Rapport
Laboration 2

Datastrukturer och Algorimeter

Mathias Andersson
Henrik Bäck

Datastrukturer och Algoritmer

Laboration 2

Innehållsförteckning

Bubbel sort

Förväntningar och antaganden

Värsta fallet för en bubbel sort borde vara när element är sorterade i omvänd ordning. Tiden för en körning av en sådan sortering torde vara av storleken $O(n^2)$ därför att alla element, utom det mittersta (om det finns ett sådant), behöver flyttas. Både den inre och yttre iterationen i algoritmen måste göras n gånger.

Bästa fallet for en bubbel sort borde vara när element är sorterade i ordning. Tiden för en körning av en sådan sortering borde vara O(n) eftersom en sökning av alla element behöver göras för att konstatera att de ligger i ordning.

Vanliga fallet för en bubbel sort borde vara när element är slumpvis sorterade. Tiden för en körning av en sådan sortering borde vara $O(n^2)$. Dock borde det finnas en större chans, i det långa loppet, att elementen ligger i en mer strukturerad ordning och alla element behöver inte flyttas alla steg. Alltså borde det vanliga fallet vara något bättre än det värsta fallet trots att den har samma O.

Tillvägagångssätt

Värsta fallet

Innan sortering och tidtagning startar fylls en mängd med numeriska värden. Värdena är sorterade baklänges den ordning som, i sorteringsfuntkionen, skall sorteras. Därefter startar tidtagningen på sorteringen för denna lista. Detta upprepas ett antal gånger för att få ett antal mätvärden.

Bästa fallet

Genom att köra sorteringsalgoritmen på en mängd som redan är sorterad och klar erhåller man ett bästa fall. Detta för att algoritmen kommer att avslutas då inga fler byten behöver göras. Vilket kommer att upptäckas efter första iterationen. Därefter startar tidtagningen på sorteringen för denna lista. Detta upprepas ett antal gånger för att få ett antal mätvärden.

Medelfallet

Genom att förse sorteringsfunktionen med mängd som är slumpvis sorterad kan man försöka erhålla ett så allmänt fall som möjligt. Elementen kommer att ligga på olika ställen och antalet förflyttningar kommer att variera från fall till fall. Därefter startar tidtagningen på sorteringen för denna lista. Detta upprepas ett antal gånger för att få ett antal mätvärden.

Algoritmen

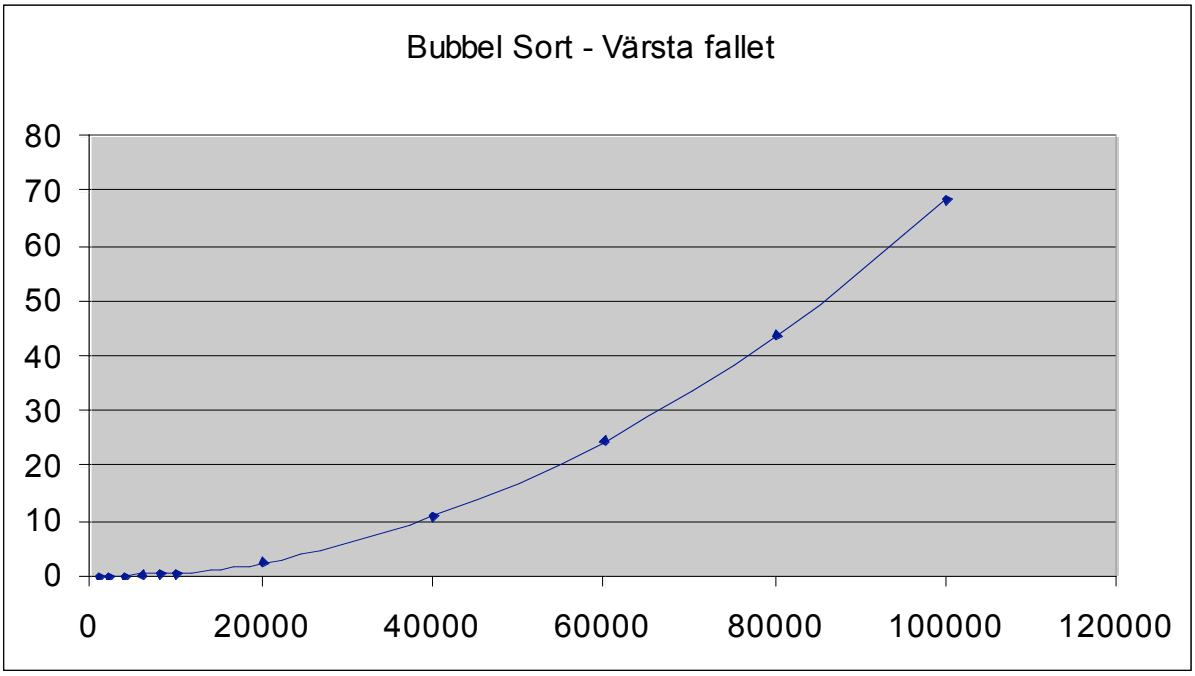
Bubbel sortalgoritmen använder en flagga som sätts ifall den senaste iterationen inte bytte några element. Detta avbryter sorteringsfunktionen eftersom den i så fall är klar.

Mätvärden

Tester för 1000, 2000, 4000, 6000, 8000, 10000, 20000, 40000, 60000, 80000 och 100000 element utfördes med tio (10) tester för varje storlek av element. Därefter räknades, för varje elementmängd, ett medelvärde ut.

Värsta fallet

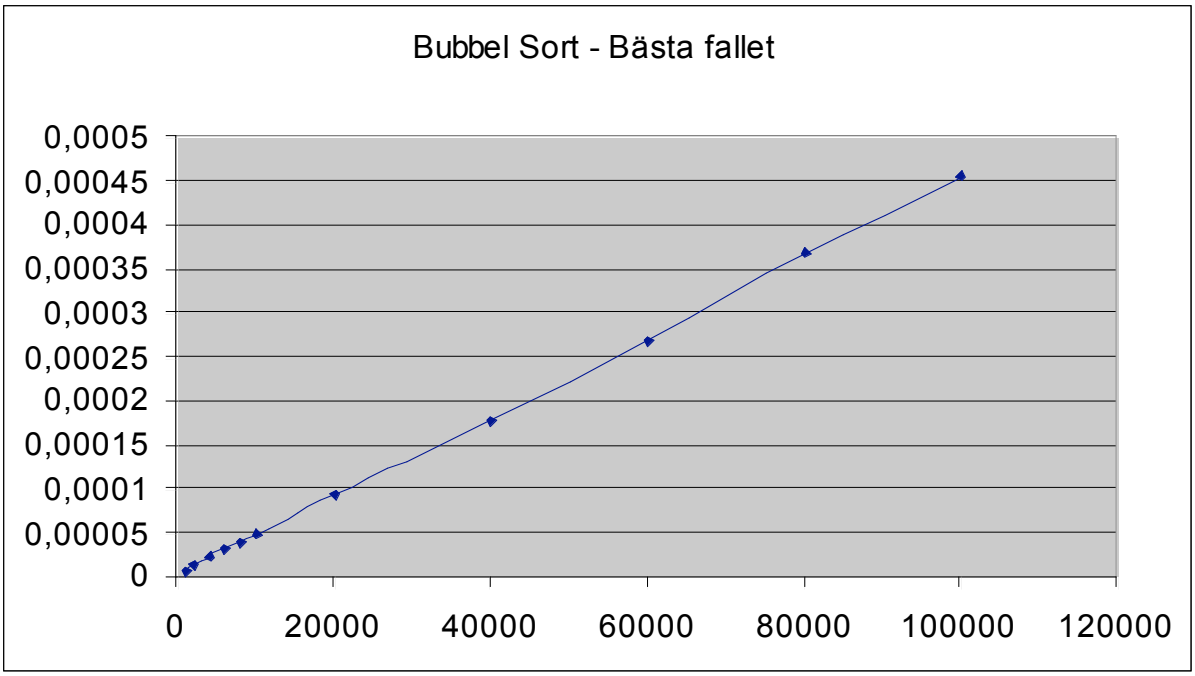
Antal element	Medelkörningstid (s)
1000	0,006791
2000	0,027566
4000	0,109217
6000	0,246685
8000	0,438812
10000	0,684724
20000	2,731864
40000	10,9622
60000	24,6818
80000	44,005
100000	68,7818



Bästa fallet

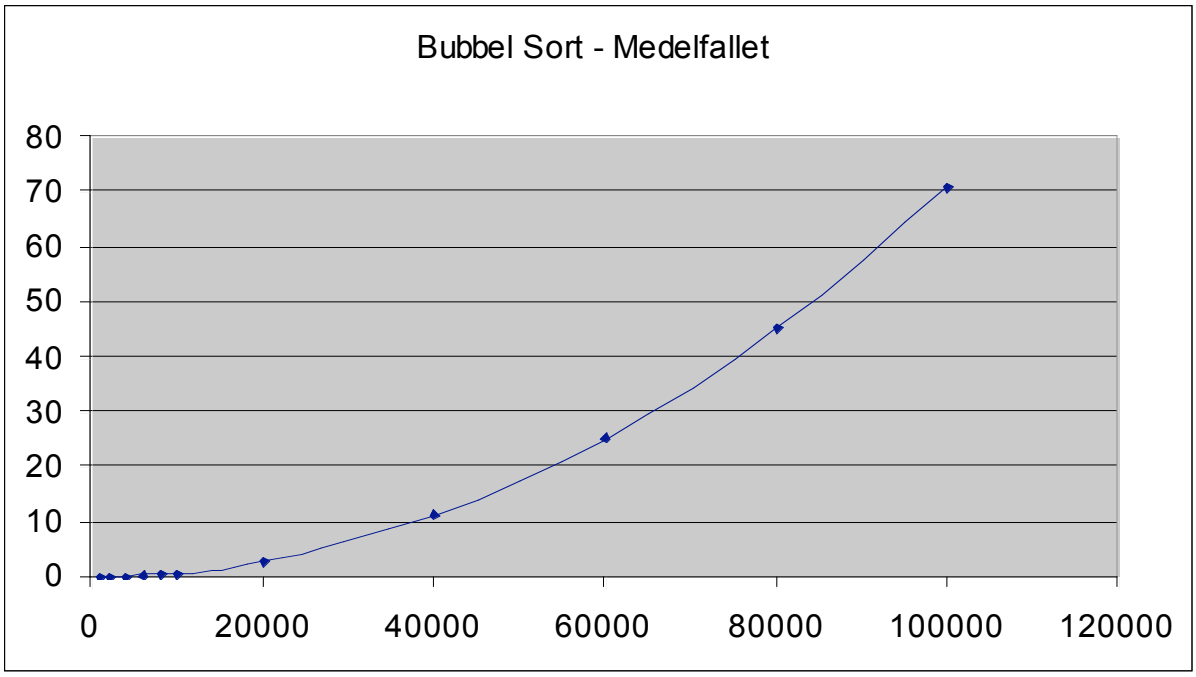
Antal element	Medelkörningstid (s)
1000	6,489E-06
2000	1,4E-05
4000	2,325E-05
6000	3,188E-05
8000	4,0793E-05
10000	4,909E-05
20000	9,38892E-05
40000	0,000179
60000	0,00027
80000	0,00037

100000	0,00046



Medelfallet

Antal element	Medelkörningstid (s)
1000	0,006828
2000	0,029351
4000	0,11258
6000	0,254155
8000	0,452163
10000	0,7114
20000	2,82541
40000	11,3179
60000	25,4452
80000	45,313
100000	70,9762



Analys

Värsta fallet

Antal element	Medelkörningstid (s)	T/n	T/n^2	T/n^3
1000	0,006791	6,791E-06	6,79E-09	6,79E-12
2000	0,027566	1,3783E-05	6,89E-09	3,45E-12
4000	0,109217	2,7304E-05	6,83E-09	1,71E-12
6000	0,246685	4,1114E-05	6,85E-09	1,14E-12
8000	0,438812	5,4852E-05	6,86E-09	8,57E-13
10000	0,684724	6,8472E-05	6,85E-09	6,85E-13
20000	2,731864	0,00013659	6,83E-09	3,41E-13
40000	10,9622	0,00027406	6,85E-09	1,71E-13
60000	24,6818	0,00041136	6,86E-09	1,14E-13
80000	44,005	0,00055006	6,88E-09	8,59E-14
100000	68,7818	0,00068782	6,88E-09	6,88E-14

Här kan ses att T/n^2 konvergerar. Detta tyder på att det värsta fallet för bubbel sort är av storleksordningen n^2. Detta ger även grafen över medelkörningstiden visst stöd för.

Bästa fallet

Antal element	Medelkörningstid (s) [T]	T/n	T/n^2	T/n^3
1000	6,489E-06	6,49E-09	6,49E-12	6,49E-15
2000	1,4E-05	7,00E-09	3,50E-12	1,75E-15
4000	2,325E-05	5,81E-09	1,45E-12	3,63E-16
6000	3,188E-05	5,31E-09	8,86E-13	1,48E-16
8000	4,0793E-05	5,10E-09	6,37E-13	7,97E-17
10000	4,909E-05	4,91E-09	4,91E-13	4,91E-17

20000	9,38892E-05	4,69E-09	2,35E-13	1,17E-17
40000	0,000179	4,48E-09	1,12E-13	2,80E-18
60000	0,00027	4,50E-09	7,50E-14	1,25E-18
80000	0,00037	4,63E-09	5,78E-14	7,23E-19
100000	0,00046	4,60E-09	4,60E-14	4,60E-19

Här kan ses att T/n konvergerar. Detta tyder på att det bästa fallet för bubbel sort är av storleksordningen n. Grafen för över medelkörningstiden ger stöd för denna analys.

Medelfallet

Antal element	**Medelkörningstid (s) [T]**	**T/n**	**T/n^2**	**T/n^3**
1000	0,006828	6,83E-06	6,83E-09	6,83E-12
2000	0,029351	1,47E-05	7,34E-09	3,67E-12
4000	0,11258	2,81E-05	7,04E-09	1,76E-12
6000	0,254155	4,24E-05	7,06E-09	1,18E-12
8000	0,452163	5,65E-05	7,07E-09	8,83E-13
10000	0,7114	7,11E-05	7,11E-09	7,11E-13
20000	2,82541	1,41E-04	7,06E-09	3,53E-13
40000	11,3179	2,83E-04	7,07E-09	1,77E-13
60000	25,4452	4,24E-04	7,07E-09	1,18E-13
80000	45,313	5,66E-04	7,08E-09	8,85E-14
100000	70,9762	7,10E-04	7,10E-09	7,10E-14

Här kan ses att T/n^2 konvergerar. Detta tyder på att medelfallet för bubbel sort är av storleksordningen n^2. Detta ger även grafen över medelkörningstiden visst stöd för.

Resultat

Värsta fallet

Enligt antagandet skall det värsta fallet för en bubbel sort vara när listan är sorterad i omvänd ordning. För att sortera listan kräver det då att vi gör n^2 iterationer. Analysen, se föregående rubrik, finner att det värsta fallet för bubbel sort är av ordning $\mathbf{O(n^2)}$.

Bästa fallet

Enligt antagandet skall det bästa fallet för en bubbel sort vara när listan är sorterad korrekt. För att säkerställa att listan är sorterad krävs att vi åtminstone gör n iterationer. Analysen, se föregående rubrik, finner att det bästa fallet för bubbel sort är av ordning $\mathbf{O(n)}$.

Medelfallet

Enligt antagandet för medelfallet gäller att listan skall vara sorterad i en slumpvis ordning. På så sätt varierar antalet iterationer för att göra listan sorderad. Analysen, se föregående rubrik, finner att medelfallet för bubbel sort är av ordning $\mathbf{O(n^2)}$.

Diskussion

Enligt resultaten för denna mätning visar det sig att det antagandet som ställts upp om att det vanliga fallet borde ta något kortare tid än det värsta fallet inte stämmer. Fortfarande görs färre

iterationer i det vanliga fallet än i det värsta fallet. Trots detta visar testet att det tar längre att sortera det vanliga fallet än det värsta. Att algoritmen eller systemet beter sig på detta sätt har försökt att utrönas genom att köra programvaran på ett annat, likvärdigt, system. Det har också försökts med att använda olika sorters funktioner för att få programmet att göra ett uppehåll innan tidtagning och sortering startar. Dessa åtgärder har inte givigt något underlag till varför det allmänna fallet tar längre tid än det värsta fallet. Det har även påvisats att det allmänna fallet har färre iterationer än det värsta fallet. När denna rapport färdigställts finns ännu ingen kunskap om varför sorteringen beter sig på detta sätt i detta fall.

Insertion sort

Värsta fallet för en insertion sort borde vara när data är sorterade i bakvänd ordning. Tiden för en körning av en sådan sortering torde vara av storleken $O(n^2)$ därför att alla element, utom det mittersta (om ett sådant finns), behöver flyttas.

Bästa fallet for en insertion sort borde vara när data är sorterade i ordning. Tiden för en körning av en sådan sortering borde vara O(n) eftersom en sökning av alla element behöver göras för att konstatera att de ligger i ordning.

Vanliga fallet för en insertion sort borde vara när data är slumpvis sorterade. Tiden för en körning av en sådan sortering borde vara $O(n^2)$. Dock borde det finnas en större chans, i det långa loppet, att elementen ligger i en mer strukturerad ordning och alla element behöver inte flyttas alla steg.

Tillvägagångssätt

Värsta fallet

Innan sortering och tidtagning startar, fylls en lista med numeriska värden. Värdena är sorterade baklänges den ordning som, i sorteringsfuntkionen, skall sorteras. Därefter startar tidtagningen på sorteringen för denna lista. Detta upprepas ett antal gånger för att få ett antal mätvärden.

Bästa fallet

Genom att köra sorteringsalgoritmen på en lista som redan är sorterad erhålles ett bästa fall. Listan fylls med detta och därefter startar tidtagningen på sorteringen för denna lista. Detta upprepas ett antal gånger för att få ett antal mätvärden.

Medelfallet

Genom att förse sorteringsfunktionen med lista som är slumpvis sorterad kan man försöka erhålla ett så allmänt fall som möjligt. Elementen kommer att ligga på olika ställen och antalet förflyttningar kommer att variera från fall till fall. Därefter startar tidtagningen på sorteringen för denna lista. Detta upprepas ett antal gånger för att få ett antal mätvärden.

Algoritmen

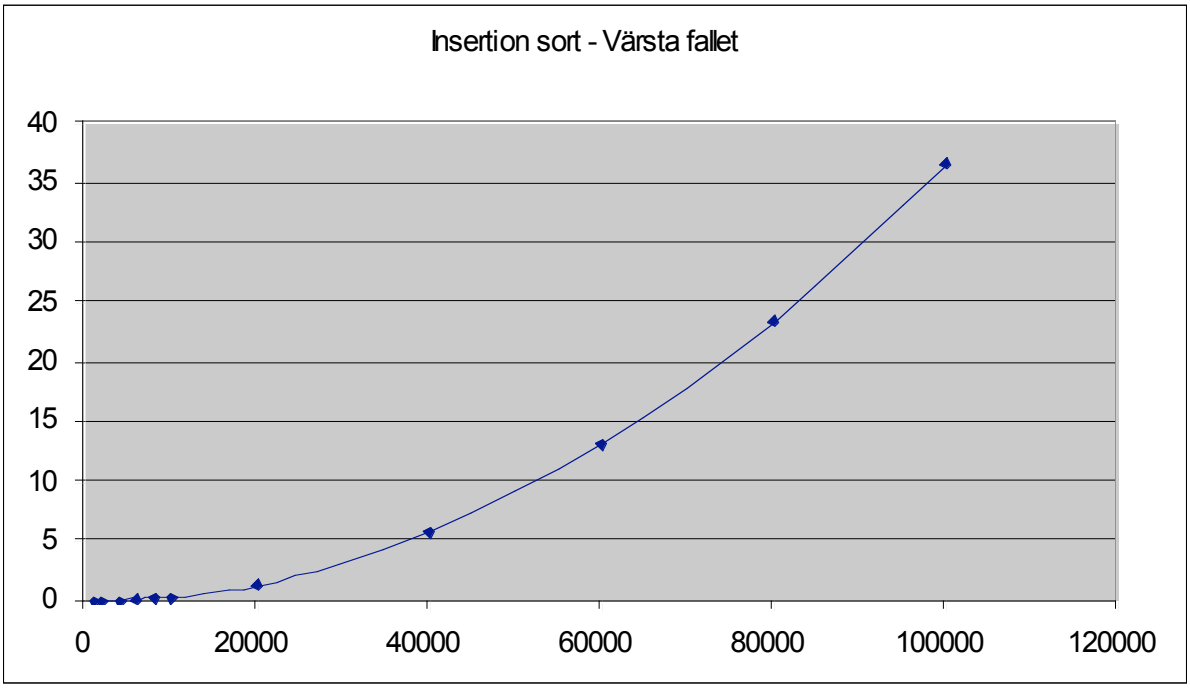
Den insertion sort som används är implementerad med endast en lista. Det betyder att den sorterar i den befintliga listan istället för att flytta till en ny.

Mätvärden

Tester för 1000, 2000, 4000, 6000, 8000, 10000, 20000, 40000, 60000, 80000 och 100000 element utfördes med tio (10) tester för varje storlek av element. Därefter räknades, för varje elementmängd, ett medelvärde ut.

Värsta fallet

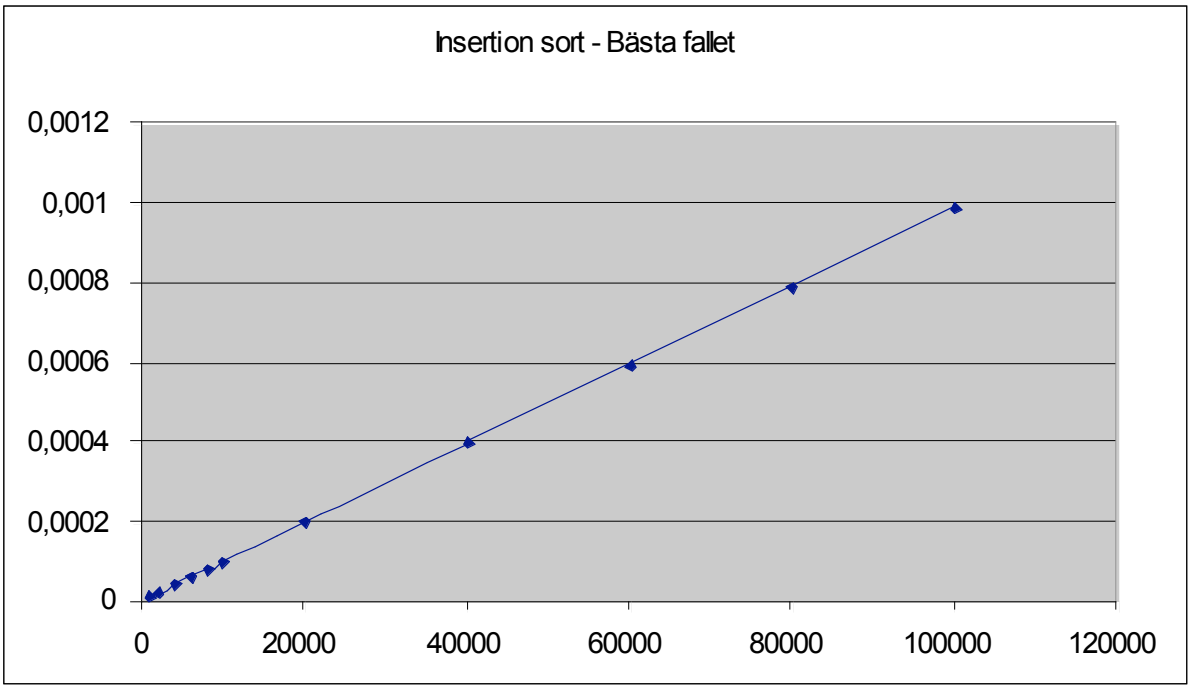
Antal element	Medelkörningstid (s)
1000	0,00365124
2000	0,0145435
4000	0,0585934
6000	0,130913
8000	0,232643
10000	0,364052
20000	1,45411
40000	5,82731
60000	13,1185
80000	23,3737
100000	36,6681



Bästa fallet

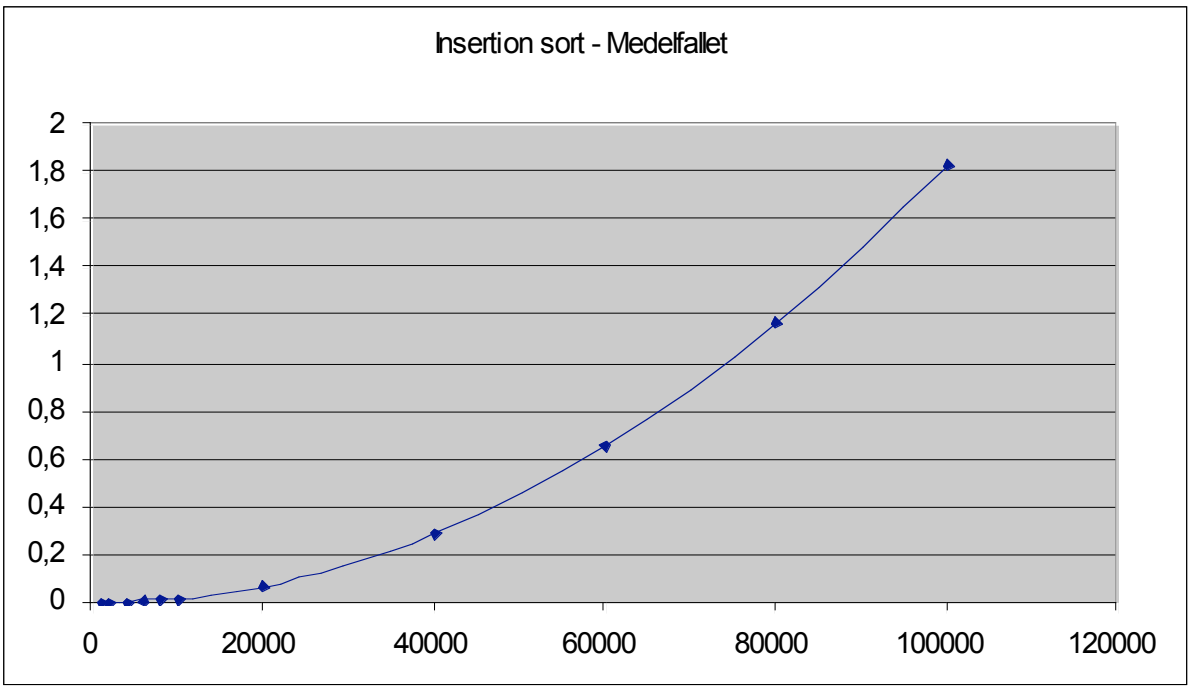
Antal element	Medelkörningstid (s)
1000	1,52349E-05
2000	2,503E-05
4000	4,4537E-05
6000	6,4111E-05
8000	8,37326E-05
10000	0,00010322
20000	0,000201702

40000	0,000398
60000	0,000593
80000	0,000792
100000	0,000989



Medelfallet

Antal element	Medelkörningstid (s)
1000	0,0001917
2000	0,000762
4000	0,0029882
6000	0,0064973
8000	0,0115984
10000	0,0182535
20000	0,0725145
40000	0,291509
60000	0,65787
80000	1,1716
100000	1,83337



Analys

Värsta fallet

Antal element	Medelkörningstid (s)	T/n	T/n^2	T/n^3
1000	0,00365124	3,65124E-06	3,65124E-09	3,65124E-12
2000	0,0145435	7,27175E-06	3,63588E-09	1,81794E-12
4000	0,0585934	1,46484E-05	3,66209E-09	9,15522E-13
6000	0,130913	2,18188E-05	3,63647E-09	6,06079E-13
8000	0,232643	2,90804E-05	3,63505E-09	4,54381E-13
10000	0,364052	3,64052E-05	3,64052E-09	3,64052E-13
20000	1,45411	7,27055E-05	3,63528E-09	1,81764E-13
40000	5,82731	0,000145683	3,64207E-09	9,10517E-14
60000	13,1185	0,000218642	3,64403E-09	6,07338E-14
80000	23,3737	0,000292171	3,65214E-09	4,56518E-14
100000	36,6681	0,000366681	3,66681E-09	3,66681E-14

Här kan ses att T/n^2 konvergerar. Detta tyder på att det värsta fallet för insertion sort är av storleksordningen n^2. Detta ger även grafen över medelkörningstiden stöd för.

Bästa fallet

Antal element	Medelkörningstid (s) [T]	T/n	T/n^2	T/n^3
1000	1,52349E-05	1,52349E-08	1,52349E-11	1,52349E-14
2000	2,503E-05	1,2515E-08	6,2575E-12	3,12875E-15
4000	4,4537E-05	1,11343E-08	2,78356E-12	6,95891E-16
6000	6,4111E-05	1,06852E-08	1,78086E-12	2,9681E-16
8000	8,37326E-05	1,04666E-08	1,30832E-12	1,6354E-16
10000	0,00010322	1,0322E-08	1,0322E-12	1,0322E-16

20000	0,000201702	1,00851E-08	5,04255E-13	2,52128E-17
40000	0,000398	9,95E-09	2,4875E-13	6,21875E-18
60000	0,000593	9,88333E-09	1,64722E-13	2,74537E-18
80000	0,000792	9,9E-09	1,2375E-13	1,54688E-18
100000	0,000989	9,89E-09	9,89E-14	9,89E-19

Här kan ses att T/n konvergerar. Detta tyder på att det bästa fallet för insertion sort är av storleksordningen n. Grafen för över medelkörningstiden ger stöd för denna analys.

Medelfallet

Antal element	**Medelkörningstid (s) [T]**	**T/n**	**T/n^2**	**T/n^3**
1000	0,0001917	1,917E-07	1,917E-10	1,917E-13
2000	0,000762	0,000000381	1,905E-10	9,525E-14
4000	0,0029882	7,4705E-07	1,86763E-10	4,66906E-14
6000	0,0064973	1,08288E-06	1,80481E-10	3,00801E-14
8000	0,0115984	1,4498E-06	1,81225E-10	2,26531E-14
10000	0,0182535	1,82535E-06	1,82535E-10	1,82535E-14
20000	0,0725145	3,62573E-06	1,81286E-10	9,06431E-15
40000	0,291509	7,28773E-06	1,82193E-10	4,55483E-15
60000	0,65787	1,09645E-05	1,82742E-10	3,04569E-15
80000	1,1716	0,000014645	1,83063E-10	2,28828E-15
100000	1,83337	1,83337E-05	1,83337E-10	1,83337E-15

Här kan ses att T/n^2 konvergerar. Detta tyder på att medelfallet för insertion sort är av storleksordningen n^2. Detta ger även grafen över medelkörningstiden stöd för.

Resultat

Värsta fallet

Enligt antagandet skall det värsta fallet för en insertion sort vara när listan är sorterad i omvänd ordning. Analysen, se föregående rubrik, finner att det värsta fallet för insertion sort är av ordning $\mathbf{O(n^2)}$.

Bästa fallet

Enligt antagandet skall det bästa fallet för en insertion sort vara när listan är sorterad korrekt. Analysen, se föregående rubrik, finner att det bästa fallet för insertion sort är av ordning **O(n)**.

Medelfallet

Enligt antagandet för medelfallet gäller att listan skall vara sorterad i en slumpvis ordning. Analysen, se föregående rubrik, finner att medelfallet för insertion sort är av ordning $\mathbf{O(n^2)}$.

Diskussion

En intressant anmärkning är skillnaden mellan det vanliga fallet och det värsta fallet som är väldigt stor trots att båda är av samma O.

Quick Sort

Det värsta fallet för Quick Sort kommer att vara när den lista som används redan är sorterad samt att den pivot som väljs det första elementet. Detta ger ett maximalt antal partitioner. I detta fall är körningstiden av storleksordning $O(n^2)$.

Det bästa fallet för Quick Sort kommer att vara när den lista som används är sorterad samt att den pivot som väljs befinner sig i mitten av listan. I detta fall är körningstiden av storleksordning O(n log n)) ty djupet av rekursionen är log n och vi har utöver det n st element att behandla.

Medelfallet för Quick Sort kommer att vara när den lista som används är slumpvist ordnad samt att den pivot som väljs befinner sig i mitten av listan. I detta fall är körningstiden av storleksordning O(n log n) ty djupet av rekursionen är log n och vi har utöver det n st element att behandla.

Tillvägagångssätt

Värsta fallet

En algoritm som väljer första elementet som pivot används. Algoritmen förses med en redan sorterad lista. Därefter startar tidtagningen på sorteringen för denna lista. Detta upprepas ett antal gånger för att få ett antal mätvärden.

Bästa fallet

En algoritm som väljer mittersta elementet som pivot används. Algoritmen förses med en redan sorterad lista. Därefter startar tidtagningen på sorteringen för denna lista. Detta upprepas ett antal gånger för att få ett antal mätvärden.

Medelfallet

En algoritm som väljer mittersta elementet som pivot används. Algoritmen förses med en redan slumpvis ordnad lista. Därefter startar tidtagningen på sorteringen för denna lista. Detta upprepas ett antal gånger för att få ett antal mätvärden.

Algoritmen

För värsta fallet används en partitionering där första elementet väljs som pivot. Detta för att åstadkomma så många partitioner som möjligt och på det sättet kunna erhålla ett värsta fall.

För de andra fallen partitioneras listan med avsende på det mittersta elementet som pivot. Detta för att kunna åstadkomma ett bästa fall. Dock används även denna för medelfallet.

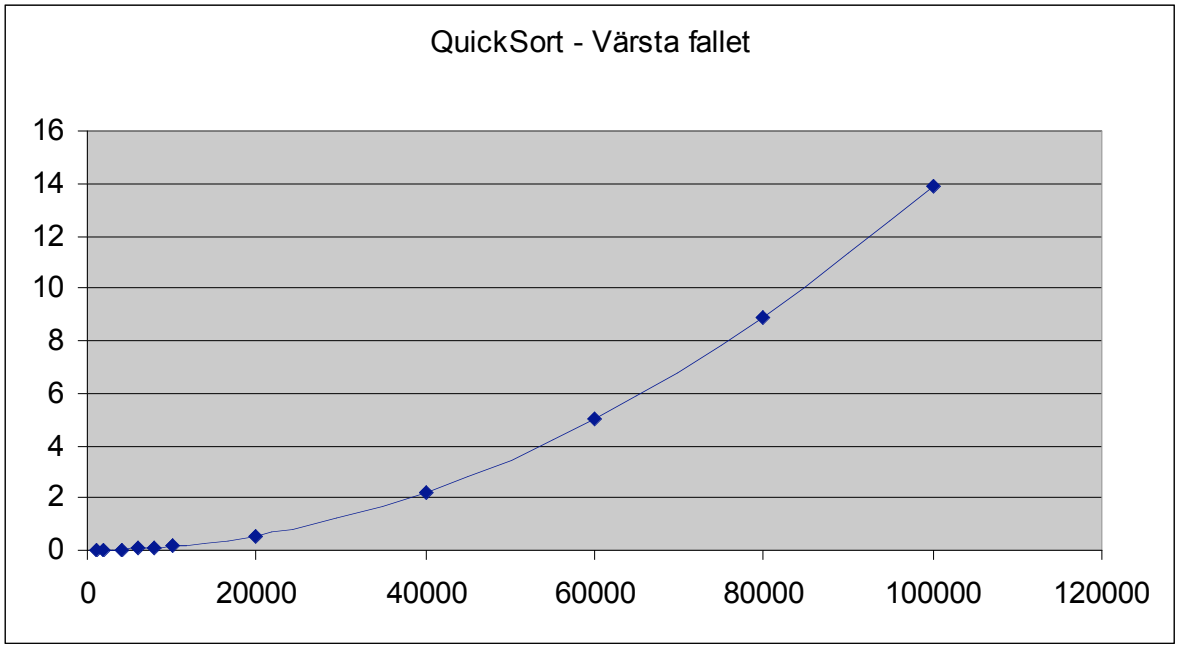
Mätvärden

Tester för 1000, 2000, 4000, 6000, 8000, 10000, 20000, 40000, 60000, 80000, 100000, 200000, 400000, 600000, 800000 och 1000000 element utfördes med tio (10) tester för varje storlek av element. Därefter räknades, för varje elementmängd, ett medelvärde ut.

Värsta fallet

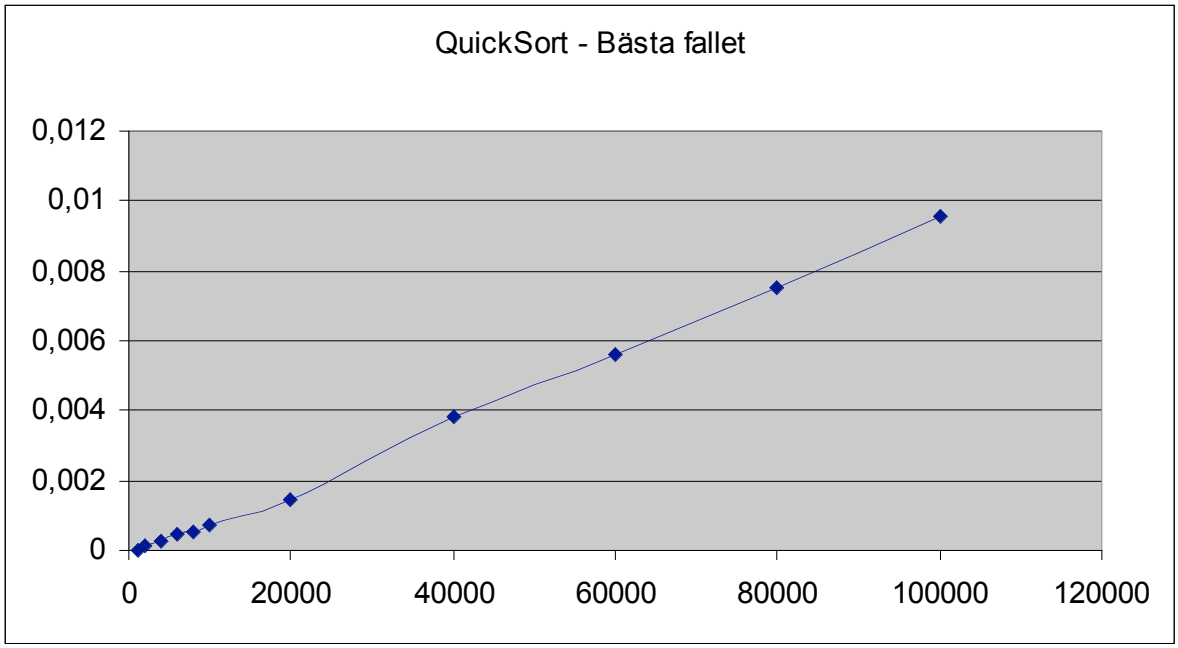
Antal element	Medelkörningstid (s)

1000	0,00139699
2000	0,0054874
4000	0,0221635
6000	0,0498062
8000	0,0885602
10000	0,139523
20000	0,554403
40000	2,21083
60000	4,97743
80000	8,86596
100000	13,9235



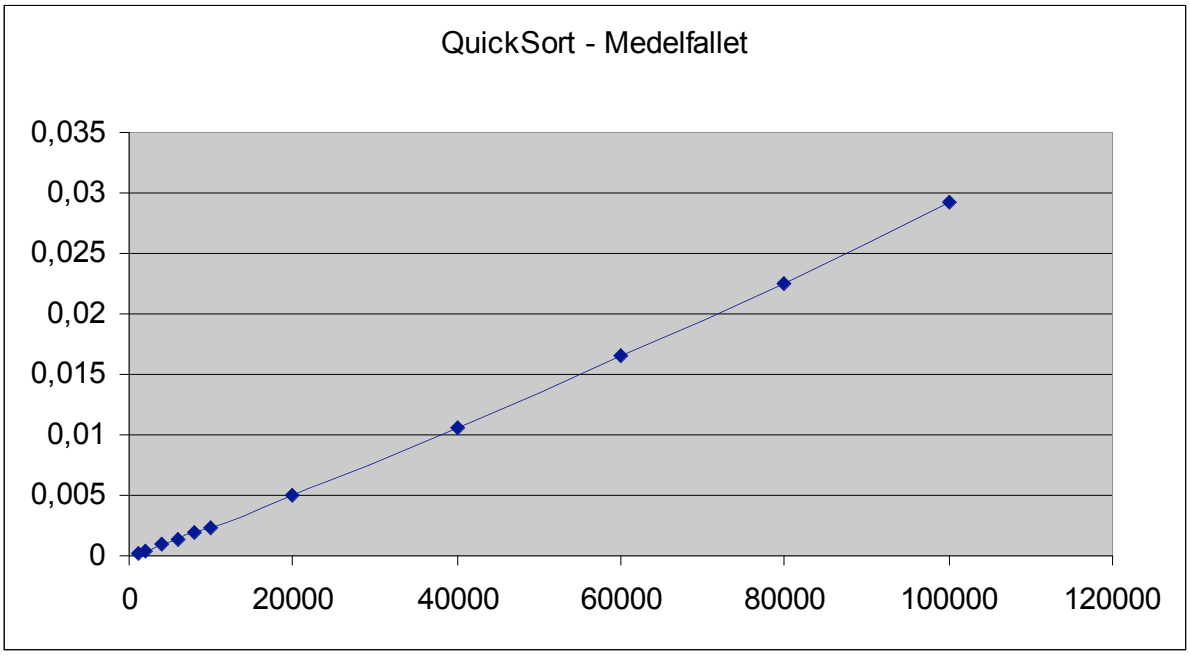
Bästa fallet

Antal element	Medelkörningstid (s)
1000	6,497E-06
2000	0,000132
4000	0,0002747
6000	0,00044
8000	0,00055459
10000	0,0006944
20000	0,00143712
40000	0,003809
60000	0,00563526
80000	0,0075031
100000	0,00954971



Medelfallet

Antal element	Medelkörningstid (s)
1000	0,0001903
2000	0,000412
4000	0,0008673
6000	0,0013461
8000	0,00184078
10000	0,0023464
20000	0,00495908
40000	0,010545
60000	0,0164679
80000	0,0224252
100000	0,0292068



Analys

Värsta fallet

Antal element	Medelkörningstid (s) [T]	T/n	T/n^2	T/n^3
1000	0,00139699	1,39699E-06	1,39699E-09	1,39699E-12
2000	0,0054874	2,7437E-06	1,37185E-09	6,85925E-13
4000	0,0221635	5,54088E-06	1,38522E-09	3,46305E-13
6000	0,0498062	8,30103E-06	1,38351E-09	2,30584E-13
8000	0,0885602	1,107E-05	1,38375E-09	1,72969E-13
10000	0,139523	1,39523E-05	1,39523E-09	1,39523E-13
20000	0,554403	2,77202E-05	1,38601E-09	6,93004E-14
40000	2,21083	5,52708E-05	1,38177E-09	3,45442E-14
60000	4,97743	8,29572E-05	1,38262E-09	2,30437E-14
80000	8,86596	0,000110825	1,38531E-09	1,73163E-14
100000	13,9235	0,000139235	1,39235E-09	1,39235E-14

Här kan ses att T/n^2 konvergerar. Detta tyder på att det värsta fallet för quick sort är av storleksordningen n^2. Detta ger även grafen över medelkörningstiden någorlunda stöd för.

Bästa fallet

Antal element	Medelkörningstid (s) [T]	T/n	T/n^2
1000	6,50E-06	6,51931E-10	6,51931E-07
2000	0,000132	6,01872E-09	1,20374E-05
4000	0,0002747	5,73928E-09	2,29571E-05
6000	0,00044	5,84295E-09	3,50577E-05
8000	0,00055459	5,34667E-09	4,27733E-05
10000	0,0006944	5,22588E-09	5,22588E-05

20000	0,00143712	5,02922E-09	0,000100584
40000	0,003809	6,22886E-09	0,000249154
60000	0,00563526	5,91715E-09	0,000355029
80000	0,0075031	5,75825E-09	0,00046066
100000	0,00954971	5,7495E-09	0,00057495

Här kan ses att T/n konvergerar. Detta tyder på att det bästa fallet för quick sort är av storleksordningen n.

Medelfallet

Antal element	Medelkörningstid (s) [T]	T/log n	T/n log n	T/n
1000	0,0001903	1,90953E-05	1,90953E-08	1,90E-07
2000	0,000412	3,75714E-05	1,87857E-08	2,06E-07
4000	0,0008673	7,24817E-05	1,81204E-08	2,17E-07
6000	0,0013461	0,000107253	1,78754E-08	2,24E-07
8000	0,00184078	0,000141972	1,77465E-08	2,30E-07
10000	0,0023464	0,000176584	1,76584E-08	2,35E-07
20000	0,00495908	0,000347087	1,73544E-08	2,48E-07
40000	0,010545	0,00068977	1,72442E-08	2,64E-07
60000	0,0164679	0,0010375	1,72917E-08	2,74E-07
80000	0,0224252	0,001376817	1,72102E-08	2,80E-07
100000	0,0292068	0,001758425	1,75842E-08	2,92E-07

Här kan ses att T/n log n konvergerar, men även T/n kandiderar. Detta tyder på att det medelfallet för quick sort är av storleksordningen n log n.

Resultat

Värsta fallet

Enligt antagandet skall det värsta fallet för quick sort vara när listan är sorterad och som pivot väljs det första elementet. Detta resulterar i att quick sort presterar enligt **$O(n^2)$**

Bästa fallet

Enligt antagandet skall det bästa fallet för quick sort vara när listan är sorterad och som pivot väljs det mittersta elementet. Detta resulterar i att quick sort presterar enligt **O(n log n)**

Medelfallet

Enligt antagandet skall det medelfallet för quick sort vara när listan är osorterad och som pivot väljs det mittersta elementet. Detta resulterar i att quick sort presterar enligt **O(n log n)**

Diskussion

Efter att ha provat att använda första elementet som pivot i medelfallet har ungefär samma resultat erhållits som när det mittersta elementet används som pivot. Därför har ett av fallen valts i denna analys.

Linear Search

De tre fall som tidigare setts i sorteringsalgoritmerna passar inte in under sökning. Att försöka analysera fram ett värsta och bästa fall för en sökning kan vara ett bekymmer. En tanke skulle kunna vara att det bästa fallet vore när elementet finnes direkt, detta kommer dock att ge ett meningslöst testfall.

Av detta att döma borde det därför enbart finnas ett medelfall för Linear Search. Hur listan över element ordnas kommer inte att påverka det element vi söker efter. Så att i detta fall använda en sorterad eller icke sorterad lista kommer inte ge något annat resultat då det sökta elementet är okänt.

Det enda fallet, alltså medelfallet, är ordnat O(n) ty elementet existerar och antalet jämförelser för att finna detsamma varierar

Tillvägagångssätt

Det enda fallet

För att erhålla ett fall, medelfallet, förses sökfunktionen med en okänd mängd element.

Algoritmen

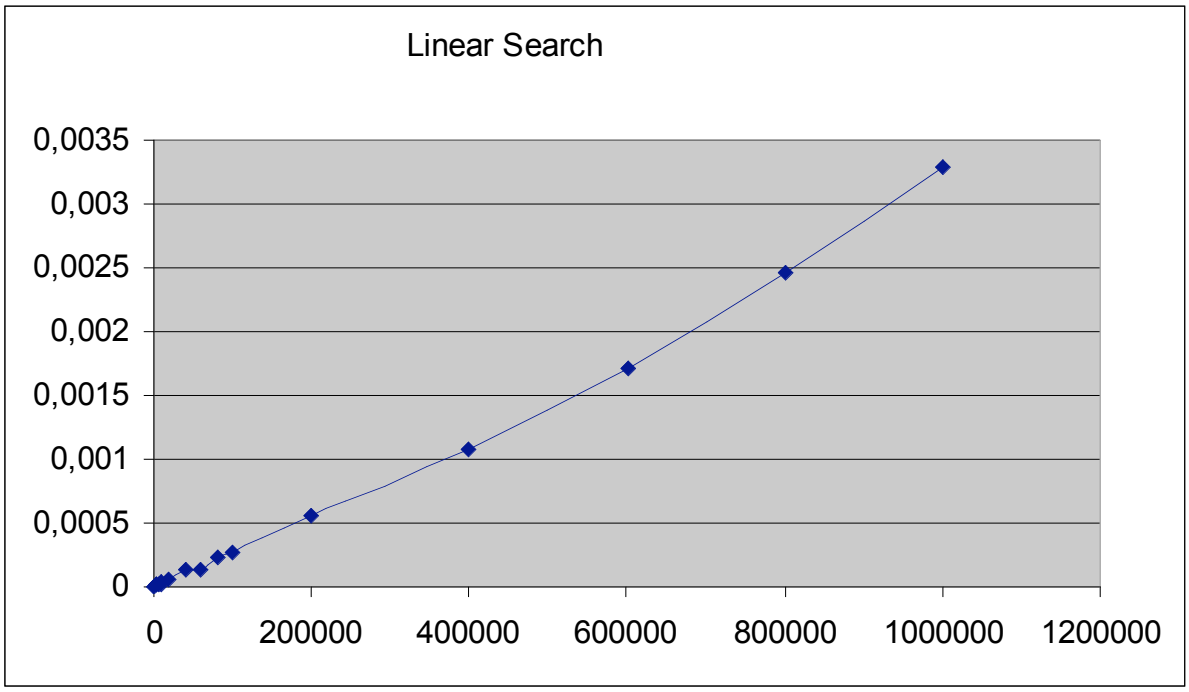
Algoritmen använder sig av en array.

Mätvärden

Tester för 1000, 2000, 4000, 6000, 8000, 10000, 20000, 40000, 60000, 80000, 100000, 200000, 400000, 600000, 800000 och 1000000 element utfördes med tio (10) tester för varje storlek av element. Därefter räknades, för varje elementmängd, ett medelvärde ut.

Det enda fallet

Antal element	**Medelkörningstid (s)**
1000	7,98702E-06
2000	1,268E-05
4000	1,3995E-05
6000	2,501E-05
8000	3,31879E-05
10000	1,8644E-05
20000	5,99384E-05
40000	0,000126
60000	0,000132108
80000	0,000221181
100000	0,000272083
200000	0,00055721
400000	0,0010848
600000	0,00171621
800000	0,00246358
1000000	0,00328131



Analys

Det enda fallet

Antal element	Medelkörningstid (s) [T]	T/n	T/n^2
1000	7,98702E-06	7,99E-09	7,98702E-12
2000	1,268E-05	6,34E-09	3,17E-12
4000	1,3995E-05	3,50E-09	8,74688E-13
6000	2,501E-05	4,17E-09	6,94722E-13
8000	3,31879E-05	4,15E-09	5,18561E-13
10000	1,8644E-05	1,86E-09	1,8644E-13
20000	5,99384E-05	3,00E-09	1,49846E-13
40000	0,000126	3,15E-09	7,875E-14
60000	0,000132108	2,20E-09	3,66967E-14
80000	0,000221181	2,76E-09	3,45595E-14
100000	0,000272083	2,72E-09	2,72083E-14
200000	0,00055721	2,79E-09	1,39303E-14
400000	0,0010848	2,71E-09	6,78E-15
600000	0,00171621	2,86E-09	4,76725E-15
800000	0,00246358	3,08E-09	3,84934E-15
1000000	0,00328131	3,28E-09	3,28131E-15

Här kan ses att T/n konvergerar. Detta tyder på att det enda fallet för linerar search är av storleksordningen n. Detta ger även grafen över medelkörningstiden någorlunda stöd för.

Resultat

Det enda fallet

Enligt antagandet för det enda fallet att vi söker efter ett godtyckligt element i listan. Analysen, se föregående rubrik, finner att det enda fallet för linear search är av ordning **O(n)**.

Diskussion

Att det bara finns ett fall för sökningen var svårtolkat. De mest uppenbara fallen, när man väljer vilket element man söker efter, ger meningslösa testfall.

Binary Search

På samma sätt som i Linear Search finns det enbart ett fall som kan uppstå för Binary Search dock måste även våra element vara sorterade i detta fall. Detta innebär, eftersom algoritmen är given, att det enbart finns ett fall.

Detta ger att det enda fallet för binary search är då ett godtyckligt element i listan sökes. Var elementet finns är okänt från sökning till sökning. Tiden för sökningen är av storleksordning O(log n) eftersom det i princip kan vara så att det behövs ett maximalt antal uppdelningar för att hitta elementet.

Tillvägagångssätt

Det enda fallet

För att erhålla ett medelfall slumpas en mängd element fram, denna mängd sorteras därpå. Ett element ur mängden eftersöktes, var elementet befann sig i mängden gick inte att veta på förhand. Då sökningen startar startas även tidtagningen för funktionen, denna avslutas när sökfunktionen kört klart.

Algoritmen

Implememententationen av algoritmen hade inga speciella egenskaper.

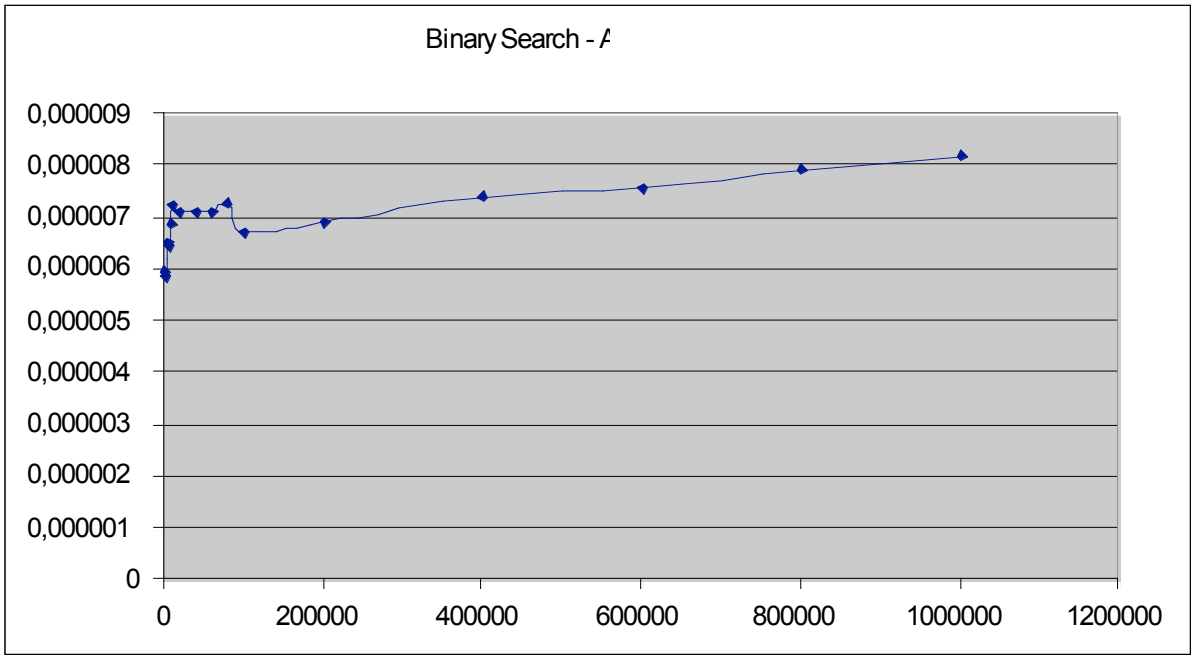
Mätvärden

Tester för 1000, 2000, 4000, 6000, 8000, 10000, 20000, 40000, 60000, 80000, 100000, 200000, 400000, 600000, 800000 och 1000000 element utfördes med tio (10) tester för varje storlek av element. Därefter räknades, för varje elementmängd, ett medelvärde ut.

Det enda fallet

Antal element	Medelkörningstid (s)
1000	5,98431E-06
2000	5,889E-06
4000	6,5327E-06
6000	6,485E-06
8000	6,8903E-06
10000	7,2479E-06
20000	7,12872E-06
40000	7,129E-06

60000	7,12872E-06
80000	7,28561E-06
100000	6,74725E-06
200000	6,92182E-06
400000	7,4162E-06
600000	7,58171E-06
800000	7,96318E-06
1000000	8,22544E-06



Analys

Det enda fallet

Antal element	Medelkörningstid (s) [T]	T/n log n	T/log n
1000	5,98431E-06	6,00486E-10	6,00486E-07
2000	5,889E-06	2,68517E-10	5,37034E-07
4000	6,5327E-06	1,36487E-10	5,45948E-07
6000	6,485E-06	8,61171E-11	5,16702E-07
8000	6,8903E-06	6,64277E-11	5,31422E-07
10000	7,2479E-06	5,45459E-11	5,45459E-07
20000	7,12872E-06	2,4947E-11	4,98941E-07
40000	7,129E-06	1,16581E-11	4,66322E-07
60000	7,12872E-06	7,48532E-12	4,49119E-07
80000	7,28561E-06	5,59134E-12	4,47307E-07
100000	6,74725E-06	4,06225E-12	4,06225E-07
200000	6,92182E-06	1,96535E-12	3,9307E-07
400000	7,4162E-06	9,96285E-13	3,98514E-07
600000	7,58171E-06	6,5832E-13	3,94992E-07
800000	7,96318E-06	5,07606E-13	4,06085E-07
1000000	8,22544E-06	4,12684E-13	4,12684E-07

Här kan ses att T/log n konvergerar. Detta tyder på att det enda fallet för binary search är av storleksordningen log n.

Resultat

Det enda fallet

Enligt antagandet skall det enda fallet för en binary search vara när ett godtyckligt element i en sorterad lista sökes. Analysen, se föregående rubrik, finner att det enda fallet för binary search är av ordning **O(log n)**.

Diskussion

Att det bara finns ett fall för sökningen var svårtolkat. De mest uppenbara fallen, när man väljer vilket element man söker efter, ger meningslösa testfall.