Testplan: sökalgoritmer

Datum:	2025-05-22
Testare:	Isak Lagerberg, Mattias Arvidsson & Joakim Sandström
Omfattning:	• Testning av sökalgoritmerna linjär sökning, binär sökning och [den algoritm ni väljer]. • Prestandamätningar på dataset med varierande storlek (1000 till 1 000 000 poster). • Analys av körtid.
Teststrategi:	Testningen utförs manuellt med hjälp av C#-programmet som utvecklats för laborationen. För varje algoritm kommer testfall skapas för att mäta körtid vid olika datamängder. Resultaten kommer dokumenteras och analyseras för att identifiera mönster och avvikelser.
Testmiljö:	• Operativsystem: Windows 11 Home ver. 24H2 • Utvecklingsmiljö: Visual Studio Enterprise 2022 (64-bit) ver. 17.13.6 • Programmeringsspråk: C#, .NET 9.0
Acceptanskriterier:	• Algoritmerna ska sortera korrekt enligt specifikation. • Körtiden ska vara inom rimliga gränser för varje algoritm och datamängd.
Risker:	• Felaktig implementering av algoritmer kan leda till inkorrekta resultat. • Stora datamängder kan orsaka minnesbrist eller lång körtid. • Manuell testning kan vara tidskrävande och innefattar risk för mänskliga fel.
Tidsplan:	Vecka 1: - Vecka 2: Implementering av sökalgoritmer och utförande av testfall för sökning. Analys och dokumentation av resultat.

<table>
<tr><th>Test no.</th><th>Testnamn</th><th>Syfte med test</th><th>Testdata eller testsituation</th><th>Resultat</th><th>Kommentar</th></tr>
<tr>
<td>1</td>
<td>BinarySearch i början av sorterad lista</td>
<td>Testa algoritmen BinarySearch där målvärdet finns i början av listan</td>
<td>Algoritm: BinarySearch
Målvärde:
<table>
<tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr>
<tr><td>1</td><td>100</td><td>123</td><td>100</td></tr>
<tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr>
<tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr>
<tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr>
<tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr>
<tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr>
</table></td>
<td><table>
<tr><th>Testfall</th><th>Resultat (µs)</th></tr>
<tr><td>1</td><td>4,40</td></tr>
<tr><td>2</td><td>4,37</td></tr>
<tr><td>3</td><td>4,58</td></tr>
<tr><td>4</td><td>4,48</td></tr>
<tr><td>5</td><td>6,45</td></tr>
<tr><td>6</td><td>3,92</td></tr>
</table></td>
<td>BinarySearch presterar stabilt oavsett liststorlek när målvärdet finns tidigt i listan. Eftersom binärsökning inte är beroende av värdets position, utan snarare av antalet iterationer ($\log_2 n$), är tiden konsekvent låg även för stora mängder data.</td>
</tr>
<tr>
<td>2</td>
<td>ExponentialSearch i början av sorterad lista</td>
<td>Testa algoritmen ExponentialSearch där målvärdet finns i slutet av listan</td>
<td>Algoritm: ExponentialSearch
Målvärde:
<table>
<tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr>
<tr><td>1</td><td>100</td><td>123</td><td>100</td></tr>
<tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr>
<tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr>
<tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr>
<tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr>
<tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr>
</table></td>
<td><table>
<tr><th>Testfall</th><th>Resultat (µs)</th></tr>
<tr><td>1</td><td>1,84</td></tr>
<tr><td>2</td><td>1,65</td></tr>
<tr><td>3</td><td>1,51</td></tr>
<tr><td>4</td><td>3,08</td></tr>
<tr><td>5</td><td>4,52</td></tr>
<tr><td>6</td><td>5,37</td></tr>
</table></td>
<td>ExponentialSearch identifierar snabbt intervall och övergår till binärsökning. När målvärdet finns tidigt i listan, är prestandan mycket god. Ökningen i tid för större datamängder är marginell, vilket tyder på god skalbarhet.</td>
</tr>
</table>

<table>
<tr><th>Test no.</th><th>Testnamn</th><th>Syfte med test</th><th>Testdata eller testsituation</th><th>Resultat</th><th>Kommentar</th></tr>
<tr>
<td>3</td>
<td>InterpolationSearch efter värde som ej finns</td>
<td>Testa algoritmen InterpolationSearch genom att söka efter ett värde som inte finns</td>
<td>Algoritm: InterpolationSearch
Målvärde:
<table>
<tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr>
<tr><td>1</td><td>100</td><td>123</td><td>100</td></tr>
<tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr>
<tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr>
<tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr>
<tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr>
<tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr>
</table>
</td>
<td>
<table>
<tr><th>Testfall</th><th>Resultat (µs)</th></tr>
<tr><td>1</td><td>3,14</td></tr>
<tr><td>2</td><td>3,05</td></tr>
<tr><td>3</td><td>3,32</td></tr>
<tr><td>4</td><td>3,77</td></tr>
<tr><td>5</td><td>5,42</td></tr>
<tr><td>6</td><td>7,65</td></tr>
</table>
</td>
<td>InterpolationSearch presterar väl på fördelade heltal men påverkas mer negativt av att målvärdet inte finns. Den försöker estimera positioner men måste ändå söka igenom intervallet, vilket ger ökande söktid med datamängd.</td>
</tr>
<tr>
<td>4</td>
<td>JumpSearch efter värde som ej finns</td>
<td>Testa algoritmen JumpSearch genom att söka efter ett värde som inte finns</td>
<td>Algoritm: JumpSearch
Målvärde:
<table>
<tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr>
<tr><td>1</td><td>100</td><td>123</td><td>100</td></tr>
<tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr>
<tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr>
<tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr>
<tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr>
<tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr>
</table>
</td>
<td>
<table>
<tr><th>Testfall</th><th>Resultat (µs)</th></tr>
<tr><td>1</td><td>2,62</td></tr>
<tr><td>2</td><td>2,86</td></tr>
<tr><td>3</td><td>3,97</td></tr>
<tr><td>4</td><td>6,38</td></tr>
<tr><td>5</td><td>7,94</td></tr>
<tr><td>6</td><td>19,90</td></tr>
</table>
</td>
<td>JumpSearch har konstant hoppstorlek, vilket innebär många steg för att utesluta intervall innan linjär sökning tar över. Det blir ineffektivt när värdet saknas, men ändå snabbare än linjär sökning.</td>
</tr>
</table>

<table>
<tr><th>Test no.</th><th>Testnamn</th><th>Syfte med test</th><th>Testdata eller testsituation</th><th>Resultat</th><th>Kommentar</th></tr>
<tr>
<td>5</td>
<td>LinearSearch efter värde som ej finns</td>
<td>Testa algoritmen LinearSearch genom att söka efter ett värde som inte finns</td>
<td>Algoritm: LinearSearch
Målvärde:
<table>
<tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr>
<tr><td>1</td><td>100</td><td>123</td><td>100</td></tr>
<tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr>
<tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr>
<tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr>
<tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr>
<tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr>
</table></td>
<td>
<table>
<tr><th>Testfall</th><th>Resultat (µs)</th></tr>
<tr><td>1</td><td>1,20</td></tr>
<tr><td>2</td><td>5,78</td></tr>
<tr><td>3</td><td>59,14</td></tr>
<tr><td>4</td><td>407,88</td></tr>
<tr><td>5</td><td>4071,99</td></tr>
<tr><td>6</td><td>44975,47</td></tr>
</table></td>
<td>Som väntat är LinearSearch mycket ineffektiv när värdet inte finns – den måste gå igenom hela listan. Detta syns tydligt i den drastiska ökningen av söktid, särskilt från 1 miljon till 10 miljoner poster.</td>
</tr>
<tr>
<td>6</td>
<td>Linjär sökning i början av sorterad lista</td>
<td>Testa algoritmen linjär sökning där målvärdet är i början av listan</td>
<td>Algoritm: Linjär sökning
Målvärde:
<table>
<tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr>
<tr><td>1</td><td>100</td><td>123</td><td>100</td></tr>
<tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr>
<tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr>
<tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr>
<tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr>
<tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr>
</table></td>
<td>
<table>
<tr><th>Testfall</th><th>Resultat (µs)</th></tr>
<tr><td>1</td><td>0,26</td></tr>
<tr><td>2</td><td>0,25</td></tr>
<tr><td>3</td><td>0,24</td></tr>
<tr><td>4</td><td>0,51</td></tr>
<tr><td>5</td><td>1,22</td></tr>
<tr><td>6</td><td>1,46</td></tr>
</table></td>
<td>LinearSearch fungerar optimalt i detta fall eftersom målvärdet ligger först. Den hittar direkt, vilket förklarar den extremt låga tiden även vid stora listor.</td>
</tr>
</table>

<table>
<tr><th>Test no.</th><th>Testnamn</th><th>Syfte med test</th><th>Testdata eller testsituation</th><th>Resultat</th><th>Kommentar</th></tr>
<tr><td>7</td><td>Linjär sökning i slutet av sorterad lista</td><td>Testa algoritmen linjär sökning där målvärdet är i slutet av listan</td><td>Algoritm: Linjär sökning
Målvärde:
<table><tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr><tr><td>1</td><td>100</td><td>123</td><td>100</td></tr><tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr><tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr><tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr><tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr><tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr></table></td><td><table><tr><th>Testfall</th><th>Resultat (µs)</th></tr><tr><td>1</td><td>0,41</td></tr><tr><td>2</td><td>3,50</td></tr><tr><td>3</td><td>30,09</td></tr><tr><td>4</td><td>302,22</td></tr><tr><td>5</td><td>2880,94</td></tr><tr><td>6</td><td>32663,47</td></tr></table></td><td>Prestandan för LinearSearch i detta fall är mycket beroende av liststorleken eftersom sökningen når slutet. Söktiden ökar linjärt med mängden data, vilket bekräftar den förväntade komplexiteten O(n).</td></tr>
<tr><td>8</td><td>Linjär sökning efter värde som ej finns</td><td>Testa algoritmen linjär sökning där målvärdet inte existerar i listan med syfte att mäta tid och ’misslyckade’ sökningar.</td><td>Algoritm: Binär sökning
Målvärde:
<table><tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr><tr><td>1</td><td>100</td><td>123</td><td>100</td></tr><tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr><tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr><tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr><tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr><tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr></table></td><td><table><tr><th>Testfall</th><th>Resultat (µs)</th></tr><tr><td>1</td><td>0,22</td></tr><tr><td>2</td><td>5,42</td></tr><tr><td>3</td><td>44,90</td></tr><tr><td>4</td><td>407,42</td></tr><tr><td>5</td><td>4121,95</td></tr><tr><td>6</td><td>46057,03</td></tr></table></td><td>Upprepar test 5 för att verifiera konsekvens. Resultaten matchar tidigare test, vilket stärker tillförlitligheten i mätningarna. Tiden ökar linjärt.</td></tr>
</table>

<table>
<tr><th>Test no.</th><th>Testnamn</th><th>Syfte med test</th><th>Testdata eller testsituation</th><th>Resultat</th><th>Kommentar</th></tr>
<tr>
<td>9</td>
<td>Binär sökning i början av sorterad lista</td>
<td>Testa algoritmen binära sökning där målvärdet är i början av listan</td>
<td>Algoritm: Binär sökning
Målvärde:
<table>
<tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr>
<tr><td>1</td><td>100</td><td>123</td><td>100</td></tr>
<tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr>
<tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr>
<tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr>
<tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr>
<tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr>
</table>
</td>
<td>
<table>
<tr><th>Testfall</th><th>Resultat (µs)</th></tr>
<tr><td>1</td><td>0,22</td></tr>
<tr><td>2</td><td>0,20</td></tr>
<tr><td>3</td><td>0,38</td></tr>
<tr><td>4</td><td>275,45</td></tr>
<tr><td>5</td><td>1,46</td></tr>
<tr><td>6</td><td>1,67</td></tr>
</table>
</td>
<td>Binär sökning påverkas inte av positionen av målvärdet, så även sökning i början ger mycket snabb respons. Tiden är låg även vid stora listor. Viss variation i resultat kan bero på systemresurser.</td>
</tr>
<tr>
<td>10</td>
<td>Binär sökning i slutet av sorterad lista</td>
<td>Testa algoritmen binära sökning där målvärdet är i slutet av listan</td>
<td>Algoritm: Binär sökning
Målvärde:
<table>
<tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr>
<tr><td>1</td><td>100</td><td>123</td><td>100</td></tr>
<tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr>
<tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr>
<tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr>
<tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr>
<tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr>
</table>
</td>
<td>
<table>
<tr><th>Testfall</th><th>Resultat (µs)</th></tr>
<tr><td>1</td><td>0,22</td></tr>
<tr><td>2</td><td>0,32</td></tr>
<tr><td>3</td><td>0,25</td></tr>
<tr><td>4</td><td>0,34</td></tr>
<tr><td>5</td><td>1,34</td></tr>
<tr><td>6</td><td>1,49</td></tr>
</table>
</td>
<td>Liknande resultat som test 9. Binärsökning visar god skalbarhet. Positionen påverkar inte tiden eftersom algoritmen alltid delar listan på hälften – logaritmisk tillväxt.</td>
</tr>
</table>

<table>
<tr><th>Test no.</th><th>Testnamn</th><th>Syfte med test</th><th>Testdata eller testsituation</th><th>Resultat</th><th>Kommentar</th></tr>
<tr>
<td>11</td>
<td>Binär sökning efter värde som ej finns</td>
<td>Testa algoritmen binär sökning där målvärdet inte existerar i listan med syfte att mäta tid och ’misslyckade’ sökningar.</td>
<td>
Algoritm: Binär sökning
Målvärde:
<table>
<tr><th>Testfall</th><th>Element</th><th>Seed</th><th>Upprepningar</th></tr>
<tr><td>1</td><td>100</td><td>123</td><td>100</td></tr>
<tr><td>2</td><td>1000</td><td>123</td><td>100</td></tr>
<tr><td>3</td><td>10 000</td><td>123</td><td>100</td></tr>
<tr><td>4</td><td>100 000</td><td>123</td><td>100</td></tr>
<tr><td>5</td><td>1 000 000</td><td>123</td><td>100</td></tr>
<tr><td>6</td><td>10 000 000</td><td>123</td><td>100</td></tr>
</table>
</td>
<td>
<table>
<tr><th>Testfall</th><th>Resultat (µs)</th></tr>
<tr><td>1</td><td>0,23</td></tr>
<tr><td>2</td><td>0,81</td></tr>
<tr><td>3</td><td>0,25</td></tr>
<tr><td>4</td><td>0,60</td></tr>
<tr><td>5</td><td>2,24</td></tr>
<tr><td>6</td><td>2,48</td></tr>
</table>
</td>
<td>Resultatet ligger i samma nivå som när värdet finns. Binärsökning hittar inte värdet men genomför samma mängd iterationer (log n). Detta visar robust prestanda även vid misslyckade sökningar.</td>
</tr>
</table>