Ukeoppgaver i uke 5, INF2440 – v2017

I neste Oblig 2 skal du greie å parallellisere det å generere alle primtall < N med en teknikk som er hentet fra jernalderen, fra en gresk matematiker som heter Eratosthenes (ca. 200 f.k.) og så faktorisere alle tall M < N*N med disse primtallene. Metoden ble presentert i forelesningene i Uke5 (se foilene) , og du kan også lese om den på Wikipedia.no (se: http://no.wikipedia.org/wiki/Eratosthenes'_sil) hvor metoden også er visualisert. Eratosthenes sil nyttes fordi den faktisk er den raskeste. Det eneste avviket vi gjør fra slik den er beskrevet i Wikipedia er flg:

1. Vi representerer ikke partallene på den tallinja som det krysses av på fordi vi vet at 2 er et primtall (det første) og at alle andre partall er ikke-primtall.
2. Har vi funnet et nytt primtall p, for eksempel. 5, starter vi avkryssingen for dette primtallet først for tallet p*p (i eksempelet: 25), men etter det krysses det av for p*p+2p, p*p+4p,.. (i eksempelet 35,45,55,…osv.). Grunnen til at vi kan starte på p*p er at alle andre tall < p*p som det krysses av i for eksempel Wikipedia-artikkelen har allerede blitt krysset av andre primtall < p. Det betyr at for å krysse av og finne alle primtall < N , behøver vi bare og krysse av på denne måten for alle primtall p ≤ sqrt(N). Dette sparer mye tid.

Sammen med ukeoppgavene ligger det en .java-fil: EratosthenesSil.java, som inneholder skjelettet til en klasse som du kan nytte til å implementere Eratosthenes sil. Selvsagt står du helt fritt til å implementere den på en annen måte hvis du vil det, men husk da at du skal ha plass til å ha representert alle primtall < 2 milliarder i den, og at ca 5-10% av alle tall er primtall (mer eksakt : det er omlag $\frac{N}{\log N}$ primtall < N) .

Ukeoppgave: I filen EratosthenesSil.java er det tre metoder: crossOut, nextPrime, isPrime som du skal implementere i Uke 5. Lag så et lite testprogram som lager et objekt av klassen EratosthenesSil, for maxNum = 100, og som så kaller printAllPrimes() så du kan sjekke at du har skrevet riktig kode.

Du skal så, når testen går OK, ta tidene så hvor lang tid i millisek. det tar å lage alle primtall med maxNum =2mill, 20 mill, 200 mil og 2000 mill. Skriv en liten rapport om det, og bruk medianen av 9 tall for å finne en 'god' verdi for kjøretiden for hver av disse valgene av maxNum.

(I neste uke skal du skrive metoden factorize(long num) som returnerer en ArrayList med de ulike faktorene i faktoriseringa av 'num'. Da har du fått en komplett sekvensiell algoritme for både å lage de primtall du trenger og faktorisere 19-sifrete tall med disse primtallene sekvensielt. I obligen skal vi se på parallellisering av disse to algoritmene: a) Generering av N primtall og b) Faktorisering av alle tall < N*N)