NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

BOKMÅL

AVSLUTTENDE EKSAMEN I

IT1105

ALGORITMER OG DATASTRUKTURER

Mandag 29. mai 2006
Kl. 09.00 – 13.00

Faglig kontakt under eksamen:

Magnus Lie Hetland, tlf. 918 51 949

Hjelpemidler:

Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Sensurdato:

19. juni 2006. Resultater gjøres kjent på http://studweb.ntnu.no og sensurtelefon 815 48 014.

Viktig:

Les hele oppgavesettet før du begynner. Les oppgaveformuleringene grundig. Det er angitt i poeng hvor mye hver deloppgave teller ved sensur. Gjør antagelser der det er nødvendig. Skriv kort og konsist. Skriv fortrinnsvis i rutene på oppgavearket.

Oppgave 1 (30%)

Hver av de følgende deloppgavene består av 5 utsagn. Ved hvert av disse utsagnene skal du sette kryss ved «ja» eller «nei». Sett kryss ved «ja» hvis du mener utsagnet er sant og ved «nei» hvis du mener at utsagnet er usant eller ikke stemmer (evt. ikke gir mening). Ikke kryss av for «nei» hvis du er *enig* i et utsagn som bruker ordet «ikke». Kryss da av for «ja», som betyr at utsagnet *stemmer*.

Hvert utsagn der krysset er satt riktig gir 1.5 poeng, et utsagn uten kryss gir 0 poeng, og et utsagn der krysset er satt galt gir –1.5 poeng. En negativ totalsum for en deloppgave (a–d) rundes opp til 0.

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.

a) Binære hauger (*binary heaps*) er sentrale, grunnleggende datastrukturer som inngår som komponenter i flere klassiske algoritmer. Betrakt en binær maks-haug (*max-heap*) med heltall, representert på vanlig måte som en tabell, $A[1] \ldots A[n]$. (Merk at haugen også kan tolkes som et binært tre, slik at i er en *node*, og $A[i]$ er verdien i noden.)

[ ] Ja [ ] Nei 1. Anta at i er delelig på 2. Foreldrenoden til i er da $i / 2$.

[ ] Ja [ ] Nei 2. Haugen A kan i verste tilfelle (*worst-case*) få en høyde på $\Theta(n)$.

[ ] Ja [ ] Nei 3. Haugen A kan i verste tilfelle (*worst-case*) bygges i lineær tid fra n heltall.

[ ] Ja [ ] Nei 4. En binær maks-haug med kapasitet $k \leq n$ kan brukes til å finne de k minste tallene i A med kjøretid $O(n \log k)$.

[ ] Ja [ ] Nei 5. Medianverdien til verdiene i haugen A kan finnes med kjøretid $\Theta(\log n)$.

b) Et klassisk algoritmisk problem er å finne den korteste veien (*shortest path*) mellom to noder i en vektet, rettet graf. Anta i det følgende at $G = (V, E)$ er en slik graf, med positive heltall som kantvekter. Anta at G' er en annen graf med vilkårlige kantvekter, og G'' enda en graf, der alle kantvektene er 1. Alle grafene kan inneholde sykler, og har m kanter og n noder.

[ ] Ja [ ] Nei 1. Kruskals algoritme finner korteste vei (*én-til-alle*) i G ved hjelp av grådighet

[ ] Ja [ ] Nei 2. Dijkstras algoritme vil finne korteste vei (*én-til-alle*) i G'.

[ ] Ja [ ] Nei 3. Bredde-først-søk vil finne korteste vei (*én-til-alle*) i G''.

[ ] Ja [ ] Nei 4. Bellman-Ford finner korteste vei (*alle-til-alle*) i G' med kjøretid $O(mn)$.

[ ] Ja [ ] Nei 5. Bellman-Ford kan oppdage negative sykler i G' med kjøretid $O(mn)$.

c) Anta at du har to binære søketrær, B_1 og B_2. B_1 bruker ingen balanseringsmekanisme (det vokser vilkårlig etter hvert som elementer settes inn) mens B_2 er et *rød-svart*-tre (*red-black tree*).

[ ] Ja [ ] Nei 1. I gjennomsnitt (*average-case*) vil B_2 være mer effektivt (asymptotisk) enn B_1.

[ ] Ja [ ] Nei 2. I beste tilfelle (*best-case*) vil B_1 kunne få lavere høyde (asymptotisk) enn B_2.

[ ] Ja [ ] Nei 3. I verste tilfelle (*worst-case*) vil B_2 alltid være mer effektivt (asymptotisk) enn B_1.

[ ] Ja [ ] Nei 4. I verste tilfelle (*worst-case*) vil B_2 kunne ha røde noder med røde barnenoder.

[ ] Ja [ ] Nei 5. I beste tilfelle (*best-case*) vil B_2 kunne ha svarte noder med svarte barnenoder.

d) Anta at du har to NP-komplette problemer A og B. Anta også at du har et problem C fra mengden P.

[ ] Ja [ ] Nei 1. Hvis A befinner seg i P vil B også gjøre det.

[ ] Ja [ ] Nei 2. Hvis du kan redusere problemet C til problemet A i polynomisk tid så følger det at C er NP-komplett.

[ ] Ja [ ] Nei 3. Hvis P = NP vil det likevel finnes problemer i NP som ikke er NP-komplette

[ ] Ja [ ] Nei 4. Hvis P ≠ NP så er NP-komplette problemer ikke beregnbare.

[ ] Ja [ ] Nei 5. 1-0-ryggsekkproblemet (kjøretid $O(nW)$) er NP-komplett.

Oppgave 2 (15%)

Vi innfører følgende ekstraregler for bygging av Huffman-trær:

1. Ved sammenslåing av to deltrær, sett alltid inn det deltreet med lavest totalkostnad til venstre.
2. Hvis du skal slå sammen to deltrær med *samme* totalkostnad, sammenlign løvnodene lengst til venstre i de to deltrærne. Deltreet med løvnoden som kommer tidligst i alfabetet settes inn til venstre.
3. Kanter nedover til venstre merkes med `0`, kanter nedover til høyre merkes med `1`.

Anta at du har en tekst med følgende tegnfrekvenser:

$f(\texttt{a}) = 1, f(\texttt{b}) = 2, f(\texttt{c}) = 3, f(\texttt{d}) = 4, f(\texttt{e}) = 5, f(\texttt{f}) = 6.$

a) Tegn et Huffman-tre for teksten, i henhold til reglene gitt over.

Svar: (10%)

b) Bruk treet i a) til å finne en binær koding for strengen «`cafe`». Oppgi den resulterende binære strengen.

Svar: (5%)

Oppgave 3 (15%)

Et *k*-regulært tre er et tre der hver interne node har *k* barnenoder. Du skal finne en funksjon $L(n)$ for hvor mange løvnoder et slikt tre med *n* interne noder har. For eksempel vil $L(1) = k$. Merk at treet *ikke* trenger være balansert.

a) Sett opp sammenhengen mellom $L(n)$ og $L(n–1)$ som en ligning (det vil si, en *rekurrensligning*), der $n > 1$.
Hint: Hvor mange ekstra løvnoder får du hvis du bytter en løvnode ut med en ny intern node?

Svar: (7%)

$L(n) =$

b) Løs ligningen (rekurrensen) fra a) og finn et asymptotisk uttrykk for $L(n)$. Uttrykket skal også vise hvordan antallet løvnoder avhenger av konstanten *k*. Vis utregningen (kort). Bruk Θ-notasjon.

Svar: (8%)

Utregning:

$L(n) \in \Theta($ $)$

Oppgave 4 (20%)

Du ønsker å rangere deltakerne i VM i Algoritmekonstruksjon ut fra ferdighetsnivå. Anta at det er mulig å finne en éntydig slik ordning. For å lette arbeidet ønsker du å begynne med å lage en *delvis* ordning, i form av en rettet, asyklisk graf, eller DAG. Prosedyren for å bygge denne DAG-en er enkel: Du plukker ut to kandidater, legger dem til i grafen som noder (hvis de ikke alt er med), og legger til en kant mellom dem, fra den beste til den dårligste. Dette gjentar du til ordningen er entydig, det vil si, til det finnes kun én topologisk ordning av grafen. Anta at det er n deltagere som skal ordnes. Du står fritt til å velge hvilke to kandidater du vil sammenligne til enhver tid. Du vet ikke noe om kandidatenes innbyrdes ordning før du begynner å sammenligne.

a) Hvor få kanter kan du bruke, i det beste tilfellet (*best-case*), hvis du er så effektiv som mulig (dvs. bruker så få kanter som mulig)? Skriv svaret som en eksakt funksjon av n.
Merk: Siden det er snakk om beste mulige tilfelle kan du anta at du har maksimal «flaks» ved utvelgelse av kandidater for sammenligning.

Svar: (8%)

b) Hvor mange kanter kan du bli nødt til å bruke, i verste tilfelle (*worst-case*), hvis du er så effektiv som mulig (dvs. bruker så få kanter som mulig)? Skriv svaret asymptotisk som en funksjon av n. Bruk Θ-notasjon.

Svar: (12%)

$\Theta($ $)$

Oppgave 5 (20%)

Du har kjøpt opp n restauranter som ligger ved siden av hverandre i samme gate. Du ønsker å slå sammen alle restaurantene til én stor restaurant, men du kan bare slå sammen to om gangen, og disse to må ligge ved siden av hverandre. Størrelsen til den nye restauranten er summen av størrelsene til de to restaurantene som blir slått sammen. Kostnaden ved en sammenslåing er lik størrelsen på den resulterende restauranten. Den totale kostnaden ved n–1 sammenslåinger er summen av kostnadene til de individuelle sammenslåingene. Du ønsker å finne en sammenslåing med så lav total kostnad som mulig.

Anta at du har oppgitt størrelsene til de n restaurantene i en tabell $S = S[1]\ldots S[n]$.

a) Du vil først beregne innholdet i tabellen $L[1\ldots n, 1\ldots n]$, der $L[i, j]$ er den totale lengden av restaurantene fra og med nummer i til og med nummer j. Du trenger kun beregne verdiene for $i \leq j$. Skissér kort en enkel algoritme som beregner L.

Svar: (3%)

Du ønsker å finne en effektiv algoritme som bruker dynamisk programmering (DP) for å beregne den minimale totale kostnaden ved å slå sammen alle de n restaurantene.

b) Hva blir de naturlige delproblemene i DP-løsningen?

Svar: (7%)

c) Beskriv kort en algoritme som løser problemet. Hva blir kjøretiden i Θ-notasjon? Du kan anta at tabellen L fra a) allerede er beregnet og kan brukes her.

Svar: (10%)

Kjøretid: Θ( )