Komplexiteten för quick sort

Quick sort

Problem

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

Algoritm

Algoritmen ”quick sort” använder en splitringsrutin (eng. partition), som reorganiserar sekvensen på ett speciellt sätt. Rutinen sätter det första elementet i sekvensen på rätt position (sorterar det), och samtidigt omplacerar de andra elementen i sekvensen. Det första elementet används som ett splitringselement (eng. pivot), som läggs på en splittringsposition (eng. pivotpoint). Vid omplaceringen hamnar de element som är mindre än eller lika med splittringselementet framför splittringspositionen, och de element som är större än detta element hamnar efter splittringspositionen. Sekvensen splitras i två delsekvenser (framför och efter splittringspositionen), som sedan sorteras rekursivt oberoende av varandra. Detta görs på varje rekursiv nivå, och fortsätter så länge de delsekvenser som erhålls har minst två element. När en delsekvens bara har ett element, så är den redan sorterad.

I splittringsrutinen används två pekare, en framåtgående pekare som först placeras på det andra elementet i sekvensen, och en bakåtgående pekare som först placeras på sista element i sekvensen. De element som pekas med frampekaren jämförs med det första elementet i sekvensen, och frampekaren flyttas fram så länge de utpekade elementen är minde än eller lika med det första elementet. Frampekaren stannar på den första positionen där ett element som är större än det första elementet ligger. De element som pekas med bakåtpekaren jämförs med det första elementet i sekvensen, och bakåtpekaren flyttas bakåt så länge de utpekade elementen är större än det första elementet. Bakåtpekaren stannar på den första positionen där ett element som är mindre än eller lika med det första elementet ligger. Nu byter de utpekade elementen sina platser. Pekarna flyttas igen på samma sätt, och när de stannar omplaceras de utpekade elementen. Detta fortsätter tills pekarna går förbi varandra. Då pekar bakåtpekaren till den position där första elementet ska ligga. Därför byter man platser mellan det första elementet och det element som pekas med bakåtpekaren.

Elementär operation

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

Tidskomplexiteten i värsta fall

(eng. worst-case time complexity)

Tidskomplexiteten för splittringsrutinen

Alla element som ligger efter första positionen jämförs med det första elementet i sekvensen. Detta gäller på varje nivå av rekursionen. Om antalet element i den aktuella sekvensen är n, så är det antalet jämförelser $n - 1$. Tidskomplexiteten av splittringsrutinen är:

$$T(n) = n - 1, \qquad n > 0$$

Tidskomplexiteten för redan sorterade sekvenser

Om antalet element i den aktuella sekvensen betecknas med n och splittringspositionen med p (pivotpoint), så finns $p - 1$ element i den ena delsekvensen och $n - p$ element i den andra delsekvensen. Detta gäller på varje rekursiv nivå för $n > 1$.

Jämförelser vid sorteringen av en sekvens omfattar de jämförelser som utförs i splittringsrutinen, de jämförelser som utförs vid sorteringen av den ena delsekvensen och de jämförelser som utförs vid sorteringen av den andra delsekvensen. Om det totala antalet jämförelser betecknas med T, så gäller följande ekvation:

$$T(n) = n - 1 + T(p - 1) + T(n - p)$$

Det går att bevisa att antalet jämförelser är maximalt när sekvensen redan är sorterad eller omvänt sorterad. I sådana fall är antigen $p = 1$ eller $p = n$. Om något av dessa värden sätts in i föregående ekvation, omvandlas den till följande ekvation:

$$T(n) = n - 1 + T(0) + T(n-1)$$

Eftersom antalet jämförelser för $n = 0$ är 0, så erhålls följande differensekvation:

$$T(n) = T(n-1) + n - 1, \qquad n > 0$$

$$T(0) = 0$$

Lösningen till denna ekvation ger antalet jämförelser som quick sort utför om sekvensen är sorterad eller omvänt sorterad. Det här antalet är:

$$T(n) = \frac{n\,(n-1)}{2}$$

Tidskomplexiteten i värsta fall

Det kan inte finnas fler jämförelser än i fall när sekvensen redan är sorterad. Det maximala antalet jämförelser $W(n)$ är:

$$W(n) = \frac{n\,(n-1)}{2}$$

Detta ska bevisas med matematisk induktion.

Induktionsbas

Påståendet gäller för $n = 1$, eftersom:

$$W(1) = \frac{1\,(1-1)}{2}$$

$$W(1) = 0$$

Det är sant, eftersom för sekvenser med bara ett element utförs inga jämförelser.

Induktionshypotes

Anta att påståendet gäller för alla sekvenser vars storlek är mindre än n. Alltså:

$$W(k) = \frac{k\,(k-1)}{2}, \qquad 1 \leq k < n$$

Induktionssteg

Det ska bevisas att påståendet även gäller för sekvenser av storlek n. Det ska bevisas att:

$$W(n) = \frac{n\,(n-1)}{2}$$

För antalet jämförelser för en sekvens med n element gäller följande ekvation:

$$T(n) = T(p-1) + T(n-p) + n - 1$$

p betecknar splittringspositionen. Eftersom $p - 1 < n$ och $n - p < n$, så gäller det enligt induktionshypotesen:

$$T(p-1) \leq W(p-1) = \frac{(p-1)\,(p-2)}{2}$$

$$T(n-p) \leq W(n-p) = \frac{(n-p)\,(n-p-1)}{2}$$

Då gäller det följande uppskattning för $1 \leq p \leq n$:

$$T(n) \leq W(p-1) + W(n-p) + n - 1 = \frac{(p-1)(p-2)}{2} + \frac{(n-p)(n-p-1)}{2} + n - 1$$

$$T(n) \leq \frac{n(n-1)}{2} + (n-p)(1-p) \leq \frac{n(n-1)}{2}$$

$$T(n) \leq \frac{n(n-1)}{2}$$

Alltså måste antalet jämförelser vara mindre än eller lika med *n(n-1)/2.* Eftersom detta antal uppnås när sekvensen redan är sorterad, så gäller det i värsta fall:

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

Slut av induktionsbeviset

Tidskomplexiteten av quick sort i värsta fall är:

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

$$W(n) \in \Theta(n^2)$$

Quick sort är i värsta fall en $\Theta(n^2)$ algoritm.

Tidskomplexiteten i ett genomsnittligt fall

(eng. average-case time complexity)

Antagande: Alla ordningar i sekvensen är lika sannolika (slumpmässig ordning).

Om p är splittringspositionen, så gäller för tidskomplexiteten:

$$T(n) = T(p-1) + T(n-p) + n - 1$$

När algoritmen används många gånger, så är alla splittringspositioner $p\,(1 \leq p \leq n)$ lika sannolika. Sannolikheten för en viss splittringsposition är *1/n.* Antalet jämförelser i genomsnitt för en given p omfattar det genomsnittliga antalet jämförelser vid sorteringen av delsekvenserna och antalet jämförelser i splittringsrutinen. Därför är tidskomplexiteten i genomsnitt:

$$A(n) = \sum_{p=1}^{n} \frac{1}{n} (A(p-1) + A(n-p)) + n - 1$$

Denna ekvation kan transformeras så här:

$$A(n) = \sum_{p=1}^{n} \frac{1}{n} A(p-1) + \sum_{p=1}^{n} \frac{1}{n} A(n-p) + n - 1$$

$$A(n) = \frac{1}{n} \sum_{p=1}^{n} A(p-1) + \frac{1}{n} \sum_{p=1}^{n} A(n-p) + n - 1$$

$$A(n) = \frac{1}{n} \sum_{p=1}^{n} A(p-1) + \frac{1}{n} \sum_{p=1}^{n} A(p-1) + n - 1$$

$$A(n) = \frac{2}{n} \sum_{p=1}^{n} A(p-1) + n - 1$$

$$nA(n) = 2\sum_{p=1}^{n} A(p-1) + n(\,n-1)$$

Denna ekvation gäller även för *n – 1*:

$$(n-1)A(n-1) = 2\sum_{p=1}^{n-1} A(p-1) + (\,n-1)(n-2)$$

Om denna ekvation subtraheras från föregående ekvation erhålls:

$$nA(n) - \,(n-1)A(n-1) = 2A(n-1) + \,2(n-1)$$

Denna ekvation kan omvandlas till följande differensekvation:

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

$$A(1) = 0$$

Lösningen till den här ekvationen är:

$$A(n) \approx 1.38(n+1)log_2 n$$

$$A(n)\,\epsilon\,\Theta(nlog_2 n)$$

I genomsnitt är quick sort en snabb algoritm för stora *n*.

Minneskomplexiteten

(eng. memory complexity, extra space usage)

I fall att sekvensen är nästan sorterad (eller omvänt sorterad) blir många nivåer av rekursionen och många records på systemstacken. I värsta fall kan det finnas *n – 1* records. Det betyder att minneskomplexiteten i värsta fall är:

$$M(n) = \,n-1$$

$$M(n)\,\epsilon\,\Theta(n)$$

Om element i sekvensen är slumpmässigt ordnade, så delas sekvensen i två ungefär lika stora delsekvenser. I så fall blir minneskomplexiteten:

$$M(n) \,\approx\, log_2 n$$

$$M(n)\epsilon\,\Theta(\,log_2 n)$$