Komplexiteten för insättningssorteringen

Insättningssorteringen

Problem

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

Algoritm

Algoritmen ”insättningssortering” sätter in element i tur och ordning, från det element som ligger på positionen 2 till det element som ligger på positionen n. Elementen sätts in på rätt ställen relativt de element som redan är sorterade.

Det element som ska sättas in kopieras till en temporär variabel, och på dess plats uppstår ”ett hål”. Elementet jämförs sedan i tur och ordning med de element som finns framför det, i riktning mot sekvensens början. Varje element som är större än det aktuella elementet flyttas en position fram, och på dess plats uppstår ett hål – det redan befintliga hålet flyttas en position bakåt. Man fortsätter med jämförelser tills ett element som är mindre än eller lika med det aktuella elementet påträffas. Detta element flyttas inte. Det aktuella elementet sätts in i hålet, och på så sätt sorteras relativt de element som var framför det.

Elementär operation

För att bestämma algoritmens tidskomplexitet, kan en elementjämförelse 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 inträffar när element i sekvensen är sorterade i omvänd ordning.

När element på positionen $k\ (2 \leq k \leq n)$ ska sättas in (sorteras relativt de redan sorterade elementen), så är det antalet jämförelser $k - 1$ (elementet jämförs med alla de element som redan sattes in). För att sätta in alla element, så krävs det följande antal jämförelser:

$$\sum_{k=2}^{n}(k-1)$$

Denna summa är:

$$\sum_{k=2}^{n}(k-1) = \sum_{k=1}^{n-1} k = \frac{(n-1)n}{2}$$

Alltså yttras algoritmens tidskomplexitet i värsta fall med följande komplexitetsfunktion:

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

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

Tidskomplexiteten i bästa fall

(eng. best-case time complexity)

Bästa fall inträffar när element i sekvensen redan är sorterade.

När element på positionen $k\ (2 \leq k \leq n)$ ska sättas in (sorteras relativt de redan sorterade elementen), så är det bara *1* jämförelse som krävs (elementet bara jämförs med det element som finns på föregående position, eftersom element i sekvensen redan är sorterade). För att sätta in alla element, så krävs det följande antal jämförelser:

$$\sum_{k=2}^{n} 1 = n - 1$$

Alltså yttras algoritmens tidskomplexitet i bästa fall med följande komplexitetsfunktion:

$$B(n) = n - 1$$

$$B(n) \in \Theta(n)$$

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).

När element på positionen $k\ (2 \leq k \leq n)$ ska sättas in (sorteras relativt de redan sorterade elementen), så är det lika sannolikt att det hamnar på vilken som helst av de k första positionerna. Sannolikheten för en viss position är $1/k$. Till varje position motsvarar ett antal jämförelser. Antalet jämförelser för olika positioner anges nedan.

position	antalet jämförelser
k	*1*
k - 1	*2*
k - 2	*3*
⋮	⋮
3	*k - 2*
2	*k - 1*
1	*k - 1*

Det aktuella elementet jämförs med alla föregående element (med $k - 1$ element) oavsett om det hamnar på positionen *2* eller positionen *1*.

Antalet jämförelser som krävs för att sätta in elementet på positionen k är i genomsnitt:

$$\sum_{j=1}^{k-1} \mathrm{j} \frac{1}{k} + (k-1)\frac{1}{k}$$

Denna summa är:

$$\sum_{j=1}^{k-1} \mathrm{j} \frac{1}{k} + \frac{k-1}{k} = \frac{1}{k}\sum_{j=1}^{k-1} \mathrm{j} + \frac{k-1}{k} = \frac{1}{k}\frac{(k-1)k}{2} + \frac{k-1}{k} = \frac{k-1}{2} + \frac{k-1}{k} = \frac{k+1}{2} - \frac{1}{k}$$

För att sätta in (sortera) alla element, så krävs det följande antal jämförelser:

$$\sum_{k=2}^{n} \left(\frac{k+1}{2} - \frac{1}{k}\right)$$

För tillräckligt stora n gäller:

$$\sum_{k=2}^{n} \frac{1}{k} \approx \ln n$$

I så fall blir antalet jämförelser:

$$\sum_{k=2}^{n}\left(\frac{k+1}{2}-\frac{1}{k}\right)=\sum_{k=2}^{n}\frac{k+1}{2}-\sum_{k=2}^{n}\frac{1}{k}=\frac{1}{2}\left(\sum_{k=1}^{n+1}k-3\right)-\sum_{k=2}^{n}\frac{1}{k}\approx\frac{(n+4)(n-1)}{4}-\ln n$$

Alltså, algoritmens genomsnittliga tidskomplexitet för stora sekvenser är:

$$A(n)\approx\frac{(n+4)(n-1)}{4}-\ln n$$

$$A(n)\,\epsilon\,\Theta(n^2)$$

Minneskomplexiteten

(eng. memory complexity, extra space usage)

Algoritmen använder bara minnesplats för några få variabler (förutom den plats som tas upp av inputdata – de element som sorteras). Antalet dessa variabler är oberoende av antalet element (n) i sekvensen. Därför kan algoritmens minneskomplexitet beskrivas så här:

$$M(n)\,\epsilon\,\Theta(1)$$