Komplexiteten för merge sort

Merge sort

Problem

Det finns n element i en sekvens. Positionerna för elementen är $1, 2, 3, \cdots, n$. Dessa element ska sorteras med ”merge sort”.

Algoritm

Algoritmen ”merge sort” delar den sekvens som ska sorteras i två (ungefär) lika långa delsekvenser. Dessa delsekvenser sorteras oberoende av varandra. Om en delsekvens har fler än ett element så sorteras den rekursivt, annars har den bara ett element och den är redan sorterad.

Utifrån två sorterade delsekvenser skapas en ny sorterad sekvens, som omfattar (alla) element från de båda delsekvenserna. Dettta görs med merge rutinen. Den här rutinen använder en extra vektor, som är lika stor som den ursprungliga sekvensen.

I merge rutinen går man genom delsekvenserna från början till slut, och jämför deras element. Det element som är mindre väljs (om aktuella elementen är likadana, så väljs ett av dem), och kopieras till den nya vektorn. När alla element från den ena delsekvensen kopierats, kopieras utan någon jämförelse även de element som återstår i den andra delsekvensen. På så sätt erhålls element från båda delsekvenserna sorterade på den nya platsen. Dessa element kopieras sedan till motsvarande del av den ursprungliga platsen.

Elementär operation

För att bestämma algoritmens tidskomplexitet, kan en elementjämförelse i merge rutinen väljas som en elementär operation. Komplexitetsfunktionen ska ge antalet jämförelser för olika längder av sekvensen.

Tidskomplexiteten i värsta fall

(eng. worst-case time complexity)

Antagande: alla element är olika.

Värsta fall i merge rutinen

Värsta fall i merge sort inträffar när det inträffar värsta fall i merge rutinen. Antalet jämförelser i merge rutinen i värsta fall är $n - 1$. Det händer t ex när det element som väljs växelvis kommer från de två delsekvenserna (först från den ena delsekvensen, sedan från den andra, igen från den första delsekvensen, o s v).

Efter $n - 1$ jämförelser är $n - 1$ element valt, så det återstår ett element till. Det väljs utan någon jämförelse, som betyder att det inte kan finnas fler än $n - 1$ jämförelser.

Fallet när n är potens av 2

Sekvensen delas i två lika långa delsekvenser, som i sin tur delas i två lika långa delar, o s v. Delningen i två lika långa delsekvenser fortsätter tills sekvenser med bara ett element erhålls.

Jämförelser vid sorteringen av en sekvens omfattar de jämförelser som utförs vid sorteringen av den ena delsekvensen, de jämförelser som utförs vid sorteringen av den andra delsekvensen och de jämförelser som utförs i merge rutinen i samband med dessa två delsekvensen. Om det totala antalet jämförelser betecknas med W, så gäller följande ekvation:

$$W(n) = W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + n - 1$$

$$W(n) = 2W\left(\frac{n}{2}\right) + n - 1$$

Dessa ekvationer gäller i fall att det finns minst två element i den sekvens som sorteras, så det går att dela den i två delsekvenser. Om det bara är ett element i en sekvens, så är denna sekvens redan sorterad. Det krävs inga jämförelser. Därför:

$$W(1) = 0$$

Antalet jämförelser kan alltså beskrivas med följande differensekvation:

$$W(n) = 2W\left(\frac{n}{2}\right) + n - 1, \qquad n > 1,$$

$$W(1) = 0$$

Lösningen till denna differensekvation ger antalet jämförelser som merge sort utför i värsta fall. Det här antalet är:

$$W(n) = nlog_2 n - (n-1)$$

$$W(n) \in \Theta(nlog_2 n)$$

Fallet när *n* inte nödvändigtvis är potens av *2*

Om det inte finns någon garanti att *n* är potens av *2*, så kan man inte räkna med det att den aktuella sekvensen delas i två lika långa delar. Längder av motsvarande delsekvenser är i så fall $\lfloor n/2 \rfloor$ och $\lceil n/2 \rceil$ ($n/2$ avrundat neråt och uppåt). Motsvarande differensekvation är:

$$W(n) = W\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + W\left(\left\lceil \frac{n}{2} \right\rceil\right) + n - 1, \qquad n > 1,$$

$$W(1) = 0$$

Denna differensekvation är inte lätt att lösas exakt, men det går att bevisa att även i det här fallet gäller:

$$W(n) \in \Theta(nlog_2 n)$$

Alltså, komplexiteten för merge sort är i värsta fall $\Theta(nlog_2 n)$.

Tidskomplexiteten i ett genomsnittligt fall

(eng. average-case time complexity)

Antagande: alla element är olika och alla ordningar i sekvensen är lika sannolika (slumpmässig ordning).

Fallet när *n* är potens av *2*

Det kan bevisas att komplexitetsfunktionen i detta fall är:

$$A(n) = nlog_2 n - 1.26n$$

$$A(n) \in \Theta(nlog_2 n)$$

Tidskomplexiteten för stora *n* är i genomsnitt ungefär som tidskomplexiteten i värsta fall. Merge sort är inte alldeles känslig på ordningen av element i den sekvens som sorteras.

Fallet när *n* inte nödvändigtvis är potens av *2*

Algoritmen är $\Theta(nlog_2 n)$ även när *n* inte nödvändigtvis är potens av *2*.

Minneskomplexiteten

(eng. memory complexity, extra space usage)

Algoritmen använder en extra vektor vid sorteringen. Denna vektor är av samma storlek som den sekvens som sorteras, alltså den har storlek *n*.

Algoritmen kräver även minne på stacken. Metoden anropar sig själv i samband med en delsekvens, som i sin tur anropar sig själv i samband med sin delsekvens, o s v. Anropskedjan fortsätter tills delsekvensen med bara ett element nås. Antalet

records på stacken kan som maximalt vara $\lceil log_2 n \rceil$. För stora n kan detta stackminne försummas i jämförelse med det minne som krävs för en extra vektor. Därför är minneskomplexiteten:

$$M(n) \approx n$$

$$M(n) \in \Theta(n)$$