

# Avsluttende eksamen i TDT4120 Algoritmer og datastrukturer

<b>Eksamensdato</b>	18. august 2011
<b>Eksamenstid</b>	0900–1300
<b>Sensurdato</b>	8. september
<b>Språk/målform</b>	Bokmål
<b>Kontakt under eksamen</b>	Magnus Lie Hetland (tlf. 91851949)
<b>Tillatte hjelpemidler</b>	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%):

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%):

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%):

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%):

- 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%):

- 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%):

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

Svar (6%):

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

Svar (6%):

## 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%):

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%):

### 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%):

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%):

Betrakt en urettet graf  $G = (V, E \cup F)$ , der  $E \cap F = \emptyset$ . (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%):

## 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%):

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 delgraf i  $T$  som inneholder  $r$ .

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

Svar (7%):

## 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%):