Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Oppgavestillere:
Magnus Lie Hetland
Jon Marius Venstad

Kvalitetskontroll:
Magnar Nedland

Faglig kontakt under eksamen:
Magnus Lie Hetland (918 51 949)

Eksamen i
TDT4120 Algoritmer og datastrukturer

Torsdag 16. desember 2010
Tid: 09:00 – 13:00 Sensur 17. januar 2011

Hjelpemidler: Ingen trykte/håndskrevne; bestemt, enkel kalkulator

Språk/målform: Bokmål

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. Lange forklaringer og utledninger som ikke direkte besvarer oppgaven tillegges liten eller ingen vekt.

Du kan beskrive algoritmer 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) (6 %) DIJKSTRA og BELLMAN–FORD løser to ulike spesialtilfeller av samme problem. Hva er forskjellen mellom disse spesialtilfellene?
Løsning: Dijkstras algoritme forbyr negative kanter mens Bellman–Ford bare forbyr negative sykler.

b) (5 %) Hvilke krav må stilles til en rettet graf for at den skal kunne sorteres topologisk?
Løsning: Den må være asyklisk.

c) (6 %) Er det mulig å beskrive *worst-case*-kjøretid med Ω-notasjon? Begrunn svaret.
Løsning: Ja. Ω kan brukes på alle kjøretidsfunksjoner, uavhengig av om de kommer av for eksempel *best-* eller *worst-case*-input.

d) (5 %) Hva er *linear probing*? Svar kort.
Løsning: Det brukes ved kollisjoner i hashtabeller. Man leter etter ledig plass ved gjentatte ganger å flytte seg et konstant antall posisjoner videre i tabellen, modulo tabellengden.

e) (6 %) Hva er Hamilton-sykler? Hvorfor er det ingen algoritmer i pensum for å finne dem?
Løsning: En sykel som besøker alle nodene i en graf nøyaktig én gang. Å avgjøre om en graf har en Hamilton-sykel er NP-komplett.

f) (5 %) En flertrådsberegning (*multithreaded computation*) kan ses som en asyklisk, rettet graf (*computation dag*), $G = (V, E)$. Hvilke egenskaper ved G representerer T_1 og T_∞? (Anta at hver deltråd, eller *strand*, tar 1 tidsenhet å utføre.)
Løsning: $T_1 = |V|$ og T_∞ er den kritiske (lengste) stien.

g) (6 %) HEAPSORT er optimal, men RADIX-SORT har bedre asymptotisk kjøretid. Forklar svært kort hvordan dette henger sammen.
Løsning: Heapsort løser det generelle sorteringsproblemet (sortering ved sammenligning), mens RADIX-SORT løser et spesialtilfelle som kan løses mer effektivt.

h) (6 %) Hvorfor er det ikke alltid nyttig å bruke memoisering i rekursive algoritmer?
Løsning: Det er ikke alltid man har overlappende delproblemer.

Oppgave 2

a) (5 %) Hvordan blir maks-haugen (*max-heap*-en) $[21, 14, 16, 0, 2, 15, 13]$ dersom vi setter inn 15, og deretter tar ut største element?
Løsning: ([21, 15, 16, 14, 2, 15, 13, 0] $\rightarrow$) [16, 15, 15, 14, 2, 0, 13]

b) (6 %) Hvis du skal flette sammen flere sorterte lister med varierende lengde, og du kun kan flette to lister om gangen (i lineær tid), hvordan vil du minimere det totale arbeidet som trengs for alle flettingene? (Du skal altså sitte igjen med én sortert liste som består av elementene fra alle de opprinnelige listene.)
Løsning: Flett hele tiden sammen de to minste, for eksempel ved hjelp av en heap. Fremgangsmåten ligner svært på Huffmans algoritme.

c) (5 %) Hvordan vil du reversere en sekvens Q (med lengde n) ved hjelp av en annen sekvens S i $O(n)$ tid hvis sekvensene kun kan aksesseres i endene? De kan brukes både som LIFO-køer (med PUSH og POP) og som FIFO-køer (med ENQUEUE og DEQUEUE). Hver av kø-operasjonene tar konstant tid. Til slutt skal elementene ligge (i reversert rekkefølge) i Q, mens S skal være tom.
Løsning:
while Q: ENQUEUE(S, POP(Q))
while S: PUSH(Q, DEQUEUE(S))

Oppgave 3 Tenk deg at du står overfor tre beslutningsproblemer, FOO, BAR og BAZ. Alle tre problemer ligger i klassen NP. Du vet at FOO ligger i NPC og at BAZ ligger i P. Du ønsker å prøve å konstruere polynomiske reduksjoner mellom noen av disse for å komme frem til ulike konklusjoner.

Merk: Det er snakk om hva du vil gjøre for å *prøve* å komme frem til de ulike konklusjonene. Hvis, for eksempel, problemet BAR *ikke* er i P så vil det jo være umulig å bevise at det *er* i P.

a) (3 %) Hvilket problem vil du redusere til hvilket for å vise at BAR $\in$ P? Forklar kort.
Løsning: Reduserer BAR til BAZ. Kan dermed løse løse BAR ved hjelp av BAZ, altså i polynomisk tid.

b) (3 %) Hvilket problem vil du redusere til hvilket for å vise at BAR $\in$ NPC? Forklar kort.
Løsning: Reduserer FOO til BAR. Alle problemer i NP kan reduseres til FOO, og nå også (indirekte) til BAR.

c) (3 %) Hvilket problem vil du redusere til hvilket for å vise at P = NP? Forklar kort.
Løsning: Reduserer FOO til BAZ. Hvis ett problem fra NPC er i P så vil alle problemer i NP være i P, så P = NP.

Oppgave 4

a) (5 %) Løs rekurrensen $T(n) = 3T(n/2) + n^2$ (asymptotisk). Begrunn svaret kort.
Løsning: $T(n) \in \Theta(n^2)$. Tilfelle 3 av masterteoremet.

b) (4 %) Løs rekurrensen $T(n) = 16T(n/4) + n$ (asymptotisk). Begrunn svaret kort.
Løsning: $T(n) \in \Theta(n^2)$. Tilfelle 1 av masterteoremet.

Oppgave 5

a) (6 %) La $G = (V, E)$ være en komplett, urettet graf, der $V = \{0, \ldots, 9\}$. La kantvektene til G være gitt ved følgende (symmetriske) matrise, W:

```
    0  1  2  3  4  5  6  7  8  9
0 [[0, 4, 7, 9, 5, 3, 6, 6, 5, 0],
1  [4, 0, 8, 9, 6, 9, 0, 3, 3, 9],
2  [7, 8, 0, 1, 0, 3, 3, 2, 3, 0],
3  [9, 9, 1, 0, 8, 2, 8, 3, 3, 0],
4  [5, 6, 0, 8, 0, 5, 2, 3, 6, 6],
5  [3, 9, 3, 2, 5, 0, 8, 9, 7, 1],
6  [6, 0, 3, 8, 2, 8, 0, 6, 3, 1],
7  [6, 3, 2, 3, 3, 9, 6, 0, 6, 3],
8  [5, 3, 3, 3, 6, 7, 3, 6, 0, 2],
9  [0, 9, 0, 0, 6, 1, 1, 3, 2, 0]]
```

Med andre ord er $w_{0,3} = 9$, $w_{4,1} = 6$ og $w_{4,2} = 0$ for eksempel. (Det er ikke noe spesielt med en vekt på 0 her; det betyr for eksempel *ikke* at kanten mangler.)

Finn det minimale spenntreet T for grafen G, og angi (som en sekvens med noder) stien fra node 1 til node 4 i treet T. (Som svar skal du kun oppgi denne stien, og *ikke* hele spenntreet.)

Hint: Går du systematisk til verks, inspirert av en av algoritmene i pensum, krever denne oppgaven atskillig mindre arbeid enn det kan se ut til. Vurder valg av algoritme nøye.
Løsning: 1, 6, 9, 2, 4
Her kan man bruke Kruskals algoritme, og vurdere kantvektene i rekkefølge fra 0 til 9. Halve matrisen kan ignoreres. Tegn en figur. For hver vekt kan man gå gjennom matrisen og lete etter forekomster, og så sjekke om disse danner sykler i figuren. Prims algoritme kan også fungere fint, om man starter i node 1 og stanser når man når node 4.

b) (5 %) La $G = (V, E)$ være en uvektet, rettet graf med noder $V = \{1, \ldots, n\}$. La X være en éndimensjonal heltallstabell som kan indekseres med nodenummerne. La $X[1] = 0$, og la $X[x] = n$ for $x = 2 \ldots n$. Hva gjør da følgende algoritme?

(Referer gjerne til lignende eller beslektede algoritmer i pensum.)

for $x \in V$
  for $(y, z) \in E$
    $X[z] = \min\{X[z], X[y] + 1\}$

Løsning: Uvektet korteste vei. Dette er bare en uvektet versjon av Bellman–Ford. Referanser til BFS (eller, for den saks skyld, Dijkstra) er også til en viss grad relevante, siden det som beregnes er det samme, selv om disse oppfører seg ganske annerledes.

Oppgave 6 Du skal fordele fag på forelesere for neste semester. Du har en liste med fag som skal foreleses, og et sett med forelesere å ta av. For hver foreleser har du en liste med fag som denne er kompetent til å forelese. Foreleserne har også ulik arbeidskapasitet, og du har oppgitt hvor mange fag hver foreleser orker å forelese i løpet av et semester. Målet er å sørge for at alle fagene foreleses uten at noen må forelese noe de ikke kan, og uten at noen må forelese flere fag enn de orker.

a) (5 %) Beskriv hvordan problemet kan løses som et flytproblem.
Løsning: Som vanlig flyt-løsning for bipartitt matching, bare at kantene fra kilden s til forelesere får kapasitet lik arbeidskapasiteten.

Du oppdager nå at det er kollisjoner mellom enkelte av fagene (det vil si, de skal foreleses på samme tidspunkt). Du vil unngå å gi en faglærer to (eller flere) fag som kolliderer. (Dersom to fag kolliderer har de eksakt samme forelesningstidspunkt og varighet. Du kan dermed anta at om fag A kolliderer med fag B og B kolliderer med fag C så kolliderer også A med C.)

b) (5 %) Beskriv hvordan du kan håndtere kollisjonene og fortsatt løse problemet som et flytproblem.
Løsning: For hver faglærer innføres et sett med kollisjonsnoder (én for hver gruppe med kolliderende fag). Lag kant fra faglærer til kollisjonsnodene, med kapasitet 1, og kanter videre til fagene som før.