NTNU
Kunnskap for en bedre verden

Institutt for datateknikk og informasjonsvitenskap

Eksamensoppgave i TDT4120 Algoritmer og datastrukturer

Faglig kontakt under eksamen	Magnus Lie Hetland
Telefon	918 51 949

Eksamensdato	15. august, 2018
Eksamenstid (fra–til)	09:00–13:00
Hjelpemiddelkode/tillatte hjelpemidler	D

Målform/språk	Bokmål
Antall sider (uten forside)	4
Antall sider vedlegg	0

Informasjon om trykking av eksamensoppgave

Originalen er

1-sidig ☑ **2-sidig** ☐

sort/hvit ☑ **i farger** ☐

Skal ha flervalgskjema ☐

Kontrollert av

Dato Sign

Merk: Studenter finner sensur i Studentweb. Har du spørsmål om din sensur må du kontakte instituttet ditt. Eksamenskontoret vil ikke kunne svare på slike spørsmål.

Les dette nøye

(*i*) Les hele eksamenssettet nøye *før* du begynner!

(*ii*) Faglærer går normalt én runde gjennom lokalet. Ha evt. spørsmål klare!

(*iii*) Skriv bare så mye du mener er nødvendig. Svar som inneholder irrelevante aspekter kan regnes som helgardering, og det kan trekke ned!

Merk: Varianter av de foreslåtte svarene nedenfor vil naturligvis også kunne gis uttelling, i den grad de er helt eller delvis korrekte.

Oppgaver med løsninger

5% 1. Betrakt følgende utsagn (der reduksjonene antas å ta polynomisk tid):

   (*i*) Et problem B i **NP** er **NP**-komplett dersom alle problemer i **NP** kan reduseres til B.

   (*ii*) For $A, B \in$ **NP**, der A er **NP**-komplett: Dersom vi kan redusere fra A til B så er B **NP**-komplett.

   Forklar kort hvorfor (*ii*) følger av (*i*).

   Det er en direkte konsekvens av at reduksjon er transitivt: Om vi kan redusere fra C til A, så kan vi også redusere til B, ved å gå via A. Summen av to polynomiske kjøretider er fortsatt polynomisk, så vi kan altså redusere fra alle problemer i **NP** til B.

   Relevant læringsmål: Forstå hvordan NP-kompletthet kan bevises ved én reduksjon

5% 2. Hva er kjøretiden til Insertion-Sort på en sortert tabell?

   $\Theta(n)$

   Relevant læringsmål: Forstå Insertion-Sort

5% 3. Hva sier heltallsteoremet (*the integrality theorem*)?

   Om alle kapasiteter er heltall, så er totalflyt og flyten i alle kanter heltall.

   Relevant læringsmål: Forstå *heltallsteoremet* (*integrality theorem*)

5% 4. Om vi viser at grådighetsegenskapen (*the greedy-choice property*) holder har vi likevel ikke vist at grådighet er korrekt. Forklar.

   Vi må vise at vi har optimal delstruktur. Grådighetsegenskapen viser at vi kan gjøre *ett* grådig valg uten å miste muligheten til å finne optimum, men det betyr ikke at vi kan *fortsette* å velge grådig. For å kunne si det, må vi bruke induksjon, og må dermed vite at vi etter et grådig valg sitter igjen med et problem av samme form.

   Relevant læringsmål: Forstå *grådighetsegenskapen* (*the greedy-choice property*)

5% 5. Du starter med en sammenhengende graf $G = (V, E)$ og sletter kanter fra E til det finnes nøyaktig én sti mellom ethvert par med noder $u, v \in V$. Hva kalles den resulterende grafen $G' = (V', E)$? (Merk: Her spørres det etter forholdet mellom G' og G, ikke hva G' er isolert.)

   Et spenntre.

   Relevant læringsmål: Vite hva *spenntrær* og *minimale spenntrær* er

5% 6. Du prøver å implementere BFS for urettede grafer, men på grunn av en kodefeil, er rekkefølgen på nodene i køen din ikke lenger FIFO, men helt vilkårlig. Kan du nå være sikker på å besøke alle nodene? Forklar.

Ja, dersom grafen er sammenhengende. (Dvs., vi får besøkt de samme nodene som for BFS.) Om vi ikke har besøkt alle nodene, må minst én av dem være nabo av en av dem vi har besøkt. Den vil dermed ha blitt lagt i køen, og vil bli plukket ut på et eller annet tidspunkt. Vi vil med andre ord ikke avslutte før alle er besøkt.

Relevant læringsmål: Forstå *traversering med vilkårlig prioritetskø*

5% 7. Vi kan memoisere en funksjon, så vi slipper å regne den ut flere ganger om den kalles med samme argumenter igjen. Det kan være en nyttig optimalisering i praksis, som en form for hurtigminne (*cache*), men har det noen teoretisk betydning for asymptotisk kjøretid? Forklar.

Ja. Vi kan spare et eksponentielt antall kall til funksjoner lenger ned i kalltreet, dersom vi har overlappende delproblemer.

Relevant læringsmål: Forstå løsning ved *memoisering* (*top-down*)

5% 8. Hvordan fungerer RELAX(u, v, w)?

Hvis $v.d > u.d + w(u, v)$ har vi funnet en snarvei, og $v.d$ oppdateres. Vi setter da $v.\pi$ til u.

Relevant læringsmål: Forstå *kant-slakking* (*edge relaxation*) og RELAX

5% 9. Hva er forskjellen på konkrete og abstrakte problemer, slik boka definerer dem?

Et abstrakt problem er en vilkårlig binær relasjon mellom instanser og løsninger, mens for konkrete problemer er de gyldige instansene alle binære strenger.

Relevant læringsmål: Forstå forskjellen på *konkrete* og *abstrakte* problemer

5% 10. Hvordan fungerer LIST-DELETE(L, x)?

Enten $x.prev.next$ (hvis den finnes) eller $\mathrm{L}.head$ settes til $x.next$, mens $x.next.prev$ (hvis den finnes) settes til $x.prev$.

Relevant læringsmål: Forstå hvordan *lenkede lister* fungerer (LIST-SEARCH, LIST-INSERT, LIST-DELETE, LIST-DELETE′, LIST-SEARCH′, LIST-INSERT′)

5% 11. Hvordan kan vi si at MERGE-SORT er optimal? Forklar.

Fordi den har kjøretid på $\Theta(n \lg n)$, og noe bedre er umulig i verste tilfelle, for sammenligningsbasert sortring.

Relevant læringsmål: Forstå hvorfor *sammenligningsbasert sortering* har en *worst-case* på $\Omega(n \lg n)$

5% 12. Anta at du har funnet den transitive lukningen (*transitive closure*) $\mathrm{G}^* = (\mathrm{V}, \mathrm{E}^*)$ av grafen $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, i form av en boolsk matrise. Vis hvordan G^* kan oppdateres effektivt dersom en kant $\{u, v\}$ legges til i G. Hva blir kjøretiden?

For alle $\{s, u\}, \{v, t\} \in \mathrm{E}$, legg til $\{s, t\}$, om nødvendig. Kjøretid $\mathrm{O}(\mathrm{V}^2)$.

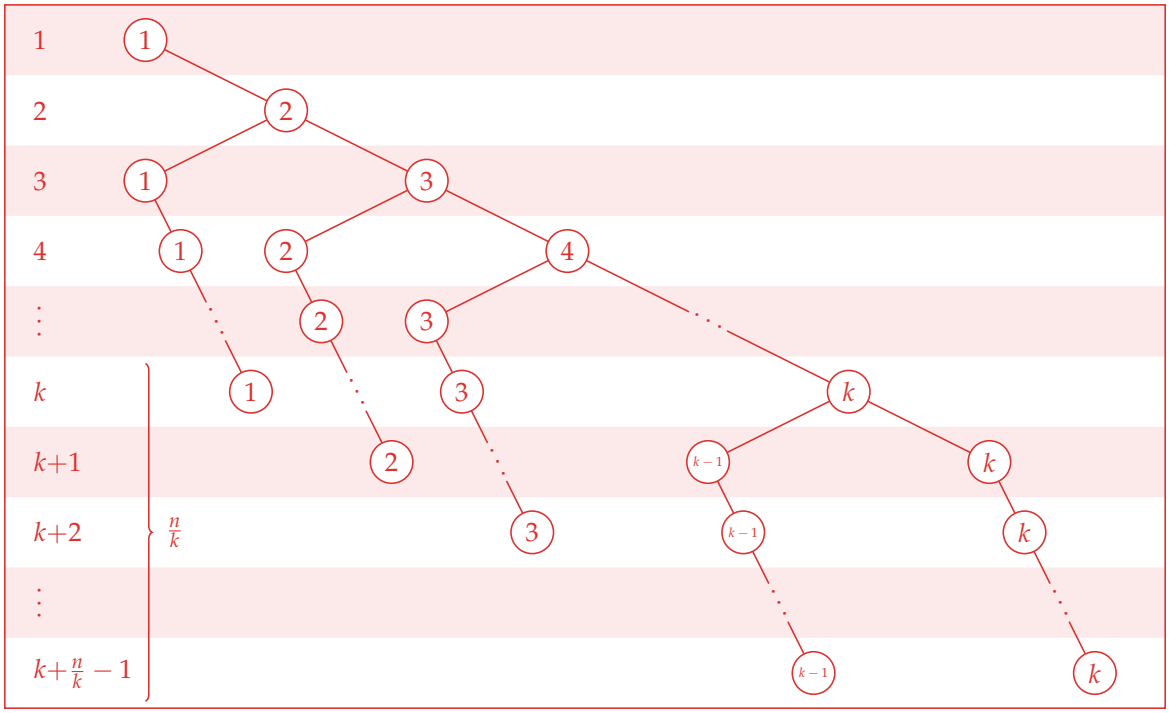
Relevant læringsmål: Forstå TRANSITIVE-CLOSURE. (Basert på oppgave 25-1(a) i læreboka.)

5% 13. Anta at du starter med et tomt binært søketre og setter inn n verdier fra verdiområdet $1, \ldots, k$, der $k \leq n$. Rekkefølgen på verdiene er $1, 2, \ldots, k, 1, 2, \ldots, k, \ldots$, altså verdiene fra 1 til k gjentatte ganger. Hva blir høyden til søketreet? Oppgi svaret som funksjon av n og k. Du kan anta at n er delelig på k. Ikke bruk asymptotisk notasjon. Forklar kort, gjerne med en figur.

(Merk: For å følge bokas implementasjon, skal nodene i venstre deltre være strengt mindre enn rota. Husk at høyden måles i antall kanter, ikke noder.)

Se figur nedenfor. Verdien k forekommer n/k ganger, og ligger på de n/k nederste nivåene. Over dette er det $k - 1$ nivåer. Totalt får vi altså $k + n/k - 1$ nivåer. Høyden blir dermed $k + n/k - 2$.

Relevant læringsmål: Forstå hvordan *binære søketrær* fungerer (INORDER-TREE-WALK, TREE-SEARCH, ITERATIVE-TREE-SEARCH, TREE-MINIMUM, TREE-MAXIMUM, TREE-SUCCESSOR, TREE-PREDECESSOR, TREE-INSERT, TRANSPLANT, TREE-DELETE)



5% 14. Løs følgende rekurrens: $T(n) = 4T(n/2) + 1/\sqrt{n}$. Skriv svaret med Θ-notasjon.

$\Theta(n^2)$

Relevant læringsmål: Kunne løse rekurrenser med *substitusjon*, *rekursjonstrær* og *masterteoremet*

Den såkalte *Ackermannfunksjonen* defineres slik:

$$A(m,n) = \begin{cases} n+1 & \text{hvis } m = 0; \\ A(m-1,1) & \text{hvis } m > 0 \text{ og } n = 0; \\ A(m-1, A(m,n-1)) & \text{hvis } m > 0 \text{ og } n > 0; \end{cases} \tag{1}$$

der m og n er ikke-negative heltall.

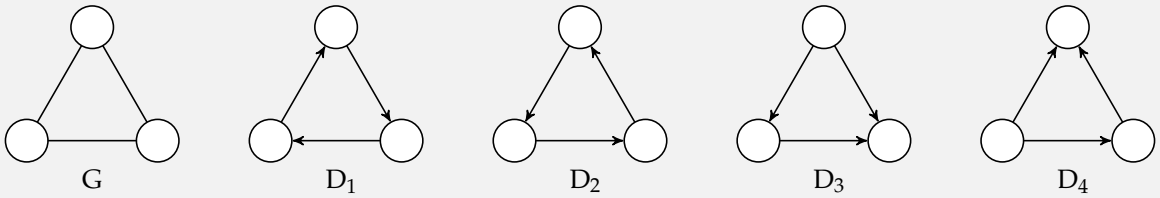
5% 15. Hva er $A(1,n)$ som funksjon av n? Ikke bruk asymptotisk notasjon. Vis utregning.

Dette er bare en lett kamuflert utgave av den veljente rekurrensen $f(n) = f(n-1) + 1$, om man lar $A(1,n) = f(n)$, siden $A(1,n) = A(0,A(1,n-1)) = A(1,n-1) + 1$. Man får $f(0) = A(1,0) = A(0,1) = 2$, så $A(1,n) = f(n) = n+2$. Litt mer detaljert utregning (kreves ikke):

$$\begin{aligned} A(1,n) &= A(0,A(1,n-1)) &&= A(1,n-1) + 1 && (1) \\ &= A(0,A(1,n-2)) &&= A(1,n-2) + 2 && (2) \\ &= A(0,A(1,n-3)) &&= A(1,n-3) + 3 && (3) \\ &\vdots &&\vdots && \vdots \\ &= A(0,A(1,n-n)) &&= A(1,n-n) + n && (n) \\ &= A(0,1) + n &&= n+2 && (*) \end{aligned}$$

Relevant læringsmål: Kunne løse rekurrenser med *substitusjon*, *rekursjonstrær* og *masterteoremet*

En *orientering* av en urettet graf $G = (V, E)$ er en tilordning av retning til hver kant $e \in E$, som resulterer i en ny rettet graf $D = (V, A)$, der hver urettet kant $\{u, v\}$ i E tilsvarer en rettet kant (u, v) eller (v, u) i A, og omvendt. Nedenfor er et eksempel på en urettet graf G og noen orienteringer D_i.



Her er D_3 og D_4 eksempler på *asykliske* orienteringer. En *sterk* orientering er *sterkt sammenhengende* (*strongly connected*, det vil si, fra enhver node u finnes det en rettet sti til enhver annen node v). Eksempler på dette er D_1 og D_2. Disse finnes bare hvis ethvert snitt inneholder minst to kanter. I en *k-orientering* har enhver node v maksimalt $k(v)$ inn-kanter, for en funksjon $k : V \to \mathbb{N}$.

5% 16. Din venn Smartnes prøver å finne én bestemt orientering med visse egenskaper og hun bruker en form for binærsøk: Til å begynne med er alle orienteringer mulige løsninger, men for hver iterasjon halveres antallet kandidater, helt til én gjenstår. Én iterasjon tar lineær tid som funksjon av antall kanter, m. Hva blir den totale kjøretiden? Oppgi svaret med Θ-notasjon. Forklar kort.

Det er 2^m mulige orienteringer, og $\log_2 2^m = m$, så vi får $\Theta(m)$ iterasjoner, og kjøretid $\Theta(m^2)$.

Relevante læringsmål: Kjenne til enkel kombinatorikk, som permutasjoner og kombinasjoner; kunne definere *asymptotisk notasjon*, O, Ω, Θ, o og ω; forstå BISECT og BISECT′.

10% 17. Hvordan kan du effektivt finne asykliske og sterke orienteringer?

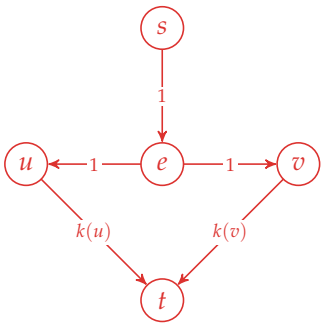
Bruk DFS, og bruk retningen til tre-kantene (*tree edges*). Hver gjenværende kant er mellom en etterkommer og en av stamforeldrene i DFS-treet. Rettes disse nedover i treet, brytes alle sykler, og orienteringen blir asyklisk.

En sterk orientering har vi hvis og bare hvis ethvert par $u, v \in E$ ligger på en sykel. Hvis en slik orientering finnes, holder det om vi bevarer alle urettede sykler som rettede, og det kan vi gjøre ved å gi bakoverkantene retning *oppover* i treet.

Relevante læringsmål: Forstå hvordan DFS *klassifiserer kanter*; kunne konstruere nye effektive algoritmer

10% 18. Hvordan kan du effektivt finne k-orienteringer ved hjelp av maks-flyt?

For eksempel la hver kant «produsere» 1 enhet flyt som kan gå til én av endenodene, og gi hver node v en kapasitet på $k(v)$. Dette gjør du ved å splitte kantene med nye noder koblet til kilden, og å kilde hver opprinnelig node til sluket. For en kant $e = \{u, v\} \in E$ får vi f.eks.:



Relevante læringsmål: Kunne definere *flytnettverk*, *flyt* og *maks-flyt-problemet*; kunne konstruere nye effektive algoritmer