# In3030, oblig3

## Introduction

Denne rapporten beskriver hvordan jeg har gjort den sekvensielle «Seven» parallell ved bruk av tråder, samt hvordan jeg finner faktoriene av de 100 største som er mindre enn n*n, i parallell og sekvensielt format.

## Hvordan kjøre programmet

Programmet kjøres ved å først kompilere med javac *java, og deretter kjøre med java Main <n> <k>.

For eksempel: java Main 10000 12

## Parallel Sieve of Eratosthenes

Jeg var I starten veldig forvirret over hvordan jeg kunne gjøre løsningen parallell. Men etter å ha lest litt på nettet, tror jeg at jeg har en grei løsning. Det første jeg gjør når algoritmen starter er å hente de første primtallene opp til roten av N. Om jeg gjør denne lille sekvensielle delen først, så blir den parallelle delen langt simplere, og krever langt mindre kommuniksasjon mellom trådene og hovedtråden.

Dette gjøres på en bakgrunns tråd, mens hovedtråden fordeler 0-N mellom trådene. Når bakgrunns tråden er ferdig, kan trådene starte. De har da tilgang til de primtallene som forekommer før roten av N, og kan dermed sile ut alle tall som ikke er primtall fra sitt intervall. Når alle trådene returnerer til hovedtråden, er alle ikke-primtall fra 0-n markert. hovedtråden kan deretter enkelt iterere gjennom å hente alle tall som ikke er markert. Jeg bruker relativt simple synkroniserings funksjoner, i form av å dele N opp i intervaller samt bruk av en syklisk barriere, for å få hovedtråden til å vente til alle tråder er ferdige å markere sitt intervall. Ved å bruke en slitt blokk-distrubusjon av problemet, slipper trådene å kommunisere med hovedtråden over hvor den skal arbeide til enhver tid, da intervallet alltid er det samme. Siden vi henter de initielle primtallene først, så slipper trådene også å vente på at hovedtråden skal supplere disse fortløpende. Trådene kan altså jobbe uten å tenke over andre faktorer. Jeg syns likevel jeg får lite speedup før tallene blir relativt enorme.

## Parallel factorization of a large number

Jeg slet med denne, og skjønner ikke hvordan jeg kan parallellisere en faktorisering mer enn jeg har gjort. Jeg starter en tråd for hvert tall, og kan dermed kjøre flere faktoriseringer på likt. Jeg skjønner ikke hvordan jeg skal dele opp problemet i flere deler enn dette, men jeg opplever likevel gode speedups. Hver tråd tar seg av faktoriseringen av et tall, og hovedtråden venter til alle tallene er ferdig faktorisert. Når tallene er ferdig faktrosierit legges resultatene i en 2d array, som returneres. Jeg bruker en sykliskbarriere for å få hovedtråden til å vente før den returnerer.

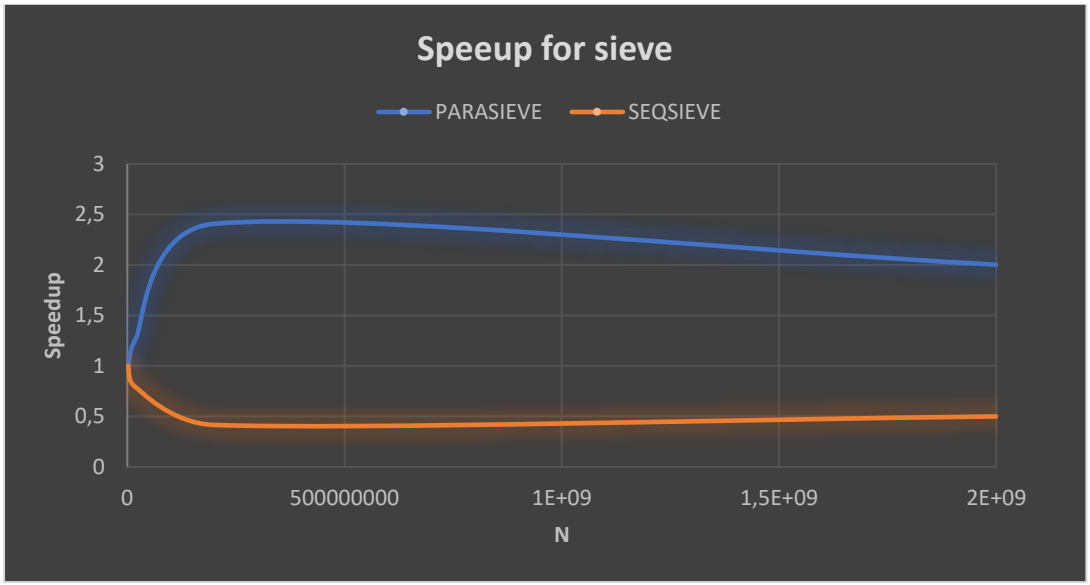
## Tests

For å teste sjekker jeg egentlig bare om output fra den sekvensielle og paralllele løsningen er identisk, og det er det, så antar jeg at alt stemmer. Faktoriserings output skrives til fil.

## Resultsater

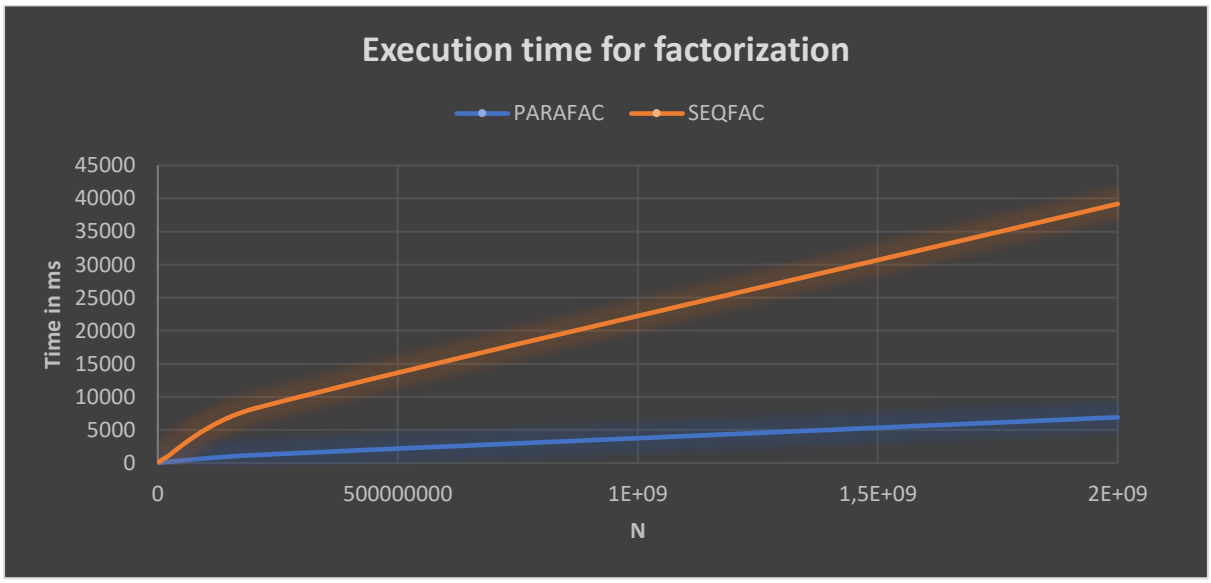
På sieven så jeg relativt greie resulater, men ikke noe kjempe store sprik fra den sekvensielle. Jeg syns, fortsatt at det går litt tregt, men nå skal det sies at jeg kjører disse testene på en laptop. Som du kan se ut fra grafene er speedupen rundt 2, og begge grafene vokser tilnærmet likt tempo. Hadde jeg hatt mulighet til å teste tall større enn 2mrd, så kanskje vi hadde sett en større speedup, men den ser ut til å stabilisere seg på rundt 2x.

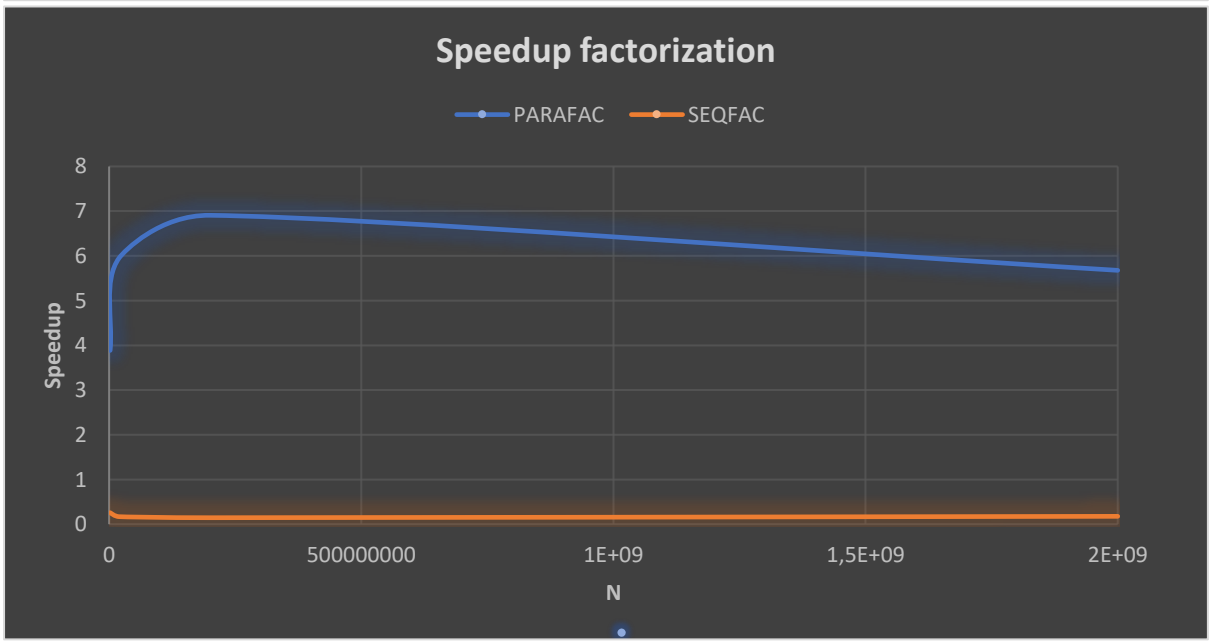
| N | PARASIEVE | SEQSIEVE |
|---|---|---|
| 2000000 | 10ms | 10ms |
| 20000000 | 81ms | 103ms |
| 200000000 | 668ms | 1608ms |
| 2000000000 | 7797ms | 15609ms |

På faktoriseringen så jeg langt større grad av vinning med bruk av paralelliseirng av problemet. Som du kan se under, så løses problemet veldig mye raskere ved bruk av tråder. Maks speedup er rundt 7, men stabiliserer seg på rundt 6, som må sies å være enn svært god speedup. Likevel tror jeg denne kunne vært enda høyere, om jeg hadde fått til å dele problemene opp i enda mindre deler, og fordelt det videre.

| N | PARAFAC | SEQFAC |
|---|---|---|
| 2000000 | 35ms | 136ms |
| 20000000 | 162ms | 965ms |
| 200000000 | 1188ms | 8206ms |
| 2000000000 | 6903ms | 39185ms |



## Konklusjon

For å konkludere kan jeg si at jeg er fornøyd med å ha klart å få over 1 i speedup på begge algoritmene, samt at algoritmene aldri bruker mer enn 30 og 60 sekunder, slik som oppgaven sier de ikke burde. Jeg tror jeg kunne sett noe økning i speedup om jeg testet på min stasjonære pc, da denne har flere og raskere kjerner enn laptopen. Laptopen min har forresten 6 fysiske kjerner, på 2.3 ghz.