Avsluttende eksamen i TDT4125
Algoritmekonstruksjon, videregående kurs (Bokmål)

Kontakt under eksamen Magnus Lie Hetland (tlf. 91851949)
Tillatte hjelpemidler Alle trykte/håndskrevne; bestemt, enkel kalkulator

Bruk gjerne blyant! Les hele oppgavesettet først, disponer tiden og forbered evt. spørsmål til faglærer kommer til eksamenslokalet. Gjør antagelser der det er nødvendig. Svar konsist, fortrinnsvis i svarskjemaet.

Svarskjema

1a (13 %)
1b (8 %)
2a (11 %)
2b (8 %)
2c (8 %)
3a (9 %)

3b (12 %)

3c (10 %)

4a (12 %)

4b (9 %)

Oppgave 1

La A være en boolsk tabell (*array*) av lengde n og la ☆ være en hvilken som helst (assosiativ) binær logisk operator.

a. Beskriv en algoritme med kjøretid $\Theta(\lg n)$ som beregner $A_1 \star A_2 \star \ldots \star A_n$ på en EREW PRAM-maskin.

b. Beskriv en annen EREW-algoritme for samme problem som er så effektiv som mulig (med lavest mulig kjøretid), der produktet av kjøretid og antall prosessorer er forventet å være $\Theta(n)$.

Oppgave 2

En binærhaug (*binary heap*) er gjerne representert som en tabell (*array*), og har dermed en fast kapasitet.

a. Hvordan kan man øke kapasiteten til en slik haug uten at den amortiserte kjøretiden til de ulike operasjonene påvirkes? Begrunn svaret.

Betrakt problemet med å finne billigst maks-flyt (*min-cost max-flow*) fra kilde s til sluk t i en graf (uten parallelle kanter) der alle kantene har en kapasitet på 1. Grafen har n noder og m kanter og alle nodene kan nås fra s. Anta at ingen kantkostnader er negative.

b. Gi en tett øvre grense (O-notasjon) for kjøretiden (som funksjon av n og m) for Busacker-Gowen på dette problemet, hvis du bruker prising (*pricing*) og Dijkstras algoritme med en binærhaug (*binary heap*) for å finne de flytforøkende stiene (*augmenting paths*).

La BIPARTITE være problemet å avgjøre hvorvidt en graf er bipartitt.

c. Vis at BIPARTITE $\in$ NL.

Oppgave 3

Anta at du skal flytte eiendeler med størrelser $x_1, x_2, \ldots, x_n$ fra sted A til sted B. Størrelsene er reelle tall mellom 0 og 1 ($0 \leq x_i \leq 1$, for alle i), og eiendelene skal flyttes i en bil med lastekapasitet 1. Det gjelder å lage en plan for alle flyttelassene slik at det blir så få kjøreturer (lass) som mulig. Merk at du skal lage en fullstendig plan før noen kjøringer foretas. Dette problemet er NP-hardt (*NP hard*), og du skal vurdere følgende enkle *first-fit*–approksimerings-algoritme:

Skriv eiendel x_1 på listen for det første lasset og deretter, for $i = 2, 3, \ldots, n$, skriv x_i på listen til det første av de planlagte lassene der det er stor nok kapasitet.

a. Gi et eksempel med $n = 4$ som viser at first-fit–algoritmen ikke er optimal.

b. Vis at first-fit–algoritmen er en 2-tilnærming (2-*approximation*).

(I begrunnelsen kan du bruke m for antall lass i first-fit-løsningen og m^* for det optimale antallet lass. Du kan anta at det er snakk om minst to eiendeler.)

c. Hvis first-fit–algoritmen har en kjøretid på $f(n)$, beskriv en randomisert algoritme med en parameter k, som har kjøretid $k \cdot f(n)$, som alltid gir minst like godt svar som first-fit–algoritmen, som alltid har mulighet til å finne den optimale løsningen, og der større verdier for k gir en høyere forventet kvalitet.

Oppgave 4

Betrakt følgende problem **P**: Du skal planlegge produksjon av et produkt for n måneder fremover. For hver måned $i = 1 \dots n$ har du oppgitt følgende:

p_i Produksjonskostnad per enhet
h_i Lagringskostnad per enhet
d_i Etterspørsel (antall enheter)

La x_i være antall enheter som produseres i måned i, mens o_i er antall enheter som til nå (ved slutten av måned i) er produsert, men foreløpig ikke solgt. Merk at x_i, o_i og d_i er ikke-negative heltall for $i = 1 \dots n$, mens kostnadene er positive rasjonale tall. Målet er å finne en produksjonsplan ($x_1 \dots x_n$) som tilfredsstiller all etterspørsel, men med lavest mulig totalkostnad. Lagringskostnaden (h_i) betales per enhet på lager ved slutten av måned i (altså o_i stk). Etter siste måned (n) skal lageret være tømt.

Anta at du har en algoritme **A** tilgjengelig, som kan løse min-kost–maks-flyt–problemet (*min-cost max-flow*).

a. Tegn **P** for $n = 3$, representert som et nettverksproblem som kan løses av **A**. Forklar figuren.

Betrakt nå problem **Q**, som er likt **P**, bortsett fra følgende ekstrakostnad:

f_i Fast kostnad (oppstartskostnad) for produksjon

Denne faste kostnaden betales kun dersom det produseres minst én enhet i måned i. (Merk at dette er en engangskostnad for måned i, og er *ikke* en kostnad per enhet, i motsetning til p_i.) Hvis $f_i = 0$ for $i = 1 \dots n$ så er **Q** = **P**.

b. Løs problemet **Q** så effektivt som mulig ved hjelp av dynamisk programmering. Du trenger kun finne *kostnaden* til den optimale planen (dvs., det er ikke nødvendig å finne selve planen).

Merk: En løsning med kjøretid $\Theta(n^2)$ eller $\Theta(n^3)$ gir full uttelling. En kjøretid på $\Theta(n^4)$ gir noe trekk.

Hint: Du kan anta at produksjonen i en måned (hvis den er større enn null) fullstendig dekker etterspørselen for et antall påfølgende måneder. Det vil si, hvis $x_i > 0$, så kan du anta at $x_i = d_i + d_{i+1} + \dots + d_{i+k}$, for en eller annen $k \geq 0$. Du kan også anta at $x_i = 0$ hvis $o_i > 0$. En optimal løsning med disse egenskapene vil alltid eksistere.