TDT4120 Algoritmer og datastrukturer

Eksamen, 19. desember 2022, 15:00–19:00

Faglig kontakt	Magnus Lie Hetland
Hjelpemiddelkode	E

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!

1 Hva er kjøretiden til DIJKSTRA med en binærhaug som prioritetskø?

Oppgi svaret i O-notasjon. Du kan anta $|E| = \Omega(V)$.

> $O(E \lg V)$
>
> **Relevante læringsmål:** Forstå DIJKSTRA; kjenne kjøretiden under ulike omstendigheter, og forstå utregningen.

2 $Q = \langle 0,0,0,0,0,0,0,0,0,0 \rangle$ er en tabell brukt til å implementere en FIFO-kø. Utfør følgende prosedyre.

```
1  Q.head = 1
2  Q.tail = 2
3  ENQUEUE(Q, 1)
4  ENQUEUE(Q, 2)
5  Q.head = 9
6  Q.tail = 10
7  ENQUEUE(Q, 3)
8  ENQUEUE(Q, 4)
```

Hvordan ser Q ut etterpå?

> $Q = \langle 4,1,2,0,0,0,0,0,0,3 \rangle$
>
> Her kan man også få noe uttelling om man har byttet om på rollen til *head* og *tail*, og satt inn mot venstre, så man ender med $\langle 1,0,0,0,0,0,0,4,3,2 \rangle$.
>
> **Relevant læringsmål:** Forstå hvordan *køer* fungerer (inkl. operasjonene ENQUEUE og DEQUEUE).

3 Hvorfor er ikke memoisering nyttig når man bruker designmetoden *splitt og hersk* (*divide and conquer*)?

> Fordi man ikke har overlappende delproblemer.
>
> Dvs., hver delinstans løses maksimalt én gang, så det er ingen gevinst i å mellomlagre løsningen, som man gjør med memoisering.
>
> **Relevante læringsmål:** Forstå designmetoden *splitt og hersk*; forstå designmetoden *dynamisk programmering*; forstå hva *overlappende delinstanser* er; forstå løsning ved *memoisering*.

4 Hva brukes *kjeding* (*chaining*) til?

Du trenger ikke forklare hvordan det fungerer.

> Til å håndtere kollisjoner i hashtabeller.
>
> **Relevant læringsmål:** Forstå *konfliktløsing ved kjeding* (*chaining*).

5 Gi nedre og øvre asymptotiske grenser for uttrykket $n + \Theta(n^2) + O(n^3)$.

> $\Omega(n^2)$ og $O(n^3)$
>
> **Relevant læringsmål:** Kunne definere *asymptotisk notasjon*, O, Ω, Θ, o og ω.

6 Forenkle uttrykket $\Omega(n + \Theta(n^2) + O(n^3))$.

> $\Omega(n^2)$
>
> **Relevant læringsmål:** Kunne definere *asymptotisk notasjon*, O, Ω, Θ, o og ω.

7 Løs rekurrensen $T(n) = 4T(n/2) + n^2 \lg n$. Uttrykk svaret med Θ-notasjon.

> $\Theta(n^2 \lg^2 n)$
>
> Her kan man bruke tilfelle 2 av masterteoremet.
>
> **Relevant læringsmål:** Kunne løse rekurrenser med *substitusjon*, *rekursjonstrær*, *masterteoremet* og *iterasjonsmetoden*.

8 Start med et tomt binært søketre, og sett så inn følgende verdier, i rekkefølge, med TREE-INSERT:

$\langle 7, 1, 0, 5, 4, 8, 3, 2, 9, 6 \rangle$

Utfør deretter INORDER-TREE-WALK på rotnoden i det resulterende treet. Hva skriver algoritmen ut?

Du skal her kun svare med output fra algoritmen.

0 1 2 3 4 5 6 7 8 9

Her vil man naturligvis få rett svar om man eksplisitt utfører algoritmene som anvist, men om man har forstått at en *inorder*-traversering av et binært søketre alltid besøker nodene i sortert rekkefølge, kan man her også finne svaret direkte.

Relevant læringsmål: Forstå hvordan *binære søketrær* fungerer (inkl. operasjonene TREE-INSERT og INORDER-TREE-WALK).

9 Følgende matrise er vektmatrisen til en vektet, rettet graf:

$$W = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & \infty & 2 \\ 2 & 1 & 0 & 5 & 1 \\ 3 & 7 & \infty & 0 & 1 \\ 4 & 5 & 8 & 6 & 0 \end{array}$$

Utfør SLOW-APSP på grafen. Hva blir $l_{3,1}^{(2)}$?

6

Relevant læringsmål: Forstå SLOW-APSP; vite hvordan den oppfører seg; kunne utføre algoritmen, trinn for trinn.

10 Anta at du legger inn en sjekk i BELLMAN-FORD som avslutter algoritmen dersom ingen avstandsestimater endrer seg i løpet av en iterasjon. Hva blir da den totale kjøretiden, i beste tilfelle, om du antar at det finnes stier fra startnoden til alle andre? Forklar kort.

Selv om vi kan nå frem til alle nodene, kan vi ende med at alle får rett avstand etter én iterasjon, så kjøretiden i beste tilfelle blir $\Theta(V + E)$, som i dette tilfellet kan forenkles til $\Theta(E)$.

Relevant læringsmål: Forstå BELLMAN-FORD; kjenne kjøretiden under ulike omstendigheter, og forstå utregningen.

11 Hva er det minste og største antallet elementer i en binærhaug med høyde h?

2^h og $2^{h+1} - 1$

2^{h-1} og $2^h - 1$ gis også full uttelling.

Om haugen er full, er dette summen $1 + 2 + 4 + \cdots + 2^h$, som er $2^{h+1} - 1$. Det minste vi kan ha er én mer enn en full haug av høyde $h - 1$, som altså blir 2^h.

Om man her har brukt gal definisjon av høyde, og telt antall nivåer med noder i stedet for lengste sti fra rot til løvnoder, vil man få 2^{h-1} og $2^h - 1$. Siden poenget med oppgaven ikke var å teste bruk av riktig definisjon her, vil dette også gi full uttelling.

Dette er oppgave 6.1-1 fra læreboka.

Relevant læringsmål: Forstå hvordan *hauger* fungerer.

12 Hva sier heltallsteoremet (*the integrality theorem*)? Forklar kort med egne ord.

Hvis vi har heltallskapasiteter, vil flyten funnet av Ford-Fulkerson-metoden være heltallig.

Her kan man ev. også presisere at $f(u, v)$ er heltallig for hvert nodepar u, v, og at summen $|f|$ også er et heltall.

Relevant læringsmål: Forstå *heltallsteoremet*.

13 Hva er restkapasitet (*residual capacity*) og hvordan regner man det ut? Forklar kort.

Det er den gjenværende kapasiteten til en kant i et flytnett. For en kant (u, v) med flyt $f(u, v)$ er restkapasiteten $c_f(u, v)$ hvor mye som gjenstår, altså $c(u, v) - f(u, v)$, mens $c_f(v, u)$ er hvor mye vi kan oppheve, altså $f(u, v)$.

Det er altså snakk om kapasiteten i restnettet/residualnettverket. Strengt tatt har vi også kapasitet og restkapasitet mellom noder der det ikke finnes noen kant, men denne er alltid 0.

Relevant læringsmål: Kunne definere *restnettet* til et flytnett med en gitt flyt.

14 Din venn Smartnes mener at *grafisomorfi* er minst like vanskelig som *faktorisering*. For å etablere dette tenker hun å vise at en løsning på det ene problemet kan, med litt ekstra beregning, brukes til å løse det andre. Forklar hvilket problem sin løsning som i så fall må kunne brukes på det andre problemet, og hvorfor det fører til den ønskede konklusjonen.

Lurvik må vise at man kan bruke en tenkt løsning på grafisomorfiproblemet til å løse faktoriseringsproblemet, dvs., redusere fra faktorisering til grafisomorfi. Så lenge reduksjonen ikke innebærer mye ekstra arbeid, betyr det at hvis man kan løse grafisomorfi, så kan man løse faktorisering (f.eks. i polynomisk tid), men ikke nødvendigvis omvendt; altså vil man ha etablert at grafisomorfi er minst like vanskelig som faktorisering.

Om vi skriver G for grafisomorfi og F for faktorisering, og vi bruker reduksibilitetsreduksjonen $\leqslant_P$, betyr altså $F \leqslant_P G$ at F kan reduseres til G i polynomisk tid, og indikerer at G er minst like vanskelig som F (mtp. løsbarhet i polynomisk tid). Andre typer reduksjoner kan gi andre betydninger av «vanskelig»; det er det generelle poenget, og spesielt reduksjonsretningen, vi er ute etter her.

Relevant læringsmål: Forstå *redusibilitets-relasjonen* $\leqslant_P$.

Algoritme 1 Lurviks versjon av *randomized select*

```
RANDOMIZED-SELECT(A, p, r, i)
1  if r ⩽ p
2        return A[p]
3  q = RANDOMIZED-PARTITION(A, p, r)
4  k = q − p + 1
5  if i == k
6        return A[q]
7  elseif i < k
8        RANDOMIZED-SELECT(A, p, q − 1, i)
9        RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

15 Din venn Lurvik har prøvd å skrive ned pseudokode for *randomized select* etter hukommelsen. Resultatet (algoritme 1) er ikke helt rett. Beskriv hva som må fikses for at algoritmen skal bli korrekt.

Man må sette **return** foran kallene til RANDOMIZED-SELECT, og **else** først på linje 9. Altså:

```
8        return RANDOMIZED-SELECT(A, p, q − 1, i)
9  else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

Ev. kan man også korrigere betingelsen på linje 1:

```
1  if p == r
```

På linje 1 i versjonen i læreboka er sammenligningen $p == r$, ikke $r \leqslant p$, men det påvirker ikke oppførselen for den korrigerte algoritmen, siden vi da aldri får $r < p$. Om man også korrigerer dette, vil det ikke gi noe trekk.

Relevant læringsmål: Forstå RANDOMIZED-SELECT.

16 Hvilket problem løser algoritme 1, dersom den kalles som følger, der $A[1:n]$ er en tabell med tall?

RANDOMIZED-SELECT$(A, 1, n, 0)$

Forklar kort.

Den vil sortere A. Det ser vi fordi den da vil oppføre seg akkurat som RANDOMIZED-QUICKSORT.

Relevant læringsmål: Forstå RANDOMIZED-QUICKSORT.

17 Din venn Gløgsund har laget to versjoner av Ford–Fulkerson-metoden der hun bruker henholdsvis DIJKSTRA og TRANSITIVE-CLOSURE til å finne forøkende stier. Hvilke av disse to metodene vil garantert finne maks-flyt i polynomisk tid? Forklar kort.

Anta at $w(u, v) = 1$ for alle kanter (u, v) i restnettet, og at Gløgsund vedlikeholder en Π-tabell med forgjengere i TRANSITIVE-CLOSURE for å finne de faktiske stiene.

DIJKSTRA vil finne korteste forøkende sti, og vil dermed gi et polynomisk antall iterasjoner, akkurat som når vi bruker BFS i Edmonds–Karp.

TRANSITIVE-CLOSURE vil ikke nødvendigvis finne korteste forøkende stier, og vi risikerer å ende med et eksponentielt antall iterasjoner.

Man trenger ikke argumentere for at TRANSITIVE-CLOSURE faktisk kan velge stier som gir eksponentiell kjøretid. Det sentrale er at argumentet for polynomisk kjøretid for Edmonds–Karp bryter sammen.

Man kan også få nesten full uttelling om man ikke nevner TRANSITIVE-CLOSURE, om det kommer tydelig frem at man implisitt mener den vil gi galt svar fordi stiene den finner ikke nødvendigvis er kortest mulig.

Relevant læringsmål: Forstå FORD-FULKERSON; forstå BFS; forstå DIJKSTRA; forstå TRANSITIVE-CLOSURE.

18 Vi sier at en kvinne og en mann er *ment for hverandre* om de ender opp sammen i alle mulige stabile matchinger. Konstruer en effektiv algoritme som bestemmer om en kvinne og en mann er ment for hverandre.

Kjør både kvinne- og manns-orientert GALE-SHAPLEY, og se om noen matches i begge tilfeller. De er da hverandres beste og verste partner over alle stabile matchinger, og er dermed ment for hverandre.

Relevant læringsmål: Forstå GALE-SHAPLEY; kunne konstruere nye effektive algoritmer.

19 Hvordan kan vi løse delsumproblemet (*the subset-sum problem*) i polynomisk tid hvis den ønskede delsummen (*target*) er oppgitt i entallssystemet?

I entallssystemet representeres k som en streng $111 \cdots 1$ av lengde k.

Løses som det binære ryggsekkproblemet (*0-1 knapsack*), der vekt er lik verdi, og kapasiteten settes til den ønskede delsummen. Svaret er "ja" hvis og bare hvis vi får fylt opp ryggsekken helt.

Siden kjøretiden er $O(nW)$, og antall bits i input er $\Omega(nW)$, så er kjøretiden polynomisk.

Dette er oppgave 34.5-3 fra læreboka.

Relevant læringsmål: Forstå løsningen på *det binære ryggsekkproblemet*; forstå hvorfor løsningen på det binære ryggsekkproblemet *ikke er polynomisk*; forstå definisjonen av klassen P; kjenne det NP-komplette problemet SUBSET-SUM.

20 Et kongerike består av flere regioner. Kongen ønsker å bygge en mur som går rundt én eller flere av regionene, inkludert den som inneholder det kongelige slott. Byggekostnadene varierer med terrenget, og kongen har bedt deg om å finne den billigste løsningen. Hvordan vil du gå frem?

Du kan anta at muren følger regiongrenser.

Lag en graf med hver region, inkl. utlandet, som en node, og med en kant over hver grense, med kapasitet lik kostnaden ved å bygge langs grensen. Finn så et minimalt snitt mellom slottet og utlandet, ved hjelp av Ford-Fulkerson (spesifikt, Edmonds–Karp-algoritmen).

Her kan man forstå oppgavebeskrivelsen på litt ulike måter. F.eks. er det ikke sikkert man ser på det som et krav at det skal være én ringmur, men at området den omslutter kan deles opp av mindre murer, etc. Men siden man skal gjøre det billigst mulig, vil den billigste løsningen likevel være å unngå dette.

Man kan også tolke det som at man har oppgitt én eller flere regioner som skal omsluttes. Siden det er snakk om én mur, må det likevel løses på en måte som minner om den i løsningsforslaget, bare at man kan innføre en superkilde, og koble til alle disse regionene med kanter som har uendelig høy kapasitet (så alle havner på kilde-siden av snittet).

Om man her heller bruker en graf der grensene er kanter (dvs., den duale grafen av det som brukes i løsningen over), og prøver å finne den korteste sykelen rundt slottet, kan det også gi uttelling, selv om det kan være utfordrende å finne en algoritme som gir riktig svar, dvs., å ikke bare finne den billigste sykelen, men den billigste som omslutter slottet.

Relevant læringsmål: Forstå Ford-Fulkerson; forstå *maks-flyt/min-snitt-teoremet*; kunne konstruere nye effektive algoritmer.