Algoritmer - Problem och lösningar

Tony Johansson
1MA239: Specialkurs i Matematik II
Uppsala Universitet

VT 2018

Problem 1

Visa att BINÄR$(k; k_1, k_2, \ldots, k_n)$ terminerar efter högst $\lceil \log_2 n \rceil$ jämförelser, förslagsvis enligt följande steg:

1. Låt $T(n)$ vara antalet jämförelser som krävs (i värsta fallet) om listan har längd n. Vi har $T(1) = 0$. Visa att för $n > 1$ är
$$T(n) = 1 + T\left(\left\lceil \left(\frac{n}{2}\right) \right\rceil\right). \tag{1}$$
2. Visa med induktion att om $T(n)$ uppfyller (1) så är $T(n) \leq \lceil \log_2 n \rceil$ för alla heltal $n \geq 1$.

OBS: Steg (a) och (b) ovan är ett av många sätt att lösa problemet. Om du kan visa att det krävs högst $\lceil \log_2 n \rceil$ steg på något annat sätt behöver du inte göra (a) och/eller (b).

Lösning

Låt $T(n)$ vara största möjliga antalet jämförelser som krävs för att hitta indexet i för vilket $k = k_i$ in en lista $k_1 < k_2 < \cdots < k_n$. Vi visar med induktion att
$$T(n) \leq \lceil \log_2 n \rceil.$$

Basfallet: Om $n = 1$ har vi $T(1) = 0 = \lceil \log_2(1) \rceil$, eftersom inga jämförelser krävs.

Induktionssteget: Anta att $n > 1$ och att $T(k) \leq \lceil \log_2 k \rceil$ för $k = 1, 2, \ldots, n-1$. Efter en jämförelse med en lista av längd n anropar vi (i värsta fallet) binärsökningsalgoritmen med en lista av längd $n - \lfloor n/2 \rfloor = \lceil n/2 \rceil$. Så

$$T(n) = 1 + T\left(\left\lceil \frac{n}{2} \right\rceil\right).$$

Induktionsantagandet ger

$$T(n) \leq 1 + \left\lceil \log_2 \left(\frac{n}{2}\right) \right\rceil = 1 + \lceil \log_2 n - 1 \rceil = \lceil \log_2 n \rceil.$$

Problem 2

Bubblesort som den presenteras i texten kräver alltid exakt $(n-1)^2$ jämförelser. I beviset av Proposition 2 noterar vi att när $i > 1$ är det garanterat onödigt att utföra steget $j = n$, eftersom vi vet att $a_{(n)}$ redan är det största elementet. På samma sätt är $j = n - 1$ onödigt när $i > 2$, och så vidare.

Skriv om algoritmen så att den använder exakt $1 + 2 + \cdots + (n-1) = \frac{n(n-1)}{2}$ steg, och fortfarande garanterar en sorterad lista.

Lösning

Algoritm 1 nedan undviker de $i - 1$ onödiga jämförelserna i slutet av varje runda. Den gör total $(n-1) + (n-2) + \cdots + 1 = n(n-1)/2$ jämförelser.

Algorithm 1 Bubblesort

```
1: procedure BUBBLESORT(a_1, ..., a_n)
2:     (a_(1), ..., a_(n)) ← (a_1, ..., a_n)
3:     for i = 1 to n − 1 do
4:         for j = 2 to n − i + 1 do
5:             if a_(j−1) > a_(j) then
6:                 swap(a_(j−1), a_(j))
       return (a_(1), ..., a_(n))
```

Problem 3

Låt $(a_1, \ldots, a_n)$ vara en lista av distinkta tal, $n \geq 2$, och låt $T(a_1, \ldots, a_n)$ vara antalet jämförelser som Quicksort använder för att sortera listan, om

det första elementet i listan alltid väljs som pivotelement. Visa att om $a_1 < a_2 < \cdots < a_n$, det vill säga vi matar in en sorterad lista, så är

$$T(a_1, \ldots, a_n) = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}.$$

Du behöver bara visa första likheten. Notera att $T(a_1) = 0$ och $T(\emptyset) = 0$.

Detta bevisar att QuickSort ibland kan vara lika dålig som BubbleSort. (**Ledtråd:** använd induktion.)

Lösning

Quicksort börjar med att välja ut a_1 som pivotelement. Den jämför sedan a_1 med $a_2, a_3, \ldots, a_n$ ($n-1$ jämförelser) och konstaterar att a_1 är minst. Sedan anropas Quicksort på listan $a_2 < a_3 < \cdots < a_n$. Vi får

$$T(a_1, a_2, \ldots, a_n) = n - 1 + T(a_2, a_3, \ldots, a_n).$$

När $n = 1$ är $T(a_1) = 0$ (inga jämförelser krävs). Från detta är det en enkel övning i induktion att visa att

$$T(a_1, \ldots, a_n) = \sum_{k=1}^{n-1} k.$$

Problem 4

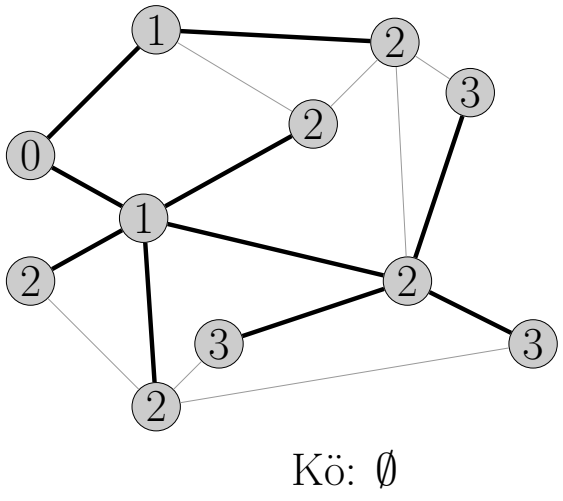
I exemplen för BFS och DFS använder vi bokstavsordning för att bestämma vilken ordning vi väljer noder i. Om vi istället använder omvänd bokstavsordning ("högre" bokstäver först) blir de resulterande träden annorlunda. Gör både BFS och DFS med omvänd bokstavsordning och markera tydligt de kanter som respektive algoritm använder.

För att vara exakt: i första steget av vår BFS lade vi i första steget de två grannarna b, c till a i kön i ordningen b, c (bokstavsordning). Om vi istället lägger dem i ordningen c, b, hur ser den sista bilden ut? På samma sätt valde vi i DFS att först gå från a till b, men jag vill veta hur det ser ut om vi istället går från a till c och konsekvent fortsätter i omvänd bokstavsordning.

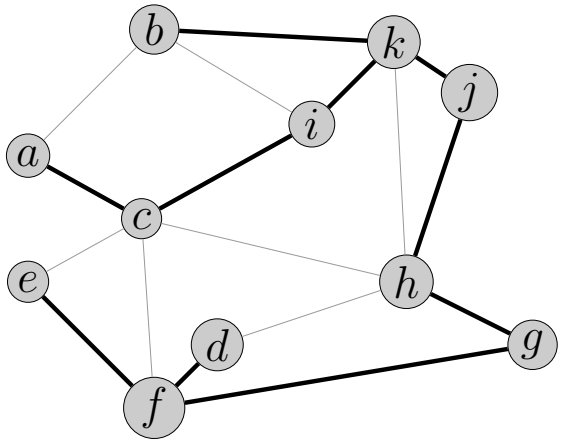
På kurshemsidan ligger en mall för att rita grafen i LaTeXmed hjälp av paketet tikz, för de som vill använda det.

Lösning

Slutbilderna är följande.



Figur 1: BFS med omvänd bokstavsordning.



Figur 2: DFS med omvänd bokstavsordning.