Avsluttende eksamen i
IT1105 Algoritmer og datastrukturer

Eksamensdato	8. juni
Eksamenstid	0900–1300
Sensurdato	29. juni
Språk/målform	Bokmål
Kontakt under eksamen	Magnus Lie Hetland (tlf. 91851949)
Tillatte hjelpemidler	Alle trykte/håndskrevne; bestemt, enkel kalkulator

Vennligst les hele oppgavesettet før du begynner, disponer tiden og forbered evt. spørsmål til faglærer kommer til eksamenslokalet. Gjør antagelser der det er nødvendig. Skriv kort og konsist. Lange forklaringer og utledninger som ikke direkte besvarer oppgaven tillegges liten eller ingen vekt.

Oppgave 1

Fakta-/pensum-kunnskap o.l. Basert på tilfeldig valgte sider i boka.

a. Løs rekurrensen $T(n) = 3T(n^{1/3}) + \log_3 n$, $T(3) = 0$. Forklar kort utregningen.

Løsning: La $m = \log_3 n$ og la $S(m) = T(3^m)$. Vi får da ligningen

$$T(3^m) = 3T(3^{m/3}) + m \text{ , eller } S(m) = 3S(m/3) + m \text{ ,}$$

som gir $S(m) = m \cdot \log_3 m$, eller $T(n) \in \Theta(\lg(n) \cdot \lg(\lg(n)))$.

Dr. Albatross har sendt deg et brev der han stiller et snurrig algoritmespørsmål. Først definerer han *det halvstørste tallet* i en heltallstabell som det tallet som *ville ha vært* midt i tabellen dersom den var sortert (evt. tallet til venstre for midten hvis antallet tall er et partall). Han har funnet en algoritme som beregner det halvstørste tallet i en tabell, men han er bekymret for at hans rival Dr. Znurrebarth vil kunne fremtvinge en dårlig asymptotisk kjøretid ved å konstruere slemme datasett. Du gjenkjenner algoritmen fra pensum og kan betrygge ham om at han er trygg for Dr. Znurrebarth.

b. Hvilken algoritme er det snakk om, og hvilken asymptotisk kjøretid kan du garantere Dr. Albatross?

Løsning: Doktorens begrep «halvstørst tall» er selvfølgelig ekvivalent med *median*, og algoritmen det er snakk om er SELECT, med *worst-case*-kjøretid $O(n)$.

c. Hvis et problem har optimal substruktur men *ikke* overlappende delproblemer – hvordan kan det da være naturlig å løse problemet?

Løsning: Med rekursjon, etter Splitt-og-hersk-prinsippet (divide and conquer; se s. 28).

d. Den vanlige løsningen på *0-1 knapsack*-problemet med dynamisk programmering ser på overflaten ut til å ha polynomisk kjøretid, men har det egentlig ikke. Diskuter svært kort.

Løsning: Den har eksponensiell kjøretid som funksjon av problemstørrelsen. La $m = \lg W$, der W er kapasiteten. Da er kjøretiden $\Theta(nW) = \Theta(n2^m)$.

e. Dijkstras algoritme fungerer generelt ikke på rettede grafer som har negative kantvekter. Gi ett eksempel på en rettet graf med minst én negativ kantvekt der Dijkstras algoritme gir riktig svar (fra én bestemt node) og ett eksempel på en rettet graf med minst én negativ kantvekt der Dijkstras algoritme gir galt svar.

Løsning: Det enkleste eksempel der den gir riktig svar er en graf med to noder og en negativ kant imellom dem. (Den gir da riktig svar samme hvilke av nodene man starter i.) Det enkleste eksemplet der den gir galt svar har tre noder *x*, *y* og *z*, med kanter (x, y), (y, z) og (x, z) med kortere vekt på (x, z) enn (x, y) og negativ kantvekt på (y, z), som resulterer i en kortere sti x–y–z enn direkte x–z.

Oppgave 2

Du ønsker å finne den lengste stien mellom to oppgitte noder i en vektet, rettet graf med positive kantvekter. (Merk: En sti kan ikke inneholde sykler.)

a. Vis kort at dette problemet ikke har optimal substruktur.

Løsning: Se f.eks. tegning på s. 343 i pensum. Om man finner den lengste veien mellom to noder vil ikke nødvendigvis delstiene være de lengste mellom sine respektive endenoder.

b. En naturlig løsningsidé kan være å multiplisere alle kantvekter med –1 og så bruke for eksempel Bellman-Ford for å finne korteste vei med de nyeste vektene. Vis at denne løsningen ikke nødvendigvis vil gi korrekt svar.

Løsning: Her vil ethvert eksempel på ukorrekt atferd gi full uttelling. Evt. kan man argumentere for at (positive) løkker i input-grafen vil få BF til å feile. Merk: Man kan argumentere for at det å gi opp her faktisk vil være korrekt *atferd* – poenget er at man da ikke får noe svar (ei heller et korrekt et). Med andre ord vil algoritmen aldri gi et *galt* svar.

c. For en bestemt, generell type rettede, positivt vektede grafer vil løsningsideen i **b** gi korrekt svar. Hvilken type grafer er dette, og hvordan kan problemet løses mer effektivt på disse? Hva blir kjøretiden for denne mer effektive løsningen? Skriv kort.

Løsning: Asykliske grafer (DAGs). Her kan man bruke dynamisk programmering (som i DAG-Shortest-Path) og løse problemet i lineær tid. (Hvis studenten kommer med en annen relativt generell klasse grafer der man får riktig svar så kan det også aksepteres.)

d. Diskutér kort følgende påstand: «Man kan i polynomisk tid transformere enhver rettet, vektet graf G til en annen rettet, asyklisk graf G' slik at en løsning på *korteste vei*-problemet i G' tilsvarer en løsning på *lengste vei*-problemet i G.»

Løsning: En diskusjon som påpeker at korteste vei-problemet kan løses i polynomisk tid (det vil si, at problemet er i P), at lengste vei-problemet er i NPC, og at en slik reduksjon ville plassere det i klassen P (med konklusjonen at påstanden er lite trolig) vil kunne gi en del poeng, selv om den er gal. Både korteste vei-problemet og lengste vei-problemet er NP-komplette (og helt ekvivalente) hvis man ikke har noen begrensninger på kantvektene. Korteste vei-problemet der man ikke har negative sykler, derimot, er i P (akkurat som at lengste vei-problemet for grafer uten positive sykler er i P). Man kan få høy uttelling uten å innse f.eks. at korteste vei-problemet (i sin generelle form) er NP-komplett.

Oppgave 3

Interpoleringssøk er en videreutvikling av binærsøk. Input er en sortert tabell $A[1\ldots n]$ og en nøkkel k. Anta at alle verdiene i A er forskjellige. Når det søkes i et intervall fra og med indeks q til og med indeks r vil binærsøk splitte intervallet rundt indeksen gitt ved $(q + r)/2$ (rundet ned) mens interpoleringssøk splitter rundt indeksen gitt ved

$$q + (q - r) \cdot (k - A[q]) \,/\, (A[r] - A[q]) \,,$$

rundet ned. Hvis elementene i A stiger lineært vil denne indeksen være posisjonen til den søkte nøkkelen (eller like ved). Søketiden vil da være $O(1)$.

a. Beskriv hvordan input-data må fordeles og hva søkenøkkelen må være for å frembringe *worst-case*-kjøretid. (Tabellen A skal fremdeles være sortert.) Hva blir denne kjøretiden?

Merk: Her er det en skrivefeil i oppgaven – det skulle ha stått $(r - q)$. Studenter som har tolket formelen slik den var ment (ut fra oppgaveteksten) har fått uttelling i henhold til dette. En bokstavlig tolkning har også blitt vurdert som korrekt. På grunn av problematisk oppgaveformulering tas denne oppgaven kun med i sensur dersom det teller positivt for studentens poengsum.

Løsning (tenkt formulering): Et enkelt eksempel er f.eks. $A = [1, 2, 3, \ldots, n-1, n^2]$, $k = n$. Kjøretiden blir da $\Theta(n)$.

Oppgave 4

Et arbeidsgruppe skal settes sammen for et bestemt prosjekt. De ulike stillingene i gruppen utgjør et hierarki som kan beskrives med en trestruktur, der rotnoden i et deltre er sjefen for deltreet. For hver av de n stillingene er det k kandidater (ingen er kandidat til mer enn én stilling). Din oppgave er å velge ut hvilke kandidater som skal tilsettes i de ulike stillingene, i henhold til de følgende retningslinjene:

- Hver kandidat x har en *kompetanse* $A(x)$
- Et hvert par kandidater x og y har en *kompatibilitet* $B(x, y)$

Kvaliteten til gruppen er summen av de tilsatte kandidatenes kompetanser og kompatibiliteten mellom kandidater som er i nabo-stillinger (sjef og direkte underordnet).

a. Beskriv hvordan du vil løse dette problemet så effektivt som mulig. Hva blir kjøretiden som funksjon av n og k? Begrunn svaret.

Løsning: Problemet kan løses med dynamisk programmering. Hvert deltre utgjør et naturlig delproblem, og man kan finne optimal kvalitet for enhver rot-kandidat. For en gitt rot-kandidat kan de ulike rotkandidatene til deltrærne (de direkte underordnede) velges uavhengig av hverandre. Når en kandidat i en node skal bestemmes må alle k kandidater prøves ut. For hver barnenode må også hver kandidat prøves ut. Det vil si at for hver kant er det k^2 kombinasjoner som må prøves ut. Det er $O(n)$ kanter så kjøretiden blir $O(nk^2)$.