TDT4120 Algoritmer og datastrukturer

Eksamen, 10. august 2023, 09:00–13:00

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!

5 % **1** Algoritme 1 sorterer tabellen $A[1:n]$. Hva er kjøretiden, som funksjon av n?

Oppgi svaret i asymptotisk notasjon.

$\Theta(n^2)$

Algoritmen er hentet fra `https://arxiv.org/abs/2110.01111`.

Relevant læringsmål: Kunne analysere algoritmers effektivitet; kunne definere *asymptotisk notasjon* (Θ).

5 % **2** Hva er et spenntre?

Det trenger ikke være minimalt.

Det er et tre som består av kanter fra en gitt graf, og som inneholder alle nodene i grafen.

Her tillates mange ulike forklaringer, så lenge man får frem poenget. Om man kun oppgir at det er et tre som kobler sammen nodene, uten å oppgi at kantene i treet hentes fra grafen, gir dette 4 poeng.

Algoritme 1

```
FUNG-SORT(A, n)
1  for i = 1 to n
2      for j = 1 to n
3          if A[i] < A[j]
4              swap A[i] and A[j]
```

Relevant læringsmål: Vite hva *spenntrær* og *minimale spenntrær* er.

5 % **3** Hvilket problem løser Floyd-Warshall?

Her er vi ikke ute etter bare navnet på problemet, men en svært kort beskrivelse av hva problemet er.

Korteste veier, fra alle noder til alle andre, i en vektet, rettet graf.

Her kan man godt bruke frasen «fra alle til alle», uten å nevne noder. Man får 4 poeng om man utelater å nevne vekting eller retning, eller begge deler.

Det er korrekt, men ikke påkrevd, å også nevne kravet om at grafen ikke har negative sykler. Det er også korrekt, men ikke direkte relevant, å nevne at algoritmen kan modifiseres til å finne transitiv lukning.

Relevant læringsmål: Forstå Floyd-Warshall (kjenne den formelle definisjonen av det generelle problemet den løser).

5 % **4** Dijkstra velger en node i hver iterasjon. Hvilken?

Den med lavest avstandsestimat, *v.d.*

Teknisk sett *en av* dem med lavest avstandsestimat, og hvilken av dem som velges er en implementasjonsdetalj. Det er selvfølgelig også korrekt å spesifisere at det er en av de *gjenværende* nodene som velges.

Relevant læringsmål: Forstå Dijkstra (vite hvordan den oppfører seg; kunne utføre algoritmen, trinn for trinn).

5 % **5** Tellesortering (*counting sort*) har bedre kjøretid enn f.eks. flettesortering (*merge sort*). Hva er det vi krever av input til tellesortering som gjør dette mulig?

Elementene i input-tabellen må være heltall i et lite verdiområde.

Her er det også akseptabelt om man oppgir at verdiområdet må ha konstant størrelse eller at størrelsen k er $O(n)$. Det er også korrekt å si at $k = o(n \lg n)$ vil gjøre tellesortering raskere.

Relevant læringsmål: Forstå Counting-Sort (kjenne til eventuelle tilleggskrav den stiller for å være korrekt; kjenne til eventuelle styrker eller svakheter, sammenlignet med andre).

5 % **6** Hva er konsekvensen av å finne en polynomisk algoritme for et problem i NPC?

> P = NP, dvs., alle problemer i NP kan løses i polynomisk tid.
>
> **Relevante læringsmål:** Forstå definisjonen av klassene P og NP; forstå definisjonen av NP-*kompletthet*.

5 % **7** Dine venner Lurvik og Smartnes har laget hver sin sorteringsalgoritme, som rett og slett utfører to andre sorteringsalgoritmer etter hverandre:

```
LURVIK-SORT(A, n)
1 INSERTION-SORT(A, n)
2 MERGE-SORT(A, 1, n)
```

```
SMARTNES-SORT(A, n)
1 MERGE-SORT(A, 1, n)
2 INSERTION-SORT(A, n)
```

Hvilken av dem har best kjøretid i verste tilfelle? Forklar kort.

> SMARTNES-SORT blir best, siden MERGE-SORT uansett har kjøretid $\Theta(n \lg n)$, og når den kjøres først, får INSERTION-SORT kjøretid $\Theta(n)$, totalt $\Theta(n \lg n)$. Om INSERTION-SORT kjøres først, risikerer vi $\Theta(n^2) + \Theta(n \lg n) = \Theta(n^2)$.
>
> En kortere forklaring kan her gi full uttelling.
>
> **Relevante læringsmål:** Forstå INSERTION-SORT og MERGE-SORT (kjenne kjøretidene under ulike omstendigheter, og forstå utregningen).

5 % **8** Lurvik og Smartnes skal på togferie. Det går direktetog mellom mange av byene de skal besøke, og Lurvik vil finne en rute som går innom hver by nøyaktig én gang, om mulig. Smartnes mener det er urealistisk. Hva mener du?

Det er her snakk om å lage en effektiv algoritme for å løse problemet generelt.

> En slik algoritme ville kunne løse HAM-CYCLE-problemet, så det er neppe realistisk.
>
> Forklaringer om at en polynomisk algoritme vil medføre at P = NP, og lignende, eller svar som påpeker at problemet er NP-hardt/NP-komplett er også korrekte. Man bør spesifikt nevne hamiltonsykelproblemet (eller ev. redusere fra et annet NP-hardt problem) for å få full uttelling. Om forklaringen innebærer en reduksjon *til* et annet NP-hardt problem, vil det gi lite eller ingen uttelling.
>
> **Relevante læringsmål:** Kjenne det NP-komplette problemet HAM-CYCLE (kunne angi presist hva input er; kunne angi presist hva output er og hvilke egenskaper det må ha); være i stand til å konstruere enkle NP-kompletthetsbevis.

5 % **9** Du skal finne et passord som består av n tegn fra et alfabet av størrelse k. Du prøver ett og ett passord (*brute force*). Hvor mange passord må du prøve før du finner det rette? Oppgi svaret i asymptotisk notasjon.

Du kan anta at du kjenner både n og k.

> $O(k^n)$
>
> Her får man også full uttelling om man skriver $\Theta(k^n)$, selv om det ikke er helt riktig.
>
> **Relevante læringsmål:** Kunne analysere algoritmers effektivitet; kunne definere *asymptotisk notasjon* (O).

5 % **10** Tre menn (Lurvik, Smartnes og Visdal) og tre kvinner (Gløgsund, Klokland og Flinckenhagen) har følgende preferanser:

Lurvik:	Gløgsund, Flinckenhagen, Klokland
Smartnes:	Gløgsund, Klokland, Flinckenhagen
Visdal:	Klokland, Flinckenhagen, Gløgsund
Gløgsund:	Lurvik, Smartnes, Visdal
Klokland:	Visdal, Smartnes, Lurvik
Flinckenhagen:	Lurvik, Smartnes, Visdal

Lurvik er matchet med Flinckenhagen, Smartnes er matchet med Gløgsund og Visdal er matchet med Klokland. Er matchingen stabil, eller finnes det et blokkerende par (*blocking pair*)? Hvem er det, i så fall? Forklar kort.

> Lurvik og Gløgsund utgjør et blokkerende par, siden de heller vil ha hverandre enn sine respektive partnere.
>
> **Relevante læringsmål:** Forstå hva en stabil matching (*stable matching*) er.

5 % **11** Det følgende er hentet fra Counting-Sort:

```
11  for j = 1 downto 1
12      B[C[A[j]] = A[j]
13      C[A[j]] = C[A[j]] − 1
```

Hva skal den sensurerte biten være?

> Se løsning i pseudokoden over.
>
> **Relevant læringsmål:** Forstå Counting-Sort (vite hvordan den oppfører seg; kunne utføre algoritmen, trinn for trinn).

5 % **12** Løs følgende rekurrens:

$$T(n) = 2T(n/2) + n/\lg n$$

Oppgi svaret med asymptotisk notasjon.

> $\Theta(n \lg \lg n)$
>
> Denne rekurrensen er diskutert på s. 105–106 i læreboka, der løsningen også er oppgitt. Den kan også løses med iterasjonsmetoden og substitusjonsmetoden, men det kan være utfordrende.
>
> Oppgaven var egentlig ment å løses med masterteoremet, men faller utenfor den varianten av teoremet som brukes i pensum. Derfor tas oppgaven ut av sensur der det er til fordel for kandidaten.
>
> Her får man 4 poeng for $\Theta(n)$, som er resultatet man får ved å bruke tilfelle 2 av masterteoremet, altså $f(n) = \Theta(n^{\log_b a} \lg^k n)$, for $a = b = 2, k = -1$. Dette svaret er *ikke korrekt*, siden dette tilfellet krever $k > 0$.

5 % **13** Tabellen $A = \langle 9, 8, 5, 7, 1, 3, 2, 4, 6 \rangle$ representerer en haug. Hvordan ser tabellen ut etter første iterasjon av HEAPSORT?

Du skal altså utføre den første av $n - 1$ iterasjoner. Svar ved å liste opp elementene i tabellen. Oppgi hele tabellen, inkludert deler som ikke lenger er en del av haugen.

> $8, 7, 5, 6, 1, 3, 2, 4, 9$
>
> **Relevant læringsmål:** Forstå HEAPSORT (vite hvordan den oppfører seg; kunne utføre algoritmen, trinn for trinn).

5 % **14** Flytnett (*flow networks*) kan defineres på litt forskjellige vis, men i versjonen i pensum tillates ikke antiparallelle kanter (dvs., at man både har en kant fra u til v og en kant fra v til u). Hvor stor begrensning er dette? Forklar kort.

> Det er ikke noen egentlig begrensning. Om vi har et nettverk som har antiparallelle kanter, kan vi splitte den ene av hvert par med en ny node, og få et ekvivalent nettverk som ikke har det.
>
> **Relevant læringsmål:** Kunne håndtere *antiparallelle kanter*.

5 % **15** Algoritme 2 finner antall mulige permutasjoner av elementene i mengden S, rekursivt. Hva taler for og imot bruk av memoisering for å optimere den?

Du kan f.eks. bruke en hashtabell med mengder som nøkler.

Algoritme 2

```
PERMUTATIONS(S)
1  if S == ∅
2      return 1
3  else n = 0
4      for each element x ∈ S
5          n = n + PERMUTATIONS(S − {x})
6      return n
```

Her kan mange svar gi uttelling, men hovedargumentet *for* er at man har overlappende delproblemer og et hovedargument *mot* er at memo-tabellen blir eksponentielt stor. Man kan naturligvis også oppgi som argument at det finnes atskillig mer effektive løsninger på problemet.

Relevante læringsmål: Forstå designmetoden *dynamisk programmering*; forstå løsning ved *memoisering*; forstå hva *overlappende delinstanser* er; kunne analysere algoritmers effektivitet.

5 % **16** Beskriv hvordan du kan bruke rekursjon til å finne avstanden fra startnoden s til en gitt node v i en vektet, rettet graf.

Merk: Det er forventet at løsningen vil ha eksponentiell kjøretid.

Her kan du svare svært kort. Det kreves ingen grundig pseudokode e.l. Du kan anta at det finnes en sti fra s til enhver annen node i grafen.

For hver inn-nabo u, finn avstanden rekursivt og legg til vekten $w(u, v)$. Velg så det minste av svarene.

Relevant læringsmål: Forstå strukturen til *korteste-vei*-problemet.

5 % **17** Et byggefirma har flere store oppdrag og skal fordele sine ansatte på disse. Hvert prosjekt har et sett med roller (tømrer, elektriker, rørlegger, etc.) og et antall som trengs av hver av disse. Hver ansatt er kompetent til å fylle én eller flere slike roller, men kan maksimalt delta i ett prosjekt, og fyller da nøyaktig én rolle. For å holde reiseavstandene nede kan hver ansatt bare bli tilordnet et prosjekt innenfor en gitt avstand fra hjemstedet.

Hvordan ville du ha funnet en gyldig fordeling?

Kan løses som et flytproblem, med ansatte, roller (per prosjekt) og prosjekter som noder. Kanter fra kilde til ansatte (kapasitet 1), fra ansatte til roller de kan inneha i prosjekter de kan delta i (kapasitet 1), fra roller til tilhørende prosjekter (kapasitet lik antall som trengs) og fra prosjekter til sluk (f.eks. ubegrenset kapasitet).

Relevante læringsmål: Være i stand til å konstruere reduksjoner til maks-flyt-problemet; forstå *heltallsteoremet* (*integrality theorem*).

5 % **18** I beviset for at CIRCUIT-SAT er NP-komplett konstrueres en logisk krets som simulerer en datamaskin som utfører en verifikasjonsalgoritme. Hva er input for denne kretsen?

Her trenger du ikke beskrive konstante «inputs», bare dem man står fritt til å settet til 0 eller 1 for å løse oppfyllbarhetsproblemet. Du kan svare svært kort.

Sertifikatet.

Poenget er at vi vil avgjøre om det eksisterer et slikt sertifikat y som oppfyller kretsen, og dermed altså gir $\mathrm{A}(x, y) = 1$.

Relevant læringsmål: Forstå beviset for at CIRCUIT-SAT er NP-komplett.

5 % **19** Din venn Gløgsund har klart å slette alle mellomrom og all tegnsetting i en avhandling hun skriver på, og hun vil ha din hjelp til å splitte teksten opp i enkelt-ord. Det du har å hjelpe deg med er en liste med gyldige ord, og en oversikt over ord som aldri forekommer ved siden av hverandre. Beskriv en algoritme som løser problemet.

Det kan være flere gyldige løsninger. I så fall holder det at du finner én av dem.

Prøv hvert mulige ord i starten, og løs resten rekursivt med memoisering. Hopp over delinstanser (rekursive kall) som starter med ord som ikke kan stå ved siden av det første.

Problemet ligner på stavkappingsproblemet, det er noen vesentlige forskjeller, så om man henviser til dette, så må man også forklare hvordan løsningen må modifiseres.

Relevante læringsmål: Kunne konstruere nye effektive algoritmer; forstå designmetoden dynamisk programmering; forstå eksemplet *stavkapping*.

5 % **20** Anta at du har en prosedyre A som avgjør beslutningsproblemet VERTEX-COVER i konstant tid. Beskriv hvordan du kan bruke A til å finne et minst mulig nodedekke. Løsningen din skal ha så lav asymptotisk kjøretid som mulig. Gitt denne kjøretiden, skal den bruke så få kall til A som mulig. (Du skal altså ikke øke den asymptotiske kjøretiden bare for å redusere antall kall til A.)

Merk at du her faktisk skal finne nodedekket, ikke bare størrelsen.

Bruk binærsøk i verdiområdet $1 \ldots |V|$ for å finne størrelsen k til minste nodedekke.

Fjern så en node v og sjekk om resten har et dekke av størrelse $k - 1$. I så fall tas v med i løsningen; ellers forkastes den. Fortsett på samme måte.

Om vi antar at nodeslettingen også sletter tilstøtende kanter, får vi kjøretid $\Theta(V + E)$ i verste tilfelle. Antall kall til A blir $(\lfloor \lg |V| \rfloor + 1) + (n - 1) = \lfloor \lg |V| \rfloor + n$. Merk at det ikke spørres etter kjøretid eller antall kall, så det er ikke nødvendig å oppgi dette i svaret.

Relevante læringsmål: Kunne konstruere nye effektive algoritmer; kjenne det NP-komplette problemet VERTEX-COVER (kunne angi presist hva input er; kunne angi presist hva output er og hvilke egenskaper det må ha).