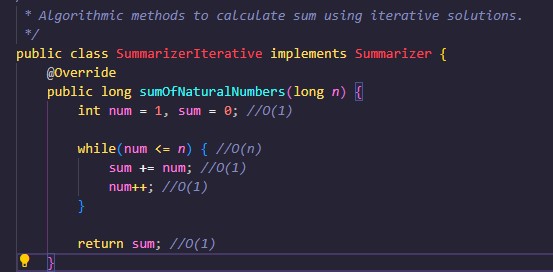
Både de empiriska metoderna och min egen användes på samlingar lagrade som två metodtyper: *Iterativ* och *Rekursiva*. Mina egna metoder tidsstyrdes i effektivitet med klassen *System.nanoTime* i Java-verktygspaketet. Dessa tider omvandlades till millisekunder för att matcha de empiriska resultaten. Sedan kartlades båda tiderna och analyserades för jämförelse.

Resultat

## Metod 1 – sumOfNaturalNumbers

### Iterativa

#### Min metod



En logisk analys av koden för min metod för att beräkna summan av alla naturliga tal från ett till en given gräns skulle resultera i en tidskomplexitet på *O(n)* på grund av behovet att iterera genom hela intervallet från ett till den givna gränsen. Varannan instruktion har en given tid på *O(1)*. På grund av det konstanta behovet av att iterera genom det givna intervallet är både det bästa fallet och det sämsta fallet desamma, *O(n).*

#### Empiriska

A graph with different colored lines

AI-generated content may be incorrect.

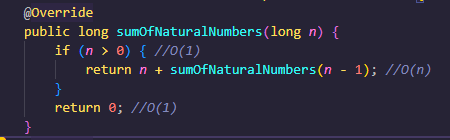
När empirin plottas visar grafen att tidskomplexiteten ligger mellan tidskomplexitet för *O(n)* och *O(n log(n)).* Detta säger mig att det sannolikt itererar genom det givna intervallet med några extra steg. För att bestämma vilken tidskomplexitet som är mest sannolikt, visar en jämförelse mellan skillnaden mellan varje tidskomplexitet och den givna informationen att den är närmare *O(n)*, vilket är vettigt om algoritmen itererar genom intervallet en gång.

A screenshot of a graph

AI-generated content may be incorrect.

### Rekursiva

#### Min metod



Min rekursiva metod anropar sig själv lika många gånger till den givna gränsen för naturliga tal. Vartannat steg har en given tidskomplexitet på *O(1)*. Detta ger min algoritm en Bästa fall-tidskomplexitet på *O(1)* om den angivna gränsen är mindre än 1 och en värsta fall-tidskomplexitet på *O(n).*

#### Empiriska

A graph with different colored lines

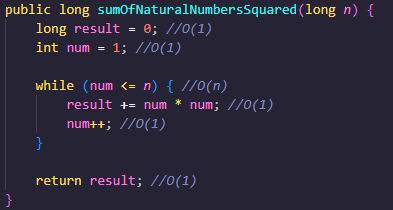
AI-generated content may be incorrect.

Den empiriska data visar en liknande tidskomplexitet som min egen algoritm med denna datastruktur. Tidskomplexiteten på en graf faller något över men följer lutningen för *O(n),* vilket är vettigt för mig. Det verkar inte finnas något sätt att kortväga lägga till summorna, ens med en rekursiv metod.

## Metod 2 – sumOfNaturalNumbersSquared

### Iterativa

#### Min metod



Min algoritm använder liknande logik som min andra iterativa metod, med en *While*-loop för att iterera från en tills den övre gränsen nås. Vartannat steg har en tidskomplexitet på *O(1)*. Detta ger den en tidskomplexitet i bästa fall av *O(1)* om den angivna gränsen är 1 eller mindre, annars har den mycket mer sannolika tidskomplexiteten av *O(n)*.

#### Empiriska

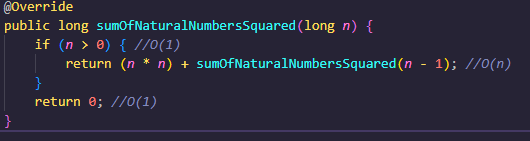
A graph with different colored lines

AI-generated content may be incorrect.

De empiriska uppgifterna var lite överraskande här när de kartlades. Den fortsätter att sitta mellan kurvorna för *O(n)* och *O(n log(n))* som den andra iterativa metoden, men den verkar närmare kurvan för *O(n log(n))* den här gången. Detta säger mig dock att den återigen bara itererar genom det givna intervallet en gång, men förmodligen har några extra steg som ökar tidskomplexiteten en aning.

### Rekursiva

#### Min metod



Min rekursiva metod kallar sig återigen för att börja vid den högre gränsen som ges tills den når 0. Detta ger den en tidskomplexitet i bästa fall på *O(1)* om 1 eller mindre anges som gräns, och ett genomsnittligt fall av *O(n)* eftersom den kallar sig själv ett antal gånger lika med den givna gränsen.

#### Empiriska

A graph of numbers and numbers

AI-generated content may be incorrect.

Denna metods tidskomplexitet är mycket överraskande, eftersom den varken följer *O(n)* eller *O(n log(n)).* Det verkar faktiskt ha en brantare kurva än *(n log(n))* tidskomplexitet, vilket kan innebära flera rekursiva anrop varje gång? Det är inte så brant att det är *O(2^n)* som i den fruktade *fibonacci*-rekursiva metoden, men det är alarmerande att tidskomplexiteten stiger brant med en högre gräns att arbeta med. Den närmaste tidskomplexitetskurvan är *O(n log(n))* med extra steg i några av de rekursiva anropen men inte alla.