Avsluttende eksamen i
TDT4120 Algoritmer og datastrukturer

Eksamensdato	18. august 2011
Eksamenstid	0900–1300
Sensurdato	8. 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.

Oppgave 1

a) Hva er kjøretiden til MERGE-prosedyren i *mergesort*? (Anta n elementer.)

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

b) Gi et eksakt svar på rekurrensen $T(n) = 2T(n/2) + n$ (dvs. uten bruk av asymptotisk notasjon), der $n = 2^k$ for et heltall $k > 1$, og $T(2) = 2$.

Svar (6%): $T(n) = n \lg n$
Merk: Det forventes at studenten kjenner den asymptotiske løsningen, $\Theta(n \lg n)$. Innsetting for $n = 2$ gir svaret (som kan sjekkes med induksjon).

I følgende setninger fra læreboka er enkelte ord fjernet:

> «Even if the input is not drawn from a _____ distribution, _____ may still run in linear time. As long as the input has the property that the sum of the squares of the _____ sizes is linear in the total number of elements, equation ____ tells us that _____ will run in linear time.»

c) Hvilken algoritme er det snakk om?

Svar (7%): *Bucket sort*.
Sitatet er tatt fra side 204. Det forventes ikke at studenten skal ha full oversikt over matematikken her. Følgende ledetråder er nok til å konkludere riktig: Input er vanligvis forventet å følge en bestemt fordeling – algoritmen har da lineær kjøretid; man kan velge størrelsen på komponenter som brukes; algoritmen arbeider på «elementer».

d) Hva er den forventede kjøretiden til et oppslag i en hashtabell, hvis du antar at antallet posisjoner (*slots*) i tabellen minst er proporsjonalt med antall elementer?

Svar (6%): $\Theta(1)$

e) For at dynamisk programmering skal være korrekt må problemet ha såkalt optimal substruktur. For at dynamisk programmering skal *ha noe for seg* (altså gi en ytelsesgevinst) krever vi også en annen egenskap ved problemet. Hvilken?

Svar (7%): Overlappende delproblemer

f) Hvilket av følgende uttrykk (1 til 3) er korrekt?

1. $T_p \leq T_1/P + T_\infty$
2. $T_p = T_1/P + T_\infty$
3. $T_p \geq T_1/P + T_\infty$

Svar (6%): 1

g) Læreboka sin formulering av korteste-vei–problemet som et lineært program er basert på en pensumalgoritme som løser samme problem – hvilken?

Svar (6%): Bellman-Ford

h) Hva blir summen $1 + 2 + \ldots + n$, uttrykt i asymptotisk notasjon?

Svar (6%): $\Theta(n^2)$

Oppgave 2

Du har oppgitt to tabeller, A og P med lengder på henholdsvis n og m. Hvert element $P[i]$ er en indeks som angir et element i A (det elementet er altså $A[P[i]]$). Du kan anta at P er sortert. Du skal endre A så den fortsatt inneholder de opprinnelige elementene, men muligens i en annen rekkefølge (en permutasjon), slik at elementene angitt av P ligger først i A.

a) Beskriv en algoritme som løser problemet så effektivt som mulig. Hva blir kjøretiden, som funksjon av m og n?

Svar (6%): For $i = 1 \ldots m$, bytt $A[i]$ og $A[P[i]]$. Kjøretid: $\Theta(m)$.
(Merk at hvis P ikke var sortert kunne man risikere at $P[i] < i$, som vil gi feil – $P[i]$ vil være «ødelagt». Hvis P er sortert vet vi at $P[i] \geq i$, så alle tidligere elementer i A er trygge. Korrekthet vises lett ved induksjon.)

Du mottar en jevn strøm av heltall over et nettverk, og ønsker til enhver tid å ta vare på de m minste verdiene du har mottatt så langt. Du ønsker å minimere den asymptotiske kjøretiden (som funksjon av m) for å prosessere hvert element.

b) Beskriv en algoritme/datastruktur som løser dette problemet. Oppgi kjøretiden per element du mottar, som funksjon av m, etter at minst m elementer har blitt mottatt.

Svar (6%): Bruk en binær maks-heap. Så lenge den er mindre enn m, legg inn hvert element. Etter det, legg inn nye elementer kun hvis de er mindre enn maksimum, og fjern det største elementet. Kjøretid: $\Theta(\lg m)$.
Andre løsninger, som å bruke lineært søk i en usortert tabell med m elementer, vil også gi noe (men ikke full) uttelling.

Oppgave 3

Anta at du har oppgitt en rettet, asyklisk graf (DAG) $G = (V, E)$, samt noder s og t i V.

a) Hvordan ville du telle antall mulige stier fra s til t i G?

Svar (6%): Dynamisk programmering: For hver node, legg sammen antall stier til forgjengerne (der s har 1). Bruk memoisering, eller topologisk sortering (som i DAG-SHORTEST-PATH).

b) Hvordan vil du telle (det maksimale) antall stier som kan følges *samtidig* fra s til t, hvis disse stiene ikke kan dele kanter?

Svar (6%): Kjør maks-flyt fra s til t med kapasiteter på 1.

Betrakt en urettet graf $G = (V, E \cup F)$, der $E \cap F = \varnothing$. (Med andre ord, kantene til G kan partisjoneres i de ikke-overlappende mengdene E og F.)

Kantene i E representerer konflikter (E for *enemies*) og F representerer vennskap (F for *friends*). Du ønsker nå å avgjøre om nodene i V kan partisjoneres i to (ikke-overlappende) mengder A og B, slik at *ingen* av kantene i E går mellom to noder i samme mengde, og slik at *alle* kantene i F gjør det. Det vil si, for hver kant $e = \{u, v\}$ i grafen G:

- Hvis e er i E må enten u være i A og v i B eller omvendt.
- Hvis e er i F må både u og v være i samme mengde, enten A eller B.

c) Beskriv en algoritme som (så effektivt som mulig) enten finner en slik partisjon eller som avgjør at det er umulig. Oppgi kjøretiden.

Svar (6%): Traverser grafen og «fargelegg» noder med A eller B. Hver gang en kant fra E følges, skift «farge» fra A til B eller omvendt. Hvis du møter en node som alt har feil farge er problemet uløselig. Ellers er svaret rett. Kjøretiden blir lineær.

Oppgave 4

Følgende oppgave ble gitt ved fjorårets kontinuasjonseksamen:

> «Du skal invitere venner til fest. Du vurderer et sett med n kandidater, men du vet at hver av dem bare vil ha det hyggelig dersom han eller hun kjenner minst k andre på festen. (Du kan anta at dersom A kjenner B så kjenner B automatisk A.) Beskriv en algoritme som finner en størst mulig delmengde av de n vennene dine der alle kjenner minst k av de

andre, dersom en slik delmengde eksisterer. Forklar kort hvorfor algoritmen er korrekt og optimal.»

Du ønsker nå å arrangere en *liten* sammenkomst, og vil dermed løse samme problem, bortsett fra at du vil finne en *minst* mulig delmengde der alle kjenner minst k av de andre.

a) Vis (dvs. forklar kort) at det er urealistisk å løse denne nye varianten av oppgaven.

Svar (6%): Den er NP-hard. Vises ved reduksjon fra CLIQUE: Du kan avgjøre om det finnes en klikk av størrelse k ved å se om minste gruppe består av k elementer.

Grafen $T = (V, E)$ er et tre med rot $r \in V$ og en vektfunksjon w over nodene (dvs., hver node $v \in V$ har vekt $w(v)$). Merk at denne vekten kan være negativ. La $S = (U, F)$ være en delgraf av T. Vektsummen til S er da summen av $w(u)$ for alle noder u i U. Du ønsker å finne vektsummen til den tyngste sammenhengende delgrafen i T som inneholder r.

b) Beskriv en algoritme som løser problemet og som er så effektiv som mulig. Oppgi kjøretiden.

Svar (7%): Direkte rekursiv løsning (à la DFS). Resultatet for et deltre med rot v er enten 0, hvis deltreet ekskludert, eller ($w(v)$ + svaret for hvert barn). Kjørtiden blir lineær.

Oppgave 5

I enkelte typer biologiske sekvenser (som DNA) er såkalte *palindromiske subsekvenser* viktige. Et palindrom er en sekvens som er identisk når den reverseres (for eksempel, «`agnes i senga`») og en palindromisk subsekvens er en subsekvens (av en annen sekvens) som er et palindrom. Sekvensen «`CTATACGGTACGATA`» inneholder for eksempel (blant annet) de palindromiske subsekvensene «`TAT`» og «`AACGGCAA`».

Merk: En *subsekvens* er ikke det samme som en *substreng*. Hvis X er en subsekvens av Y så må hvert av elementene i X kunne finnes igjen i Y, i samme rekkefølge, men ikke nødvendigvis ved siden av hverandre.

Du ønsker nå, gitt en sekvens S av lengde n, å finne lengden til den *lengste* palindromiske subsekvensen P i S. (Det kan selvfølgelig være flere med samme lengde.)

a) Beskriv svært kort (gjerne med referanse til pensum) en algoritme som løser problemet så effektivt som mulig. Argumenter svært kort for at løsningen er korrekt. Oppgi kjøretiden.

Svar (7%): LCS-LENGTH(S, S'), der S' er S reversert. Kjøretid: $\Theta(n^2)$
Siden P er en felles subsekvens for S og S' kan ikke svaret være større. Det kunne bare være mindre om LCS ikke var palindromisk i S – dvs. hvis forekomstene i S og S' ikke overlappet. Man kunne da lage en palindromimsk sekvens ved å kombinere de to forekomstene (union av elementene), som ville gi en lengre felles subsekvens for S og S' (en selvmotsigelse).