Laborationsdata

Namn: Thomas Nordenmark
LoginID: thno1200
Lab nr: 4
Datum: 2014-02-17
Utvecklingsmiljö: Arch Linux x64, QT Creator IDE 2.8.1, GCC 4.8.2

Filer

arraysort.cpp (huvudprogram)
intarray.cpp
intarray.h
timer.h

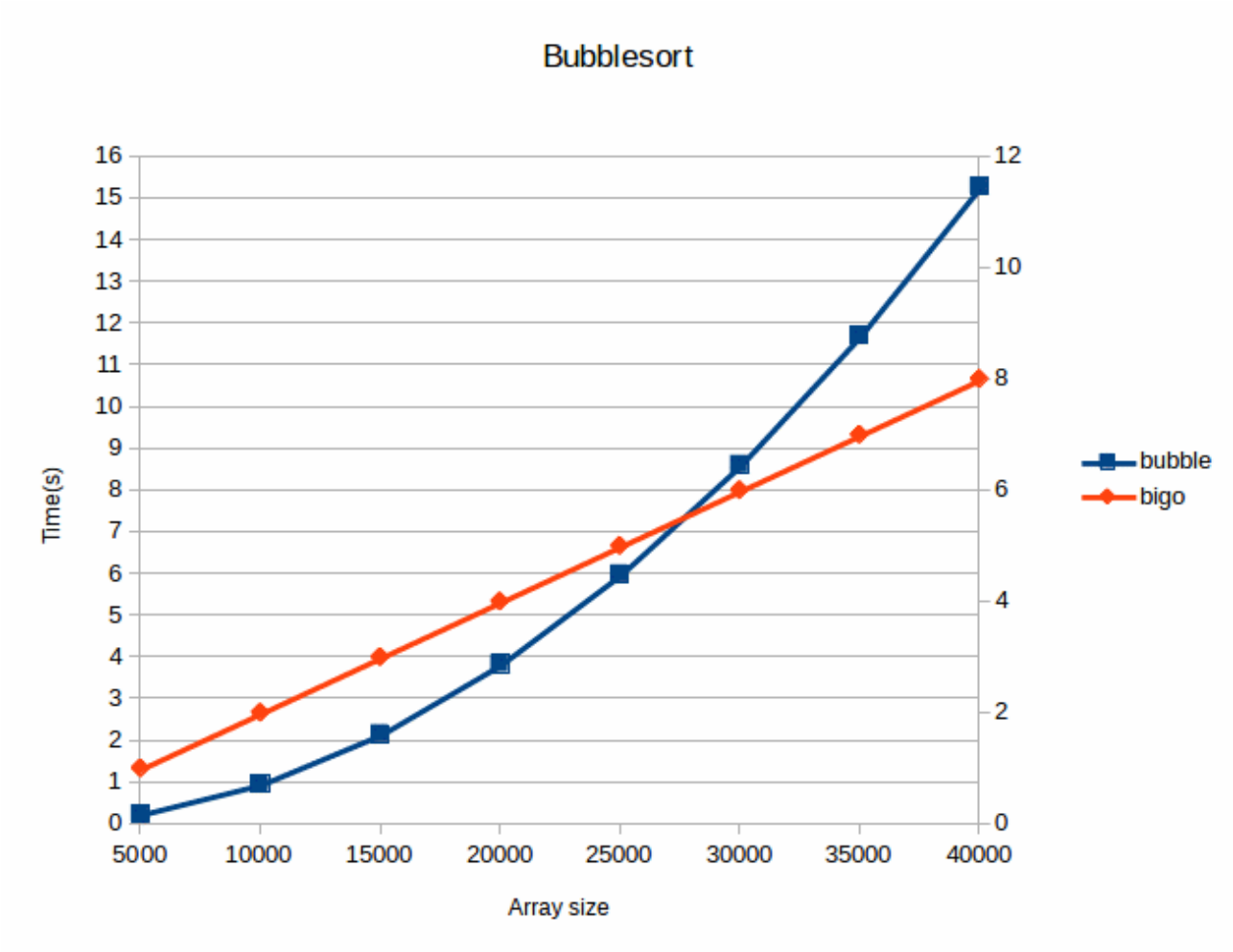
Slutsatser och kommentarer

Började glatt koda men insåg några timmar in att jag gjort en designmiss så programmet kastades i Plus-Sverkers soptunna och jag satte mig och planerade lite med papper och penna. Sen gick det relativt snabbt att få till ett fungerande program. Jag är dock fortfarande något osäker på om mina överlagrade operatörer verkligen är korrekta, även om programmet fungerar och gör vad det ska.

Bubblesort

Här kan man konstatera att mina praktiska värden skiljer sig en hel del från de teoretiska värdena och att jag nog mest troligt använt en långsam variant av algoritmen. I det stora hela är bubblesort oerhört ineffektiv jämfört med de andra undersökta algoritmerna. Åtminstone på det datamängder som är testkörda i den här laborationen.

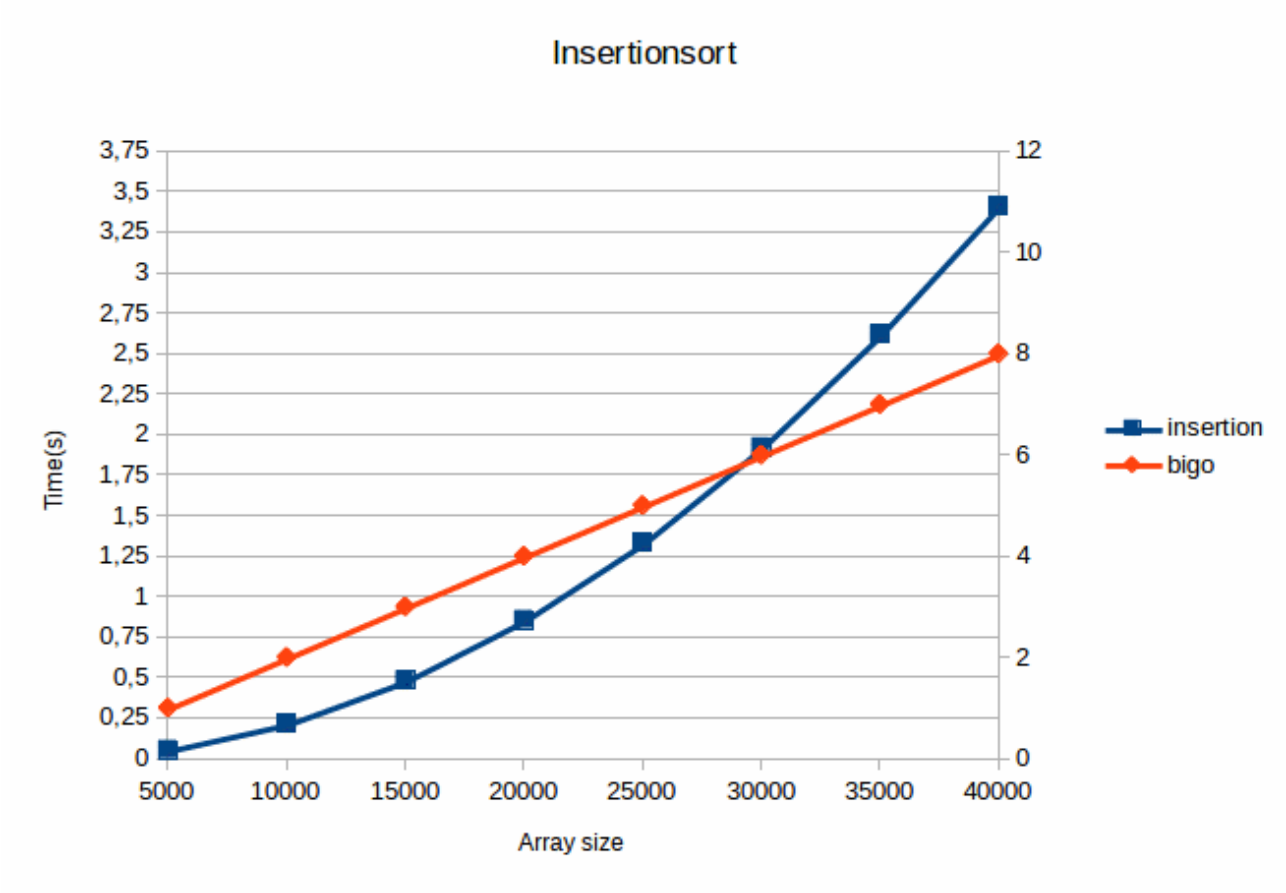
Algoritm	Arraystorlek	Tid i sek	O(n^2)
Bubblesort	5000	0,238229	25000000
Bubblesort	10000	0,956803	100000000
Bubblesort	15000	2,154283	225000000
Bubblesort	20000	3,831269	400000000
Bubblesort	25000	5,985887	625000000
Bubblesort	30000	8,611687	900000000
Bubblesort	35000	11,710654	1225000000
Bubblesort	40000	15,287497	1600000000



Insertionsort

När det gäller den här algoritmen stämmer praktik och teori överens något bättre än i föregående test. Dock är det fortfarande en relativt stor avvikelse som eventuellt kan härledas till slumpen eller kanske plattformsspecifika egenheter.

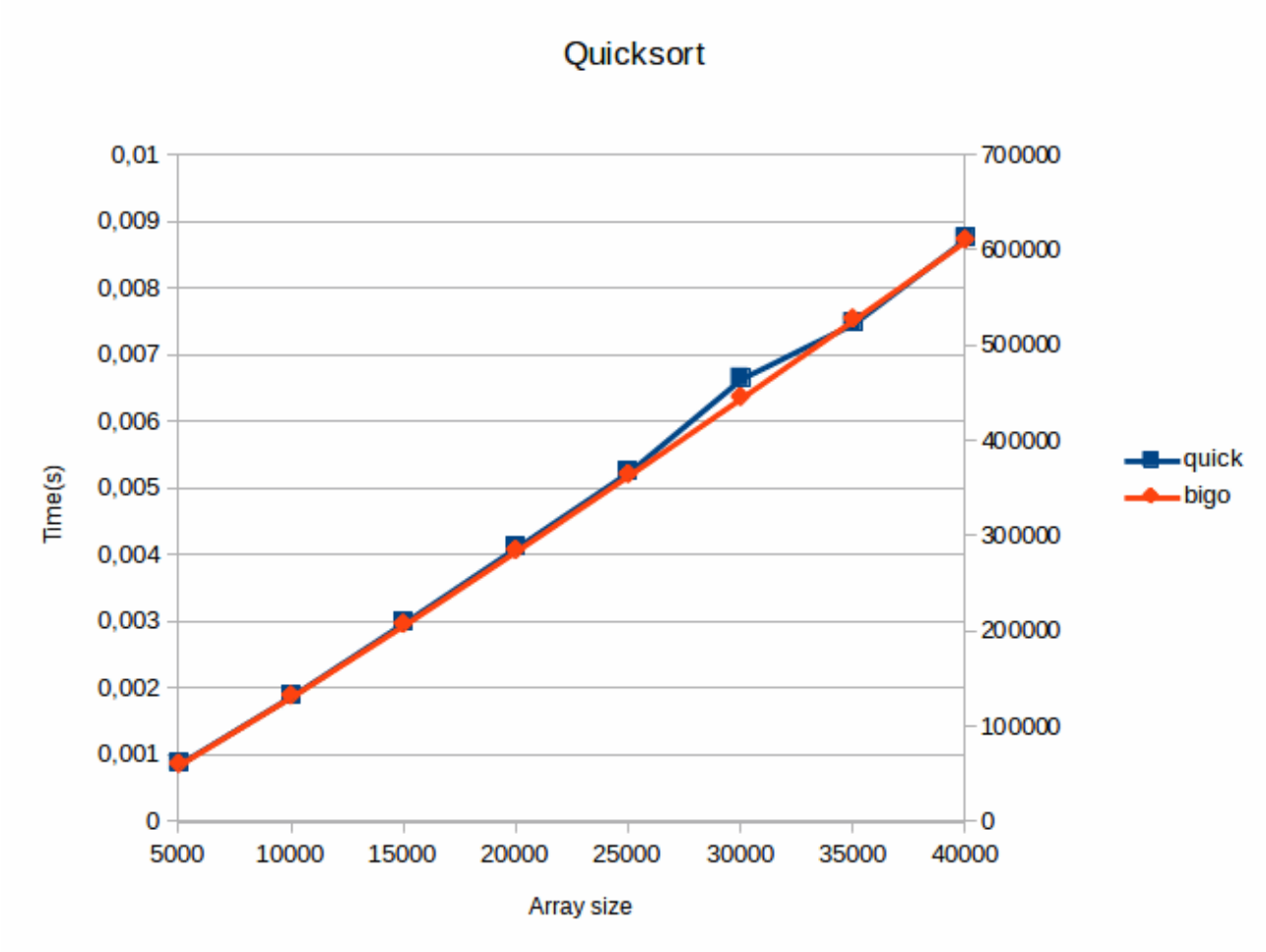
Algoritm	Arraystorlek	Tid i sek	O(n^2)
Insertsort	5000	0,052444	25000000
Insertsort	10000	0,214397	100000000
Insertsort	15000	0,481246	225000000
Insertsort	20000	0,855243	400000000
Insertsort	25000	1,331451	625000000
Insertsort	30000	1,921778	900000000
Insertsort	35000	2,619682	1225000000
Insertsort	40000	3,414045	1600000000



Quicksort

Här kan vi konstatera att quicksort gjorde processen kort med de övriga algoritmerna när det gäller prestanda. Praktik och teori stämmer också mycket bra överens med varandra.

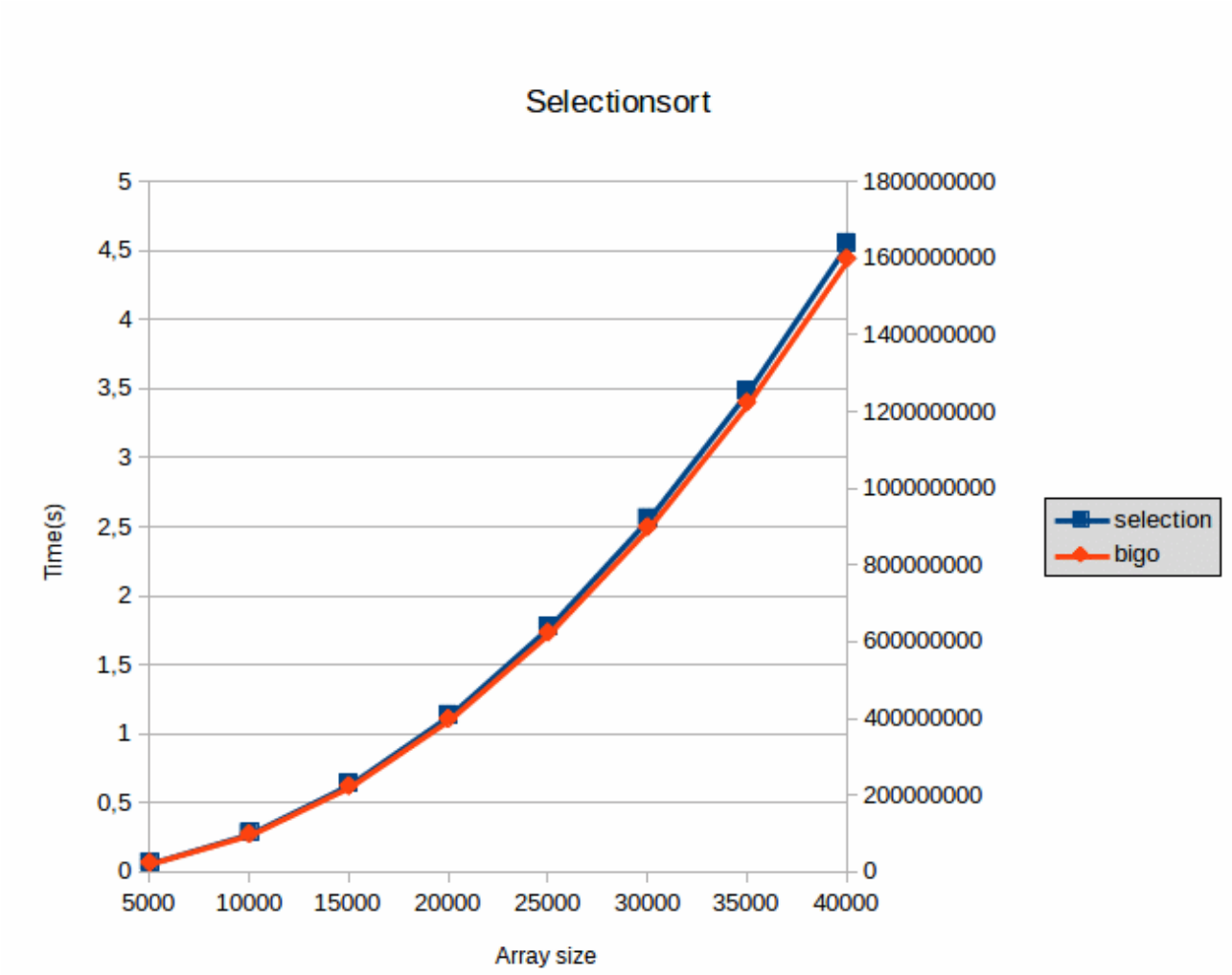
Algoritm	Arraystorlek	Tid i sek	O(n log(2) n)
Quicksort	5000	0,000888	61438,56
Quicksort	10000	0,001906	132877,12
Quicksort	15000	0,003006	208090,12
Quicksort	20000	0,004132	285754,25
Quicksort	25000	0,005267	365241,01
Quicksort	30000	0,006664	446180,25
Quicksort	35000	0,007493	528327,36
Quicksort	40000	0,008762	611508,50



Selectionsort

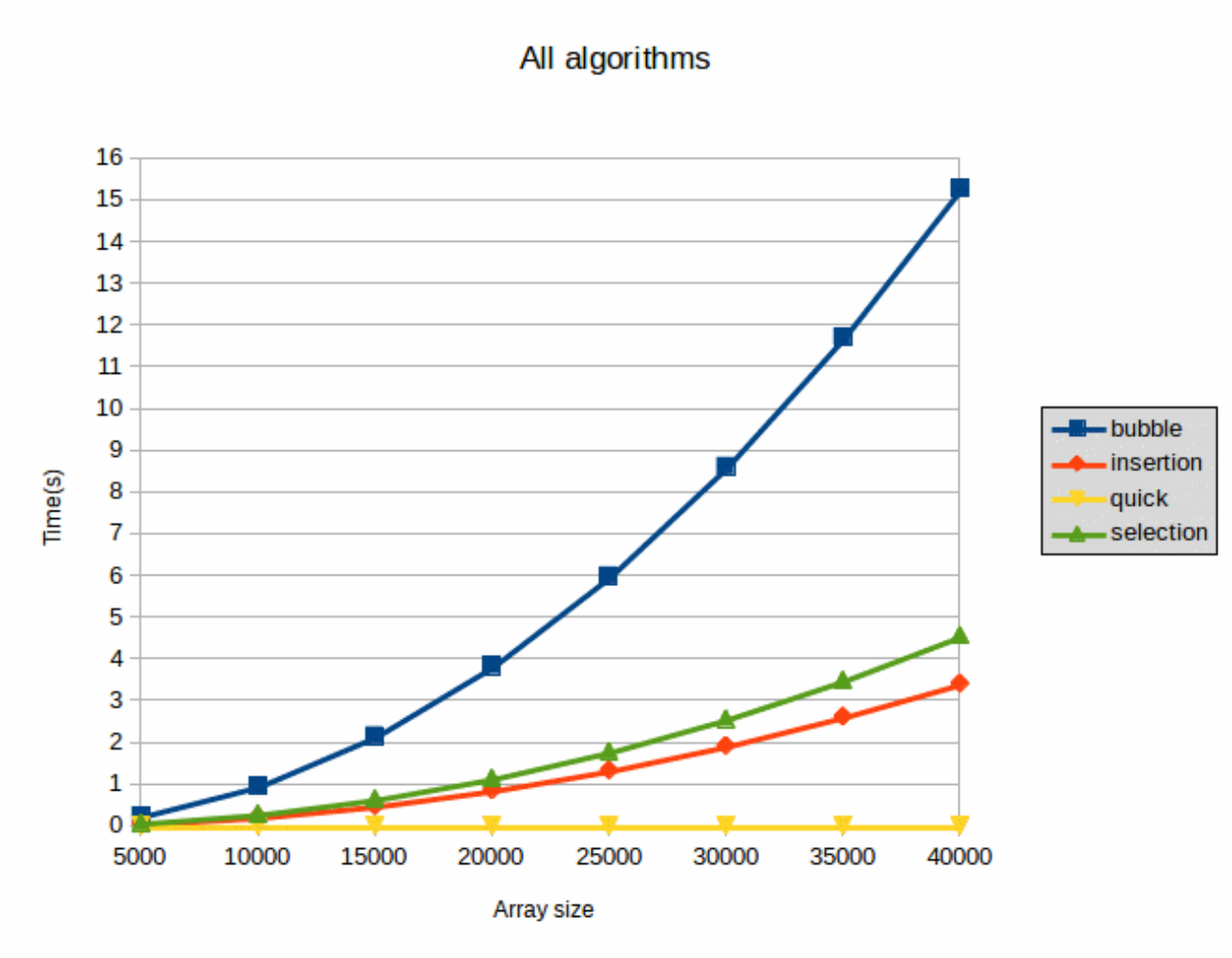
Selectionsort kvalar in för bronsmedaljen hack i häl på insertionsort och varvar bubblesort flera gånger om. Dock har den ingen chans alls mot quicksort. Teori och praktik stämmer mycket väl överens för denna algoritm enligt mina testresultat.

Algoritm	Arraystorlek	Tid i sek	O(n^2)
Selectsort	5000	0,071652	25000000
Selectsort	10000	0,285695	100000000
Selectsort	15000	0,642253	225000000
Selectsort	20000	1,140869	400000000
Selectsort	25000	1,782016	625000000
Selectsort	30000	2,565305	900000000
Selectsort	35000	3,490443	1225000000
Selectsort	40000	4,558584	1600000000



Samtliga algoritmer

Det står utom alla rimliga tvivel klart att quicksort är vida överlägsen de andra algoritmerna om man ser till ren prestanda. Quicksort är så snabb att det är svårt att rita en graf på samma skala som de övriga algoritmerna och se några förändringar alls. Detta är troligtvis en anledning till varför just denna algoritm varit standard i många UNIX-system och programmeringsbibliotek (t.ex. qsort()) i flera decenium. Insertionsort och selectionsort presterar relativt likvärdiga resultat medan bubblesort är hopplöst långsammast.


All algorithms
Time(s)
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
5000
10000
15000
20000
25000
30000
35000
40000
Array size
bubble
insertion
quick
selection