DM507 Algoritmer og datastrukturer

Forår 2012

Projekt, del I

Institut for matematik og datalogi
Syddansk Universitet

20. februar, 2012

Dette projekt udleveres i tre dele. Hver del har sin deadline, således at afleveringerne, og dermed arbejdet, strækkes over hele semesteret. Deadline for del I er mandag den 19. marts. Projektet skal som udgangspunkt besvares i grupper af størrelse to. Individuelle besvarelser er tilladt, men tilskyndes ikke.

Mål

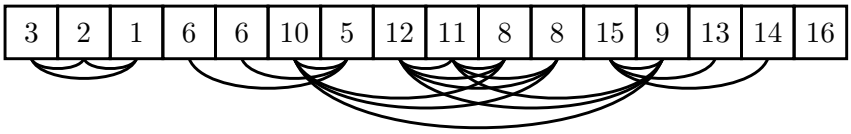
Det samlede projekt beskæftiger sig med inversioner, og med sorteringsmetoder hvis køretid afhænger af antallet af inversioner i input.

Hovedmålet for del I af projektet er at lave en algoritme, som effektivt kan tælle antallet af inversioner i et array.

Inversioner

En *inversion* i et array A af tal er et par som ikke står i sorteret orden. I dette projekt defineres sorteret som stigende (ikke-faldende) orden, og en inversion er mere præcist et par (i, j) hvor $i < j$ og $A[i] > A[j]$. *Antal inversioner* i A er antallet af par (i, j) som udgør en inversion. Da der er $\binom{n}{2} = n(n-1)/2$ par, er dette det maksimale antal inversioner i et array af længde n. Et array af længde under to har ingen inversioner.

Eksempler: et sorteret array har nul inversioner, et omvendt sorteret array har det maksimale antal inversioner $n(n-1)/2$, og nedenstående array



har 19 inversioner, nemlig de 19 indikerede par.

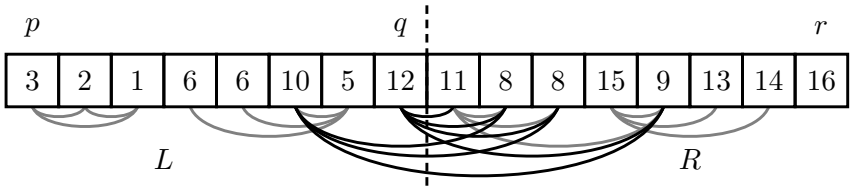
Algoritmer til at tælle inversioner i et array

En oplagt algoritme til at finde antallet af inversioner i et array bruger en dobbelt **for**-løkke, som tester alle par (i, j) (dvs. (i, j) for $1 \leq i \leq n-1$ og $i+1 \leq j \leq n$), og derfor tager $\Theta(n^2)$ tid. I del I af projektet skal vi udvikle en algoritme, som tager $\Theta(n \log n)$ tid. Algoritmen er en udvidelse af MergeSort algoritmen, og er baseret på følgende to observationer:

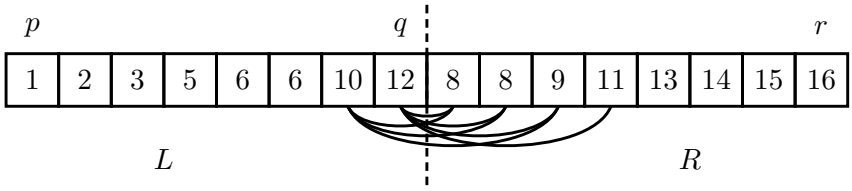
1. Hvis et (del-)array $A[p..r]$ opdeles i to dele $L = A[p..q]$ og $R = A[(q+1)..r]$ da er det samlede antal inversioner i $A[p..r]$ lig med $x + y + z$, hvor:

   - x er antallet af inversioner i L
   - y er antallet af inversioner i R
   - z er antallet af inversioner mellem L og R, dvs. inversioner (i, j) med $p \leq i \leq q$ og $q + 1 \leq j \leq r$

   som illustreret i følgende figur, hvor inversionerne i L og i R alle er vist med grå kurver, mens inversionerne mellem L og R er vist med sorte kurver:



2. Tallet z ændres ikke selv om L og R hver især sorteres, som illustreret i følgende figur:



Opgaver

1. Implementer den simple $\Theta(n^2)$ algoritme (baseret på en dobbelt **for**-løkke) til at finde antallet af inversioner i et array. Implementationen skal ligge i en fil, der hedder `SimpleInv.java`. Den skal tage sit input fra en tekstfil, der indeholder en række heltal, hver adskilt af whitespace (brug f.eks. klassen `java.util.Scanner` og metoden `nextInt()` herfra til at parse input). Den skal tage inputfilens navn fra kommandolinien, sådan at programmet efter kompilering kan kaldes således: `java SimpleInv inputfilnavn`. Den skal skrive output (antal inversioner i input) på skærmen. Der skal udskrives et heltal og intet andet (da de afleverede programmer vil blive afprøvet med automatiserede tests). Et antal eksempelfiler med input i ovenstående format findes på kursets hjemmeside.

2. Implementer MergeSort efter pseudo-koden i afsnit 2.3 i Cormen et al. Som ∞ kan bruges værdien `Integer.MAX_VALUE` fra klassen `Integer`. Det må her og i resten af projektet antages, at denne værdi ikke forekommer i input. Implementationen skal ligge i en fil der hedder `MergeSort.java`. Den skal tage sit input fra en tekstfil, der indeholder en række heltal, hver adskilt af whitespace. Den skal tage inputfilens navn fra kommandolinien, og skal skrive output på skærmen (som de sorterede tal, hver adskilt af whitespace, og intet andet).

3. Beskriv en udvidelse af metoden MERGE(A, p, q, r) fra side 31 i Cormen et al. som, udover at flette indholdet af to sorterede del-arrays $L = A[p..q]$ og $R = A[(q+1)..r]$ til ét sorteret array $A[p..r]$, også beregner tallet z. [Hint: Lad metoden have en tæller *zCounter*, og vedligehold følgende invariant: Ved starten af enhver iteration af **for**-løkken svarende til linie 12 i pseudo-koden side 31 i Cormen et al. er *zCounter* lig z minus antallet af inversioner (s,t) i $A[p..r]$ med endepunkter i $A[(p+i-1)..q]$ og $A[(q+j)..r]$ (dvs. inversioner (s,t) med $p+i-1 \leq s \leq q$ og $q+j \leq t \leq r$). Her er i og j fra pseudo-koden i Cormen et al. side 31.]

4. Argumenter for korrekthed og køretid af din udvidede metode. Mht. korrekthed er det nok at argumentere for, at z beregnes rigtigt (da argumentet for at der flettes korrekt, vil svare helt til det i bogen).

5. Beskriv en divide-and-conquer algoritme baseret på MergeSort som finder antallet af inversioner i et array i $\Theta(n \log n)$ tid. [Hint: lad algoritmen som input tage et del-array $A[p..r]$, som den *både* sorterer *og* returnerer antallet af inversioner i, og lav algoritmen rekursiv i stil med MergeSort ved at bruge de to observationer ovenfor, samt den udvidede MERGE metode.]

6. Argumenter for korrekthed og køretid af din algoritme.

7. Implementer din $\Theta(n \log n)$ algoritme baseret på MergeSort til at finde antallet af inversioner i et array. Implementationen skal ligge i en fil der hedder `FastInv.java`.

Den skal tage sit input fra en tekstfil, der indeholder en række heltal, hver adskilt af whitespace. Den skal tage inputfilens navn fra kommandolinien, og skal skrive output (antal inversioner i input, og intet andet) på skærmen.

Formalia

Lav en rapport, som indeholder dine svar på opgaverne 1–7 ovenfor. Koden fra opgaverne 1, 2 og 7 skal være passende kommenteret, skal inkluderes i rapporten som bilag, og eventuelle ikke-trivielle aspekter af implementeringen skal diskuteres i rapportens hoveddel. Der skal afleveres rapporten **i pdf-format**, samt de tre Java-programmer `SimpleInv.java`, `MergeSort.java` og `FastInv.java` som separate filer (udover at de er med på tryk i rapporten). Husk at skrive navnene på personerne i gruppen på forsiden af rapporten.

Materialet afleveres i Blackboard med værktøjet "SDU Assignment" (ikke at forveksle med "Assignment hand in", som er et andet afleveringsværktøj i Blackboard). Det kan findes under "Tools" i menuen i kursussiden i Blackboard. Menuen findes ved at klikke på det lille "dobbelt-firkant"-ikon i øverste halvdel af venstre kant af kursussiden i Blackboard (om nødvendigt maksimer det fremkomne vindue).

Aflever materialet senest:

Mandag den 19. marts, 2012, kl. 23:59.