Avsluttende eksamen i TDT4120 Algoritmer og datastrukturer

Eksamensdato	13. august 2012
Eksamenstid	0900–1300
Sensurdato	3. september
Språk/målform	Bokmål
Kontakt under eksamen	Magnus Lie Hetland (tlf. 91851949)
Tillatte hjelpemidler	Ingen trykte/håndskrevne; bestemt, enkel kalkulator

Les alle oppgavene før du begynner, disponer tiden og forbered spørsmål til faglærer ankommer lokalet. Gjør antagelser der det er nødvendig. Skriv kort og konsist på angitt sted. Lange forklaringer og utledninger som ikke direkte besvarer oppgaven tillegges liten eller ingen vekt.

Algoritmer kan beskrives med tekst, pseudokode eller programkode, etter eget ønske, så lenge det klart fremgår hvordan den beskrevne algoritmen fungerer. Korte, abstrakte forklaringer kan være vel så gode som utførlig pseudokode, så lenge de er presise nok. Kjøretider oppgis med asymptotisk notasjon, så presist som mulig.

a) Anta at du har en DAG G med n noder og m kanter. Hva er kjøretiden for å finne en topologisk sortering av G?

Svar (6%): $\Theta(n + m)$

Anta at den urettede grafen $G = (V, E)$ er sammenhengende. Anta at hver node v i V får tilordnet en av sine nabokanter $e(v)$ i E, og at alle disse er forskjellige ($e(u) \neq e(v)$ hvis $u \neq v$).

b) Forklar kort hvorfor det ikke er sikkert at nodene i V, sammen med sine tilordnede kanter, danner et spenntre for G.

Svar (6%): Siden vi har n kanter vil det garantert oppstå sykler. (Det vil også gi full uttelling å si at det *kan* oppstå sykler. Det vil også gi noe uttelling å svare at den nye grafen ikke nødvendigvis vil være sammenhengende.)

Funksjonen F er definert ved $F(n) = (F(n–1) + F(n))/2 + 1$, der $F(0) = 0$.

c) Hva blir $F(42)$?

Svar (6%): 84 (Gjør om til $F(n) = F(n–1) + 2$, som gir $F(n) = 2n$.)

d) Hva er kjøretiden til HEAPSORT i verste tilfelle?

Svar (6%): $\Theta(n \lg n)$

e) Hva slags kø brukes vanligvis i Dijkstras algoritme?

Svar (6%): En binærheap. (Evt. prioritetskø, implementert med binærheap.)

f) Hva er *worst-case*-kjøretiden (som funksjon av n) for å sette inn n elementer i en hashtabell som alt inneholder $n^{0.5}$ elementer?

Svar (6%): $\Theta(n^2)$. (I verste tilfelle får vi kun kollisjoner, som gir $\Theta(n \cdot n^{0.5} + n \cdot n) = \Theta(n^2)$.)

g) Hvordan vil du finne antall mulige stier fra én node til en annen, hvis stiene ikke kan dele kanter?

Svar (6%): Maks-flyt med kapasitet 1.

h) Du kan redusere fra A til B i polynomisk tid. Svarene til både A og B kan verifiseres i polynomisk tid. Fyll ut følgende scenarier (under antagelsen P ≠ NP).

Svar (6%): Kryss av i ruten (☐) ved den mest spesifikke klassen som gjelder.

A er i …	**B er i …**
P	☐ P / ☐ NPC / ☑ NP
NPC	☐ P / ☑ NPC / ☐ NP
☑ P / ☐ NPC / ☐ NP	**P**
☐ P / ☐ NPC / ☑ NP	**NPC**

Du skal beregne avstander mellom byer i veinett. Hvert veinett representeres av en urettet graf $G = (V, E)$, der hver kant $e = \{u,v\}$ i E har en vekt $w(e)$ som tilsvarer lengden på veistrekningen mellom u og v. Du vet at det er begrenset (dvs. $O(1)$) hvor mange veier som er tilkoblet hver by.

i) Du skal altså løse alle-til-alle korteste vei-problemet for slike grafer. Hvilken algoritme vil du bruke?

Svar (6%): Dijkstras algoritme fra hver node. (Det gis også noe uttelling for Floyd-Warshall)

Forklaring: Kjøretid for Dijkstra (med binærheap) fra alle noder når vi har $O(V)$ kanter (jfr. oppgavebeskrivelsen) er $O(V \cdot E \cdot \log V) = O(V^2 \log V)$. Kjøretiden til Floyd-Warshall, for eksempel, er $\Theta(V^3)$.

j) Hva blir kjøretiden til Prims algoritme på en sammenhengende (urettet) graf med n noder og m kanter om man bruker en usortert tabell som prioritetskø, men fortsatt bruker nabolister? (Man må altså gå gjennom hele prioritetskøen både for å finne neste node *og* for å oppdatere prioriteten til en node.)

Svar (6%): $O(mn)$.

Forklaring: Vi må først bygge prioritetskøen (ingen kostnadsøkning der). Deretter må vi bruke prioritetskøen én gang for hver gang vi skal finne neste node, altså $O(n)$ ganger, $O(n)$ hver gang. Vi må også endre nøkkelen for hver av nabokantene, til sammen $O(m)$ ganger, $O(n)$ her gang, i henhold til oppgaven. Vi får da $O(n^2 + nm)$. Siden grafen er sammenhengende, har vi $\Omega(n)$ kanter, og kjøretiden blir $O(nm)$.

k) Du har konstruert et Huffman-tre over n symboler. Hvis du for hver forgrening kan velge fritt hvilken kant som får verdien 0 og hvilken som får verdien 1, hvor mange ulike koder (dvs. trær) kan du konstruere?

Svar (7%): $2^{(n-1)}$ (To alternativer for hver av de n–1 interne nodene.)

Systemer som GNU Automake brukes til å pakke og kompilere programkode. I Automake kan du ha *betingede* delkataloger – kataloger med kode som ikke nødvendigvis kompileres eller brukes, avhengig av en (boolsk) betingelse. Når du bruker Automake til å bygge et arkiv (en tar- eller zip-fil) for distribusjon så inkluderes alle disse katalogene, siden man ikke på forhånd kan vite om de vil kompileres eller ikke, siden det kommer an på ulike betingelser hos brukeren. Du har en kollega som likevel ønsker å prøve å få til en viss effektivisering, ved å ikke pakke delkataloger der betingelsen *aldri kan bli sann*.

l) Hva vil være den viktigste utfordringen når man skal utvikle en slik algoritme?

Svar (7%): Man prøver her å løse SAT, som er NP-komplett.

Du skal lage et program for å designe trapper. Antall trinn (n), og trappetrinnenes bredde og dybde, er gitt. Trappene må tilpasses omgivelsene på ulike vis, og det er derfor knyttet en kostnadsfunksjon $c_i(\cdot)$ til hvert trinn i, som gir kostnaden $c_i(h)$ for å gi trinnet en viss høyde h (målt fra bakkenivå). Samtidig har du en kostnadsfunksjon $r(\cdot)$ for hvor mye høyere trinn k er enn trinn k–1. Hvis høydeforskjellen er x så er kostnaden for denne altså $r(x)$.

Du skal fylle inn en høydetabell $h[0\ldots n{+}1]$, der $h[0]$ og $h[n{+}1]$ er gitt, og der den totale kostnaden er $c_0(h[0]) + r(h[1]{-}h[0]) + c_1(h[1]) + \ldots + r(h[n{+}1]{-}h[n]) + c_{n+1}(h[n{+}1])$. Alle høyder måles i hele cm. Du kan anta at $h[0] < h[1] < \ldots < h[n] < h[n{+}1]$ og at $h[n{+}1]$ er $O(1)$.

m) Skisser en så effektiv algoritme som mulig som konstruerer en trapp med minimal totalkostnad.

Svar (8%): DP, med delproblemer bestemt av trinn og høyde. Finn beste trapp frem til det trinnet med den høyen. For hvert trinn, og hver høyde, kan du da velge forrige trinn som gir best totale kostnad, og lagre denne. (Her får man også uttelling for å beskrive hver kombinasjon av trinn og høyde som en node, og så lage en graf, og finne korteste vei med DP.)

Ved Institutt for romskip og dinosaurer har de n studenter og n prosjekter, og skal koble sammen disse. Policyen deres er å fordele prosjektene ved loddtrekning, men så la studentene bytte seg imellom. For å hjelpe studentene med byttingen, publiseres både en liste med hvem som er tildelt hvilket prosjekt og en liste med hvem som ønsket seg hvilket prosjekt. (Kun én student er tildelt et gitt prosjekt, og omvendt, men flere kan naturligvis ha ønsket seg det samme.)

Du vil hjelpe studentene, og vil lage en algoritme som løser følgende problem: Gitt både tilordningen og listen over førstevalg, finn ut om en gitt student S kan bytte til seg sitt førstevalg. Byttingen kan involvere flere personer, men du kan bare regne med at folk vil bytte hvis de får bytte til sitt førstevalg.

n) Beskriv en algoritme som løser problemet. Hva blir kjøretiden?

Svar (8%): Lag en rettet graf med prosjekter og studenter som noder. Ha en kant fra hvert prosjekt til studenten som har prosjektet, og fra hver student til prosjektet han eller hun ønsker seg. Start med S og let etter en sykel (med traversering). Lineær kjøretid.

La $G = (V, E)$ være en vektet, rettet graf der alle sykler har positiv vektsum. Vi sier at to stier fra i til j er ulike dersom sekvensen med noder man besøker når men følger stiene ikke er nøyaktig de samme. (Stiene må dermed ha noen forskjellige kanter.) Beskriv en algoritme som for hvert par med noder i, j finner antall ulike korteste stier fra i til j i G. Algoritmen skal altså beregne n^2 svar, ett for hvert av de n^2 nodeparene. (Antall stier fra en node i til den samme noden kan du sette til 0 eller 1, etter eget valg.)

o) Beskriv en algoritme som effektivt løser problemet.

Svar (10%): Utvid Floyd-Warshall.

La $D[1\ldots n, 1\ldots n]$ være avstandsmatrisa og $C[1\ldots n,1\ldots n]$ være antallet stier.

I hver iterasjon k, for hvert par i, j får du da:

$D_{new} = D[i, k] + D[k, j]$
$C_{new} = C[i, k] \cdot C[k, j]$
if $D_{new} < D[i, j]$
    $D[i, j] = D_{new}$
    $C[i, j] = C_{new}$
else if $D_{new} == D[i, j]$
    $C[i, j] = C[i, j] + C_{new}$