TDT4120 Algoritmer og datastrukturer

Eksamen, 9. august 2022, 09:00–13:00

Faglig kontakt	Magnus Lie Hetland
Hjelpemiddelkode	A

Løsningsforslag

Løsningsforslagene i rødt nedenfor er *eksempler* på svar som vil gi uttelling. Det vil ofte være helt akseptabelt med mange andre, beslektede svar, spesielt der det bes om en forklaring eller lignende. Om du svarte noe litt annet, betyr ikke det nødvendigvis at du svarte feil!

På grunn av koronapandemien er dette en hjemmeeksamen med alle hjelpemidler tillatt. Følgelig bør ikke eksamenssettet ses som representativt for ordinære eksamener.

1 Hvilke to hovedmetoder har vi for å velge pivot? Er en av dem bedre, og i så fall på hvilken måte? Forklar kort med egne ord.

De to hovedmetodene representeres ved pensumprosedyrene PARTITION, der man velger et fast pivotelement (mer spesifikt, det siste elementet) og RANDOMIZED-PARTITION, der pivot velges tilfeldig.

Av disse kan man argumentere for at RANDOMIZED-PARTITION er bedre, siden den unngår at man får konsekvent verst mulig kjøretid for en gitt type input. Samtidig så gir begge prosedyrene samme kjøretid i beste og verste tilfelle.

Dette gjelder QUICKSORT; for RANDOMIZED-SELECT brukes alltid RANDOMIZED-PARTITION. SELECT har sin egen metode for å velge en pivot som er median av medianer; dette kan godt nevnes, men er ikke naturlig å inkludere som en av de to hovedmetodene.

2 I det følgende, anta:

- $a_1 = a_2$, $b_1 = b_2$, $c_1 = c_2$ og $d_1 = d_2$
- $a_1 < b_1 < c_1 < d_1$

Sorter følgende sekvens med INSERTION-SORT:

$$\langle d_2, a_1, b_1, c_2, b_2, a_2, c_1, d_1 \rangle$$

Oppgi den resulterende sekvensen. Hvilken egenskap er det vi ser at algoritmen har her? Gjelder det generelt (dvs., for alle instanser)? Forklar kort.

Selv om f.eks. a_1 er lik a_2, må du skille mellom dem, siden det kan være satellittdata knyttet til dem.

Her er resultatet $\langle a_1, a_2, b_1, b_2, c_2, c_1, d_2, d_1 \rangle$. Egenskapen som demonstreres er stabilitet. Ja, INSERTION-SORT er stabil generelt. Det ser vi som følger: Når et element skal settes inn, vil det ha forekommet til høyre for alle like verdier som allerede er sorterte. I løkka som går nedover for å finne riktig plass, så stopper vi straks vi finner en lik verdi, og går ikke forbi denne; dermed vil ikke like verdier bytte plass.

3 Hva er kjeding (*chaining*)? Forklar kort, med egne ord, hva det brukes til og hvordan det fungerer.

Dette er en form for kollisjonshåndtering i hashtabeller, der nøkler med lik hashverdi plasseres i en lenket liste.

Her kan man gjerne forklare noe mer. Se pensum for beskrivelse.

4 Dine venner Lurvik og Smartnes vil implementere Prims algoritme med en binærhaug (*binary heap*) som prioritetskø Q. Lurvik bygger Q ved å legge inn én og én node, med MIN-HEAP-INSERT, mens Smartnes mener det gir bedre asymptotisk kjøretid totalt sett å bruke BUILD-MIN-HEAP. Hva mener du? Forklar kort.

Om man kun regner verste kjøretid for MIN-HEAP-INSERT, vil BUILD-MIN-HEAP bli asymptotisk raskere, isolert sett, men domineres av resten, så den totale kjøretiden blir den samme. Om man ser nærmere på hva som settes inn, ser man at alle nøklene er identiske, så hver *insert* tar konstant tid, og de to løsningene vil være omtrent identiske, også i praksis.

5 Diskuter kort likheter og forskjeller mellom BFS, PRIM og DIJKSTRA.

Her kan man f.eks. diskutere at de alle bruker en form for kø/prioritetskø, der nodenes prioritetes settes forskjellig. I BFS er prioriteten avstanden til start, implisitt gitt av at man bruker en FIFO-kø, og legger inn noder i avstandsrekkefølge. For Prim er prioriteten lengden på nodens korteste kant til treet som er bygget så langt, og for Dijkstra er prioriteten det beste avstandsestimatet funnet så langt.

En viktig forskjell, slik algoritmene er presentert i pensum, er at BFS legger inn noder etterhvert, mens Prim og Dijkstra bygger køen først, og justerer prioritet etterhvert.

6 Forenkle følgende uttrykk:

$$O(n^a) + \Omega(n^b) + \Theta(n^c)$$

Uttrykk svaret med asymptotisk notasjon. Forklar kort.

Du kan anta at a, b og c er positive heltallskonstanter.

$\Omega(n^b + n^c)$.

Minst så stort som den største nedre grensen, og ingen øvre grense.

Her kan man gjerne forklare litt mer. Flere forklaringer vil aksepteres.

7 Hva er kjøretiden til følgende prosedyre, Xyzzy, om den kalles på en tabell $A[1..n]$, med $p = 1$ og $r = n$? Du kan anta at Frozz(A, p, r) bruker $A[p..r]$ som input, og har kvadratisk kjøretid. Frozz endrer ikke på innholdet i A. Forklar kort hvordan du kommer frem til svaret.

```
Xyzzy(A, p, r)
 1  x = 1
 2  if p < r
 3      q = ⌊(p + r)/2⌋
 4      x = x + Xyzzy(A, p, q)
 5      x = x + Xyzzy(A, q + 1, r)
 6      x = x + Frozz(A, p, q)
 7      x = x + Frozz(A, q + 1, r)
 8      x = x + Xyzzy(A, p, q)
 9      x = x + Xyzzy(A, q + 1, r)
10  return x
```

Før vi når grunntilfellet, har vi fire kall til Xyzzy med en instans av halve størrelsen, samt et konstant antall kall til Frozz. Det gir oss rekurrensen

$T(n) = 4T(n/2) + \Theta(n^2)$. Denne kan vi f.eks. løse med masterteoremet, som gir $T(n) = \Theta(n^2 \lg n)$.

8 Hva er sertifikater, og hvilken rolle spiller de i definisjonene til NP og co-NP?

Et sertifikat brukes av en verifikasjonsalgoritme sammen med en instans av et beslutningsproblem for å verifisere at svaret er «ja» (i NP) eller «nei» (i co-NP).

Her aksepteres flere forklaringer. Man kan godt ha en kort eller skissepreget definisjon, så lenge hovedpoenget med sertifikater, og hovedforskjellen på NP og co-NP (år det skal finnes sertifikater), kommer frem.

9 I en nettavis legges det ut nyhetssaker med ujevne mellomrom, og planen er at brukerne skal motta e-post om disse, der hver e-post kan inneholde flere nyhetssaker. Kriteriene for utsending er som følger:

- Det skal aldri gå mer enn k timer fra en nyhetssak publiseres til den inngår i en e-post; og
- Det skal sendes ut så få e-poster som mulig.

Hvordan vil du løse problemet hvis du på forhånd vet når sakene skal publiseres? (For enkelhets skyld kan du anta at du har en liste med tidspunkter for alle sakene du trenger bry deg om.) Hva om du ikke vet dette? Forklar hvorfor løsningen din blir riktig.

Svar relativt grundig (f.eks. ca. 100–200 ord).

Problemet løses ved grådighet: Dekk tidspunktene med intervaller med en lengde på k timer. Det lønner seg uansett ikke å starte et intervall før første nyhetssak (grådighetsegenskapen). Om du fjerner sakene som dekkes av et intervall, sitter du igjen med et ekvivalent problem (optimal delstruktur/induksjon).

Her vil trolig en fullverdig besvarelse være noe mer omfattende, men selv et såpass kortfattet svar som angitt over kan gi full uttelling, så lenge det får med de sentrale punktene.

Dette tilsvarer oppgave 16.2-5 fra læreboka (3. utgave), som også ble brukt som diskusjonsoppgave i forelesning 7.

10 Castingdirektør Gløgsund skal besette rollene i et sett med kortfilmer. Hun har et sett med kandidat-skuespillere, der hver skuespiller er aktuell for noen av rollene, men ikke alle. Hun ønsker ikke å overeksponere noen av skuespillerne, så hun vil bruke hver av dem i maksimalt 1/3 av filmene. Hver skuespiller kan maksimalt få én rolle per film.

Blant skuespillerne finnes det en del *rivaler*. Hvis A og B er rivaler, er enten bare én av dem aktuell for roller i en gitt film X, ellers så er A og B aktuelle for de samme rollene i film X. To rivaler kan ikke bli med i samme film.

Hvordan kan Gløgsund finne sin rollebesetning? Forklar og diskuter.

Du kan anta at det finnes en løsning der alle rollene besettes.

Bruk maks-flyt. Innfør noder for skuespillere, med kant fra kilden med kapasitet lik 1/3 av antall filmer, og noder for roller, med kant til sluket med kapasitet 1. Innfør også en node per skuespiller for hver film, med kant fra skuespiller til disse nodene, og fra disse nodene til aktuelle roller. For rivaler slås disse nodene sammen til én node per film.