Uppgift 1 - Textsökning

Avancerade metoder för text- och bildbehandling DA357A

Therése Larsson

Värt att nämna: För denna uppgift har jag implementerat en naiv textsökning som hittar samtliga matchningar, dvs. sökningen avbryts inte så fort en matchning hittas. Jag hade svårt att uppfatta från uppgiftsbeskrivningen om detta var meningen eller om man skulle bryta sökningen efter att ha hittat den första matchningen. Många medstudenter hade konträra uppfattningar om saken.

Analytiskt i värsta fall

Tidskomplexiteten för naiv textsökning (/brute-force substring search) är beroende på antalet jämförelser som görs. För naiv textsökning behövs det i värsta fall n*m antal jämförelser, där m motsvarar sökningsfrasens längd och n motsvarar textmängdens längd (den text där man söker efter sökningsfrasen). Tidskomplexiteten blir då O(n*m). Ett värsta fall kan inträffa när både sökningsfrasen och textmängden har mycket lika och upprepande teckensekvenser, t.ex. om båda strängarna skulle innehålla “aaaaa”, och sluta på ett “b” och vi söker strängen “aab”. Således jämförs alla tecken i sökningsfrasen med alla tecken i textmängden, för samtliga möjliga matchningar. Vi kommer inte hitta vår matchning förens i slutet av textmängden. Då får vi n*m jämförelser, vilket är ett värsta fall scenario.

Experimentellt i värsta fall

Teckenlängd (n - textmängd)	Teckenlängd (m - sökningsfras)	Antal teckenjämförelser	Antal matchningar	Exekveringstid (sekunder)
100	3	297	1	0.0
1000	3	2997	1	0.001
10000	3	29997	1	0.002
100000	3	299997	1	0.009
1000000	3	2999997	1	0.02

Om vi tittar på mätvärdena från teckenlängden för n, teckenlängden för m och antal teckenjämförelser kan man tydligt se att antalet teckenjämförelser kommer mycket nära n*m, vilket verkar stämma med den analytiska tidskomplexiteten i värsta fall. Sökningen går dock så pass snabbt att det är svårt att göra rimliga jämförelser kring exekveringstiderna.

Experimentellt för typisk sökning

Teckenlängd (n - textmängd)	Teckenlängd (m - sökningsfras)	Antal teckenjämförelser	Antal matchningar	Exekveringstid
100	3	112	3	0.0
1000	3	1099	16	0.0

10000	3	10977	141	0.002
100000	3	110684	1452	0.01
1000000	3	1094104	12111	0.024

Om man tittar på mätvärdena för tidskomplexiteten är det svårt att se betydande skillnader från mätvärdena från experimentellt i värsta fall. Felet till detta kan jag inte förklara på ett adekvat sätt. Om jag skulle reflektera skulle det kunna bero på det arbete som en dator utför vid en given tidpunkt, t.ex. hur många program som körs samtidigt och hur mycket arbetsminne de upptar osv, som kan påverka exekveringstiden för ett program. En dator exekverar sällan en process i exakt samma hastighet varje gång, utan normalt sett så varierar exekveringshastigheten trots samma indata.

Det går att se tydliga skillnader i antalet teckenjämförelser. I detta fall görs det mycket färre jämförelser (vilket borde minska exekveringstiden), samtidigt som algoritmen lyckas hitta matchningarna. Eftersom algoritmen hittar samtliga matchningar, så kommer hela texten man söker i att gås igenom i vilket fall som helst. Hade algoritmen stannat efter att ha hittat första matchningen, så borde exekveringstiden för denna testning (experimentell typisk sökning) rimligtvis varit mycket kortare än för värsta fallet. I värsta fallet jag konstruerade hittas ingen matchning förens i slutet av textmängden man söker i. I en typiskt sökning med riktig data så hittas matchningen rimligtvis mycket snarare än i slutet (förutsatt att matchningen finns), och exekveringstiden blir i så fall kortare.