Algoritmer och Komplexitet ht 09. Övning 2

Dekomposition

Max och min med dekomposition I vektorn $v[1..n]$ ligger n tal. Konstruera en dekompositionsalgoritm som tar reda på det största och det minsta talet i v. Algoritmen ska använda högst $\lceil 3n/2 \rceil - 2$ jämförelser mellan v-element. Antalet tal i v behöver inte vara en tvåpotens.

Matrismultiplikation Strassens algoritm multiplicerar två $n \times n$-matriser i tid $O(n^{2.808})$ genom dekomposition i 2×2-blockmatriser. Anledningen till att Strassens algoritm går snabbare än $O(n^3)$ är att den gör sju multiplikationer istället för åtta för att bilda produktmatrisen.

En annan idé är att göra dekomposition i 3×3-blockmatriser istället. En forskare på Nada försökte för ett par år sedan hitta det minimala antalet multiplikationer som krävs för att multiplicera två 3×3-matriser. Han lyckades nästan komma fram till 22 multiplikationer. Om han hade lyckats, vilken tidskomplexitet hade det gett för multiplikation av två $n \times n$-matriser?

Komplex multiplikation Om man multiplicerar två komplexa tal $a + bi$ och $c + di$ på det vanliga sättet krävs fyra multiplikationer och två additioner av reella tal. Eftersom multiplikationer är dyrare än additioner (och subtraktioner) lönar det sig att minimera antalet multiplikationer om man ska räkna med stora tal. Hitta på en algoritm som bara använder tre multiplikationer (men fler additioner) för att multiplicera två komplexa tal.

Majoritet med dekomposition Indata är i denna uppgift en array A med n element. Konstruera och analysera en algoritm som tar reda på om något element i arrayen A är i majoritet, det vill säga förekommer minst $n/2$ gånger, och i så fall returnerar det. Algoritmen ska vara en dekompositionsalgoritm och ha tidskomplexiteten $O(n \log n)$. Enda tillåtna jämförelseoperationen på element i A är $=$. Det finns alltså ingen ordningsrelation mellan elementen.

Inuti eller utanför? Låt P vara en konvex n-hörnig polygon beskriven som en array av hörnen $p_1, p_2, \ldots p_n$ i cyklisk ordning. Konstruera en algoritm som talar om ifall en given punkt q är inuti P. Algoritmen ska gå i tid $O(\log n)$ i värsta fallet.

Lösningar

Lösning till Max och min med dekomposition
När man bara har två tal räcker det med en enda jämförelse för att både hitta det största och det minsta talet.

```
MinMax(v,i,j)=
  if i=j then return (v[i],v[i])
  else if i+1=j then
```

```
    if v[i]<v[j] then return (v[i],v[j])
    else return (v[j],v[i])
  else
    m:=Floor((j-i)/2)
    if Odd(m) then m:=m+1;
    (min1,max1):=MinMax(v,i,i+m-1);
    (min2,max2):=MinMax(v,i+m,j);
    min:=(if min1<min2 then min1 else min2);
    max:=(if max1>max2 then max1 else max2);
    return (min,max);
```

Beräkningsträdet kommer att få $\lceil n/2 \rceil$ löv och $\lceil n/2 \rceil - 1$ inre noder. Om n är jämnt är alla $n/2$ löv tvåelementsföljder och gör därför en jämförelse. Om n är udda är ett löv (det högraste) ett enstaka element som inte kräver någon jämförelse. I löven görs alltså $\lfloor n/2 \rfloor$ jämförelser. I varje inre nod görs två jämförelser, alltså sammanlagt $2\lceil n/2 \rceil - 2$ stycken. Totalt får vi $\lfloor n/2 \rfloor + 2\lceil n/2 \rceil - 2 = \lceil 3n/2 \rceil - 2$ stycken jämförelser.

Det går faktiskt att visa att problemet inte kan lösas med färre jämförelser. □

Lösning till Matrismultiplikation

Rekursionsekvationen blir $T(n) = 22 \cdot T(n/3) + O(n^2)$. Mästarsatsen ger lösningen $T(n) = O(n^{\log_3 22}) = O(n^{2.814})$. □

Lösning till Komplex multiplikation

Egen övning. □

Lösning till Majoritet med dekomposition

Om det finns ett majoritetselement måste det vara i majoritet i åtminstone ena halvan av arrayen. Rekursiv tanke: Kolla majoritet rekursivt i vänstra och högra halvan och räkna sedan hur många gånger halvarraysmajoritetselementen förekommer i hela arrayen. Om något element är i total majoritet returneras det.

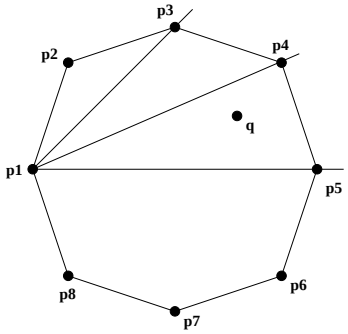
Majority($A[1..n]$) =
  if $n = 1$ **then return** $A[1]$
  $m \leftarrow \lceil n/2 \rceil$
  $v \leftarrow$ Majority($A[1..m-1]$)
  $h \leftarrow$ Majority($A[m..n]$)
  if $v = h$ **then return** v
  $vn \leftarrow 0; hn \leftarrow 0$
  for $i \leftarrow 1$ **to** n **do**
    if $A[i] = v$ **then** $vn \leftarrow vn + 1$
    else if $A[i] = h$ **then** $hn \leftarrow hn + 1$
  if $vn \geq m$ **then return** v
  if $hn \geq m$ **then return** h
  else return NULL

Tidskomplexitet: Två rekursiva anrop med halva arrayen följt av efterarbetet $O(n)$ ger med mästarsatsens hjälp tidskomplexiteten $O(n \log n)$. □

Lösning till Inuti eller utanför?

Om polygonen inte hade varit konvex hade vi räknat antalet skärningar polygonen har med en

linje från q till en punkt utanför P, vilket tar tid $O(n)$, men det har vi inte tid med här. Istället kommer vi att använda en intervallhalveringsliknande sökning för att utesluta hälften av (den återstående) polygonen i taget ända tills bara en triangel återstår. Därefter kan vi lätt (med ett konstant antal jämförelser) avgöra ifall q ligger i P. Se figur 1.



Figur 1: En konvex polygon och linjerna som algoritmen använder för att halvera den.

$InsideConvex(P, q, l, u) =$
  if $u = l + 1$ **then** $/*$ en triangel $*/$
    välj en punkt q' utanför triangeln p_1–p_l–p_u
    if linjen q–q' skär exakt en av kanterna i triangeln **then**
      return inuti
    else
      return utanför
  else
    $mid \leftarrow \lceil \frac{l+u}{2} \rceil$
    if q är på samma sida om linjen p_1–p_{mid} som p_{mid+1} **then**
      return $InsideConvex(P, q, mid, u)$
    else
      return $InsideConvex(P, q, l, mid)$

Algoritmen anropas med $InsideConvex(P, q, 2, n)$.

Om vi antar att $InsideConvex(P, q, 2, n)$ tar tid $T(n)$ så får vi rekursionsekvationen

$$T(n) = T\left(\frac{n}{2}\right) + c$$

som har lösningen $c \log n$. Tidskomplexiteten blir alltså $T(n) \in O(\log n)$. □
