Projekt i EDAA35
Jämförelse av exekveringstider av C++ och Java

Samuel Rosenqvist, sa7638ro-s@student.lth.se, Rasmus Olsson, ra0011ol-s@student.lth.se,
Douglas Tönnesen, do3718to-s@student.lth.se

***Sammanfattning*—Rapporten testar och jämför skillnader i prestation mellan programmeringsspråken Java och C++. Testen utfördes på språkens standardsorteringsalgoritmer, mergesort och en algoritm för att ta fram de n första primtalen. Java presterar bättre i samtliga test.**

I. Inledning

DENNA rapportens syfte är att undersöka och diskutera eventuella skillnader i exekveringstid mellan vanliga algoritmer implementerade i Java och C++. Jämförelsen kan underlätta valet av programspråk för en programmerare. Rapporten behandlar relativt grundläggande algoritmer och bör inte användas för att ta beslut angående valet till större program och projekt.

5 maj, 2017

II. Bakgrund

Programspråk skiljer sig åt på flera sätt och det är därför nödvändigt att jämföra skillnaderna för att kunna fatta beslut angående vilket språk som är bäst lämpat för den givna situationen. Genom att jämföra exekveringstid kommer man ett steg närmare ett beslut. Exekveringstid är inte alltid högsta prioritet men när man väljer mellan språk som C++ och Java så spelar det ofta stor roll.

III. Frågeställning

Programspråken som jämförs är Java och C++.

- Vilket programspråks standard sorteringsalgoritm har lägst exekveringstid?
- Vilket programspråk exekverar en implementation av merge sort snabbast?
- Vilket programspråk exekverar en metod för att beräkna de första 1 000 000 primtalen snabbast?

IV. Metod

Algoritmer som jämförts är C++ std::sort och Javas Collections.sort, samt implementering av merge sort och även beräkning av de första en miljon primtalen. För att beräkna primtal användes en metod att testa varje tal genom att dela det med tal från 2 upp till roten av varje tal. Exekveringstiden är uppmätt i programmen med Javas System.nanoTime och i C++ med high_resolution_clock från chrono biblioteket. Jämförelser och grafplottar har gjorts i R. Sorteringarna har som indata använt data1.txt från lab5 (800 st osorterade tal) och har exekverat sorteringen 600 gånger.

V. Validitetsdiskussion

Eftersom vår C++ kunskap ej är på samma nivå som vår Java är det möjligt att C++ inte utnyttjats till sin maximala potential. För att motverka detta har vi försökt använda implementeringar skrivna av vanare C++ användare.
Då vi ej har tillräcklig kunskap inom C++ kunde vi inte göra jämförelser på längre och mer komplicerade program, utan fick nöja oss med relativt enkla program vilket kan påverka resultaten. Vi kan inte heller garantera att det inte skett några minnesläckor vid exekveringen av C++ programmen, flera mätningar gjordes men hotet kan möjligen ha påverkat resultaten till en viss del.
För C++ har kompilatorn g++11 använts, det är möjligt att en nyare kompilator hade givit andra resultat men för projektets ändamål är resultaten nöjaktiga.

VI. Resultat

Tabell I
Exekveringstider

Program	Medelvärde av exekveringstid
Java Collections.sort	99148 ns
C++ std::sort	119001 ns
Java Merge sort	28479 ns
C++ Merge sort	76960 ns
Java primtal	16707207454 ns
C++ primtal	34979412130 ns

Enligt Welch t.test i R är Javas merge sort mellan 42025 ns och 54936 ns snabbare än C++ merge sort. Javas standardsortering är mellan 55008 ns och 94715 ns snabbare än C++ standardsortering.

Vid körning av t.test på resultaten av de båda tidigare nämnda sorteringar i samma språk var Javas mergsort snabbare i genomsnitt medan standardsorteringen var snabbare i enskillda fall. I C++ var mergesort alltid snabbare.

VII. Diskussion

Enligt erhållna resultat så är Java snabbare än C++. Vid standardsorteringarna var Java ca 1,2 gånger snabbare (se tabell I) och vid mergesort var Java ca 2,7 gånger snabbare. Dessa resultat var väldigt förvånande för oss då vi förväntade oss att C++ skulle vara snabbare. Enligt lite

egen forskning innan projektets start har vi läst att C++ fortfarande är snabbare än Java. Det är mycket möjligt att C++ är snabbare än Java bara att testerna som kördes gav Java en fördel.
Medelvärdena är över alla 600 iterationer, både C++ och Java har ett visst insvängningsförlopp som kan ses i Figur 1 och 3 respektive Figur 2 och 4, de båda Java metoderna når ett jämnviktsläge runt iteration 300 medan C++ metodernas exekveringstid fortsätter att sjunka under hela förloppet. För att försöka vara rättvisa har medelvärdet beräknats som det gjort, programspråkens effektivitet gäller både uppstarten och jämnviktsförloppen.
Våra tester var väldigt primitiva och kan ej tolkas som ett direkt svar till vilket språk som är snabbast vid t.ex. 3D-modellering. Resultaten pekar däremot på att Java är ett väldigt bra språk att använda när man ska utföra mindre beräkningar av t.ex. primtal eller sorteringar. När Java och C++ fick nästan identisk kod (Fast på de olika språken) där de skulle räkna ut de första 1 000 000 primtalen var Java ca 2 gånger snabbare. C++ är ett hårdvarunäraspråk vilket gör att bättre optimering för det använda systemet kan göras, detta har dock inte gjorts i testerna vilket också kan vara en anledning till C++ dåliga resultat gämtemot Javas. Värt att notera är att Java kommer bli lättare att optimera för alla system då Java använder sig av JVM (Java virtual machine) och kan därför ses som bättre vid implementering av enkla algoritmer.

VIII. Slutsatser

Enligt våra mätningar så kan man dra slutsatsen att Java presterar bättre än C++ i samtliga jämförelser. Vid beräkning av primtal är Java överlägset i våra mätningar trots att i stort sett identiska metoder använts.
Vid exekvering av merge sort är Java också bra mycket snabbare än C++. Dock är skillnaden i exekveringstiden liten mellan Javas och C++ standardsorteringar, eftersom det antagligen är den som oftast används i praktiska sammanhang kan argumentet göras att programspråken står på någorlunda lika nivå.

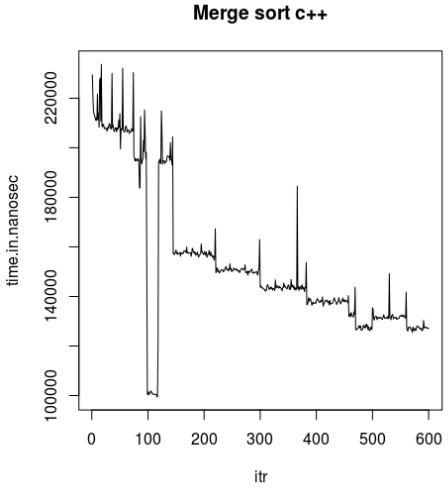
Bilaga A
Grafer

Bilaga B
Kod

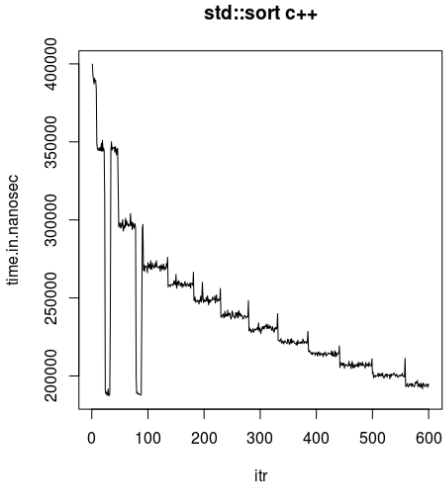
Koden har utelämnats för att spara utrymme men kan finnas på https://github.com/Samulito/edaa35projekt.

Referenser

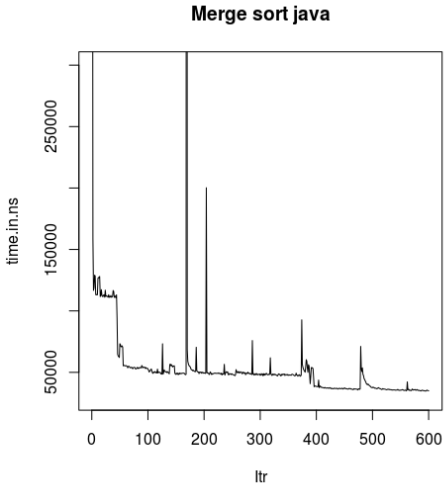
[1] http://quiz.geeksforgeeks.org/merge-sort/ *Implementation av Merge sort i C++.* Hämtad 4 maj, 2017.
[2] http://www.vogella.com/tutorials/JavaAlgorithmsMergesort/article.html *Implementation av Merge sort i Java.* Hämtad 4 maj, 2017.



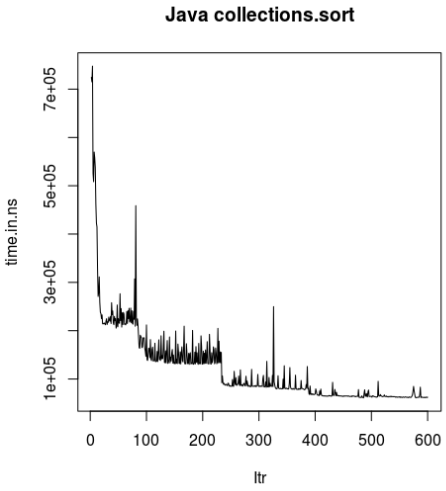
Figur 1. Mergesort i C++



Figur 2. std::sort i C++



Figur 3. Mergesort i Java



Figur 4. Collections.sort i Java