Løsningsforslag, Eksamen i

TDT4120 Algoritmer og datastrukturer

August 2005

Faglige kontakter under eksamen:
Magnus Lie Hetland, Arne Halaas
Tillatte hjelpemidler: Bestemt enkel kalkulator. Ingen trykte eller håndskrevne hjelpemidler.

Merk: Svarene skal skrives på angitt sted på oppgavearket.

Oppgave 1 (80%)

De følgende deloppgavene skal besvares med "ja" eller "nei" (sett kryss ved det som passer). Dersom deloppgaven er et utsagn (og ikke et spørsmål) betyr "ja" at utsagnet er sant og "nei" at utsagnet er usant eller ikke stemmer (evt. ikke gir mening).

En riktig besvart deloppgave gir 4 poeng, en ubesvart deloppgave gir 0 poeng og en galt besvart deloppgave gir –4 poeng. En negativ totalsum for hele denne oppgaven teller som 0 poeng når totalen for eksamen beregnes.

Les oppgavene nøye. Oppgavetekstene er utformet slik at svarene ikke skal være opplagte. Svar bare dersom du er sikker på at du har forstått oppgaven og at du vet svaret.

---

[x] Ja [x] Nei *a*) Uansett form på et binært tre med $n \geq 1$ interne noder vil antall løvnoder være $n + 1$.

Dette spørsmålet har vist seg å være for upresist. Hvis interne noder kan ha 1 eller 2 barn (mens løvnoder har 0 barn) er utsagnet galt. Dette er i tråd med definisjonen i Levitin. Hvis interne noder alltid har 2 barn (som i et søketre der alle interne noder har nøkler) vil utsagnet være riktig. Det gis derfor full score uansett hva man har svart (og også om spørsmålet er ubesvart).

[ ] Ja [x] Nei *b*) Grådighet er en generalisering av dynamisk programmering.

---

[x] Ja [ ] Nei *c*) Warshalls algoritme kan brukes når man har kanter med negativ vekt i grafen.

Warshalls algoritme finner den transitive tillukningen av naboskapsrelasjonen i grafen, og er ikke avhengig av eventuelle kantvekter. (Floyds algoritme, også ofte kjent som Floyd-Warshall, som finner korteste vei mellom alle par med noder, gjør heller ingen

antagelser om positive kanter, i motsetning til Dijkstras algoritme. Hvis den finner at en node har negativ avstand til seg selv så betyr det at grafen inneholder en negativ sykel.)

—

Du har oppgitt en graf $G = (V, E)$ og ønsker å finne det største settet $V' \subseteq V$ der alle nodene i V' har kanter til alle de andre andre nodene i V'.

[ ] Ja [x] Nei *d*) Problemet kan effektivt løses med dynamisk programmering.

Dette er Max-Clique-problemet, som er NP-komplett. Under antagelsen P≠NP (som forutsettes) kan ikke problemet løses effektivt (det vil si i polynomisk tid).

—

[x] Ja [ ] Nei *e*) Det er umulig å lage en algoritme for å sjekke om (vilkårlige) andre algoritmer er korrekte.

Dette er en variant (generalisering) av Halting-problemet, som er umulig å løse.

—

Vi har gitt en graf $G = (V, E)$, der $V = \{1, \ldots, 7\}$ og $E = \{(1, 2), (1, 4), (2, 4), (2, 5), (3, 1), (3, 6), (4, 3), (4, 5), (4, 6), (4, 7), (5, 7), (7, 6)\}$.

Vi har også gitt kantvektfunksjonen $w(u, v)$, gitt ved
$w(1, 2) = 2$, $w(1, 4) = 1$, $w(2, 4) = 3$, $w(2, 5) = 10$, $w(3, 1) = 4$, $w(3, 6) = 5$, $w(4, 3) = 2$, $w(4, 5) = 2$, $w(4, 6) = 8$, $w(4, 7) = 4$, $w(5, 7) = 6$, $w(7, 6) = 1$.

Simuler (utfør) Dijkstras algoritme på denne grafen, med 3 som start-node. Før første iterasjon er det kun node 3 som har riktig avstandsestimat. Hvis du kan velge mellom to noder, velg alltid den med lavest nummer.

Fremdriften til algoritmen blir slik (etter de angitte iterasjonene):

Iterasjon	**d[1]**	**d[2]**	**d[3]**	**d[4]**	**d[5]**	**d[6]**	**d[7]**
—	∞	∞	**0**	∞	∞	∞	∞
1	**4**	∞	**0**	∞	∞	5	∞
2	**4**	6	**0**	**5**	∞	5	∞
3	**4**	6	**0**	**5**	7	**5**	9
4	**4**	**6**	**0**	**5**	7	**5**	9
5	**4**	**6**	**0**	**5**	**7**	**5**	9
…							

[x] Ja [ ] Nei *f*) Node 4 har riktig avstandsestimat etter andre iterasjon.

[x] Ja [ ] Nei *g*) Node 5 har riktig avstandsestimat etter fjerde iterasjon.

[ ] Ja [x] Nei *h*) Nodene i grafen *G* kan ordnes topologisk i $O(|E|)$ tid.

Det kan se ut til at spørsmålet er meningsløst (asymptotisk kjøretid for en gitt probleminstans), men poenget er at grafen inneholder en sykel, så nodene kan ikke ordnes topologisk i det hele tatt.

I den følgende deloppgaven, se bort fra kantretningene (det vil si, se på *G* som en urettet graf). La *T* være et minimalt spenntre for grafen *G*, beregnet med Prims algoritme med node 1 som startnode. Hvis du kan velge mellom to noder, velg vilkårlig.

[ ] Ja [x] Nei *i*) Kanten (2, 4) er med i *T*.

Først vil kantene (1, 4), (1, 2) og (4, 5) legges til, med de siste to i vilkårlig rekkefølge. På dette tidspunktet ville kanten (2, 4) lage en sykel.

—

Du har en hashfunksjon definert ved $h(k) = (7k + 3) \bmod 5$, der *k* er en heltallsnøkkel og mod er *modulus*- eller restoperatoren. For eksempel vil $h(1) = 0$. Du har også en hashtabell *A* med 5 plasser, $A[0], \ldots, A[4]$. Du bruker lukket hashing (*linear probing*).

[ ] Ja [x] Nei *j*) Etter å ha satt inn tallene 11, 13, 15 og 21 kan 14 settes inn uten kollisjon.

Ingen av de foregående nøklene hasher direkte til posisjon $h(14) = 1$, men både nøkkel 11 og 21 hasher til posisjon 0, og på grunn av den lukkede hashingen havner nøkkel 21 da i posisjon 1, så 14 kolliderer med 21.

—

Betrakt rekurrensen $T(n) = 2T(n/2) + n$, $T(1) = 1$.

[ ] Ja [x] Nei *k*) Rekurrensen er typisk for algoritmer basert på dynamisk programmering.

Dette er en typisk rekurrens for algoritmer basert på splitt-og-hersk-tankegang. Et typisk eksempel er flettesortering. Om man løser rekurrensen får man $T(n) = \Theta(n \log n)$.

[ ] Ja [x] Nei *l*) $T(\mathrm{n}) \in \Omega(n^2)$

[x] Ja [ ] Nei *m*) $T(\mathrm{n}) \in O(n^2)$

—

I enkelte sanger (som “Gubben og gamla” og “The Twelve Days of Christmas”) øker lengden på versene med $\Theta(1)$ for hvert vers. Anta at antall vers er *n*.

[x] Ja [ ] Nei *n*) Den totale sanglengden er $\Omega(n^{1.7})$.

Lengden er kvadratisk (summen av en aritmetisk rekke, $\Theta(1 + 2 + \ldots + n)$). Mer presist: Kall verselengden etter *n* vers $V(n)$. Vi får da (fra oppgaveteksten) følgende rekurrens: $V(n) = V(n–1) + \Theta(1)$. Vi ser lett at $V(n) \in \Theta(n)$. Hvis vi kaller sanglengden $L(n)$ får vi altså følgende rekurrens: $L(n) = L(n–1) + V(n) = L(n–1) + \Theta(n)$. Løser vi denne rekurrensen får vi $L(n) \in \Theta(n^2) \subseteq \Omega(n^2) \subseteq \Omega(n^{1.7})$.

—

En algoritme tar et vilkårlig heltall n som input og kjøretiden er $\Omega(n^{1.7})$.

[x] Ja [ ] Nei *o*) Algoritmen har (nødvendigvis) eksponensiell kjøretid, som funksjon av problemstørrelsen.

Problemstørrelsen er plassen problemet tar (antall bits). Problemstørrelsen er altså her $m = \log n$, og kjøretiden blir $\Omega(2^{1.7m})$.

—

[x] Ja [x] Nei *p*) Hvis en kant e er den billigste/letteste kanten i en graf så finnes det et minimalt spenntre som inneholder e.

Siden det ikke ble presisert at det var snakk om en sammenhengende graf ble dette spørsmålet litt uheldig. Man kan mene at det er opplagt at det er snakk om en sammenhengende graf, og svaret blir da ja (nødvendig for korrekthet av f.eks. Kruskals algoritme). Gjør man derimot ikke denne antagelsen blir svaret nei, siden en usammenhengende graf ikke har noe spenntre. Oppgaven vil derfor ikke telle ved sensur (alle får full uttelling for oppgaven, enten den er besvart eller ikke.)

—

[ ] Ja [x] Nei *q*) Et minimalt spenntre er unikt hvis og bare hvis alle kantvektene er forskjellige.

Et enkelt moteksempel er et tre der kantvektene er like. Spenntreet vil likevel være unikt.

—

[x] Ja [ ] Nei *r*) Sortering ved sammenligning kan ikke løses med bedre kjøretid enn $\Omega(n \log n)$ i verste tilfelle.

—

[ ] Ja [x] Nei *s*) Ingen av sorteringsalgoritmene i pensum har kjøretid $O(n)$ i verste tilfelle.

Tellesortering har kjøretid $O(n)$ i verste tilfelle.

—

Du har n forskjellige verdigjenstander, hver med en vilkårlige positiv heltallsverdi (pris), og ønsker å finne ut om det er mulig å fordele gjenstandene likt på to av dine venner (så begge vennene mottar samme totale verdi).

[x] Ja [ ] Nei *t*) Det finnes ingen kjent metode som kan avgjøre dette med en kjøretid bedre enn $O(n^2)$.

Dette er ett av de NP-komplette problemene i pensum, *the partition problem*. Det finnes ingen kjente metoder som kan løse slike problemer i polynomisk tid.

Oppgave 2 (10%)

Det er NP-komplett å finne den lengste stien i en graf. Anta at nodene i grafen $G = (E, V)$ er sanger og at kanten (u, v) finnes i E dersom sang v er en nyinnspilling ("cover") av sang u. Anta at nodene i grafen $G' = (E', V')$ er artister og at kanten (u', v') finnes i E' dersom artist v' har gjort en nyinnspilling av en av artist u' sine sanger. Anta at du kan ha vilkårlig mange sanger og artister.

a) (5%) Enten (1) argumenter for at det er NP-komplett å finne den lengste stien i G, eller (2) beskriv kort en effektiv algoritme som finner den lengste stien i G (angi kjøretid).

Svar:

Under antagelsen P ≠ NP (som forutsettes) vil kun ett av alternativene kunne velges. (Studenter som eventuelt klarer å vise at problemet er NP-komplett *og* å løse det effektivt vil naturligvis automatisk få karakteren A.) Om studenten finner en polynomisk algoritme og argumenterer for at problemet derfor ikke er NP-komplett skal ikke dette trekke ned (selv om det formelt sett ikke er korrekt).

Problemet kan løses effektivt. Hvordan man gjør dette kommer an på et sentralt spørsmål: Kan man ha en nyinnspilling av en nyinnspilling? Eller vil denne nyinnspillingen automatisk være en nyinnspilling av originalen? Siden dette ikke var spesifisert, vil begge varianter godtas.

Variant 1: Man kan kun ha nyinnspillinger av originalsanger

Her vil de lengste stiene ha lengde 1 (hvis man antar en rettet graf og rettede stier—ellers vil lengste sti kunne bestå av to kanter), og man trenger kun sjekke om grafen har noen kanter i det hele tatt (eventuelt lete etter noder med to naboer).
Kjøretiden blir $\Theta(1)$ (eventuelt $\Theta(V)$).

For å få full uttelling for variant 1 må man eksplisitt ha forklart hvorfor løsningen er riktig (dvs., at man ikke kan ha en nyinnspilling av en nyinnspilling).

Variant 2: Man kan ha nyinnspillinger av nyinnspillinger

Uansett vil nyinnspillinger komme senere i tid, så man kan ikke ha sykler. Med kun denne observasjonen (G er en DAG) vil man kunne finne den lengste stien ved dynamisk programmering (den lengste stien til en node er kun avhengig av de lengste stiene til forgjengerne). Ved å legge til observasjonen at hver cover kun kan ha *en* forgjenger, ser man at grafen er en skog (en samling trær), og man kan for eksempel bruke BFS eller DFS.
Kjøretiden blir $\Theta(V + E)$.

b) (5%) Enten (1) argumenter for at det er NP-komplett å finne den lengste stien i *G*', eller (2) beskriv kort en effektiv algoritme som finner den lengste stien i *G*' (angi kjøretid).

Under antagelsen P ≠ NP (som forutsettes) vil kun ett av alternativene kunne velges. (Studenter som eventuelt klarer å vise at problemet er NP-komplett *og* å løse det effektivt vil naturligvis automatisk få karakteren A.)

Grafen G' er ikke nødvendigvis asyklisk. Den kan være en vilkårlig rettet graf, og kan derfor representere en vilkårlig (urettet) graf. Problemet er altså NP-komplett (fordi det er ekvivalent med det NP-komplette problemet som er beskrevet i oppgaveteksten).

Oppgave 3 (10%)

I det følgende, anta at $G = (V, E)$ er en urettet, asyklisk graf der alle kantene i E har vekt 1. Hvis du finner de korteste avstandene fra en node v i V til alle de andre nodene i V så er *eksentrisiteten* til v den *lengste* av disse avstandene.

a) (4%) Beskriv kort en effektiv algoritme som finner en eller flere noder med minimal eksentrisitet i grafen G. Oppgi kjøretid i verste tilfelle (så presist som mulig) med asymptotisk notasjon. Begrunn svaret.

En urettet, asyklisk graf er det samme som en skog. Hvis grafen ikke er sammenhengende vil alle noder ha uendelig eksentrisitet, så vi antar at grafen er sammenhengende, det vil si et tre.

Så lenge det finnes interne noder i treet vil ikke løvnodene kunne ha minimal eksentrisitet. Hvis man gjentatte ganger fjerner alle løvnodene helt til man ikke lenger har interne noder vil man ha funnet svaret.

Å finne løvnodene i første iterasjon vil ta $\Theta(n)$ tid (med n noder). Mengden med løvnoder kan så vedlikeholdes uten ekstra asymptotisk kostnad når gamle løvnoder fjernes og nye oppstår.

Siden man maksimalt kan fjerne $\Theta(n)$ noder blir kjøretiden $\Theta(n)$.

b) (3%) Hvor mange noder kan inngå i svaret til algoritmen i *a*)? Gi en kort begrunnelse.

Maksimalt to noder. Tre eller flere noder kan ikke ha samme eksentrisitet i et tre.

c) (3%) Beskriv kort et praktisk eksempel der en slik algoritme kan være av interesse.

Her er det åpent for kreative forslag. Det viktigste er at man gir uttrykk for å ha forstått problemstillingen helt presist. Man må altså ha forstått at grafen er et tre (eventuelt, i spesielle tilfeller en usammenhengende skog) og at man er ute etter noden(e) som ligger i "sentrum". Et eksempel kan for eksempel være å finne det beste stedet å legge et lager i et distribusjonsnettverk (der nettverket er uten sykler).