Noen viktige punkter:

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

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

(*iii*) Skriv svarene dine i svarrutene og levér inn oppgavearket. Bruk gjerne blyant! Evt. kladd på eget ark først for å unngå overstrykninger, og for å få en egen kopi.

(*iv*) Ekstra ark kan legges ved om nødvendig, men det er meningen at svarene skal få plass i rutene på oppgavearkene. Lange svar teller ikke positivt.

Eksamen har 20 oppgaver, totalt verdt 120 poeng.
Av disse er 20 bonuspoeng, så en poengsum over 100 regnes som 100.
Poengverdi er angitt ved hver oppgave.

* * *

For hver av de første 5 oppgavene skal du velge svaret blant disse algoritmene, og sette kryss i sirkelen til venstre for rett bokstav. For hver oppgave kan det være snakk om én eller flere algoritmer, så det er tillatt å sette flere kryss.

A Bucket-Sort

B Counting-Sort

C Heapsort

D Insertion-Sort

E Merge-Sort

F Quicksort

G Radix-Sort

H Randomized-Quicksort

I Randomized-Select

J Select

Her gis 5 poeng for rett svar, med 1 poengs trekk for hvert kryss som er satt galt eller mangler.

(5 p) 1. Hvilke av algoritmene på s. 1 er baserte på designmetoden *divide-and-conquer*?

◯ A ◯ B ◯ C ◯ D √ E √ F ◯ G √ H √ I √ J

Her regnes som korrekt både å ta med og å ekskludere C.

(5 p) 2. Hvilke av algoritmene på s. 1 har kjøretid $O(n)$ i *verste tilfelle*, dersom $k = O(n)$ og $d = O(1)$?

◯ A √ B ◯ C ◯ D ◯ E ◯ F √ G ◯ H ◯ I √ J

(5 p) 3. Hvilke av algoritmene på s. 1 bruker subrutinen PARTITION, evt. litt modifisert?

◯ A ◯ B ◯ C ◯ D ◯ E √ F ◯ G √ H √ I √ J

(5 p) 4. Hvilke av algoritmene på s. 1 benytter seg av en prioritetskø?

◯ A ◯ B √ C ◯ D ◯ E ◯ F ◯ G ◯ H ◯ I ◯ J

(5 p) 5. Hvilke av algoritmene på s. 1 endrer kjøretid fra beste til verste tilfelle?

√ A ◯ B √ C √ D ◯ E √ F ◯ G √ H √ I ◯ J

Her kan noen ha tenkt kun på variasjoner styrt av input, og ikke tilfeldighetsgeneratoren, og dermed droppet RANDOMIZED-QUICKSORT og RANDOMIZED-SELECT. Dette stemmer ikke med bokas analyse, men er ikke helt urimelig, så det gir 3 poeng.

(5 p) 6. Hva er løsningen på rekurrensen $T(n) = 3T(n/3) + n$? Oppgi svaret i O-notasjon.

$T(n) = O(n \lg n)$

(5 p) 7. Hva er løsningen på rekurrensen $T(n) = T(n/5) + n$? Oppgi svaret i O-notasjon.

$T(n) = O(n)$

(5 p) 8. Hva er løsningen på rekurrensen $T(n) = 2T(4\sqrt{n}) + \lg n$? Oppgi svaret i O-notasjon.

$T(n) = O(\lg n \lg\lg n)$

For å løse denne må man bruke to teknikker fra boka. Først variabelsubstitusjon (s. 86):

$$\begin{aligned} T(n) &= 2T(4\sqrt{n}) + \lg n \\ T(2^m) &= 2T(4\sqrt{2^m}) + m && [m = \lg n] \\ &= 2T(2^2 \cdot 2^{m/2}) + m \\ &= 2T(2^{m/2+2}) + m \\ S(m) &= 2S(m/2+2) + m && [S(m) = T(2^m) = T(n)] \end{aligned}$$

Så løser man $S(m) = 2S(m/2+2) + m$ med substitusjonsmetoden (som vist nederst på s. 84), og så bruker man definisjonene av S og m til å få løsningen på den opprinnelige rekurrensen:

$$\begin{aligned} T(n) &= S(m) \\ &= O(m \lg m) \\ &= O(\lg n \lg\lg n) \end{aligned}$$

(5 p) 9. Hvis du setter verdiene 1, 2, 9, 5, 10, 7, 6, 4, 8 og 3 inn i et tomt binærtre (én etter én, i oppgitt rekkefølge), hva blir høyden til treet (antall kanter i den lengste stien fra rota til en løvnode)?

5

Merk: Her er det ment et binært *søketre*, som er den eneste tretypen vi har en innsettingsalgoritme for i pensum. Det er ikke rimelig å anta at det er snakk om et vilkårlig binært tre, siden høyden da ikke er entydig.

(5 p) 10. Hvis du setter verdiene 1, 2, 9, 5, 10, 7, 6, 4, 8 og 3 inn i en tom binær min-heap (én etter én, oppgitte rekkefølge), hva blir høyden til heapen (antall kanter i den lengste stien fra rota til en løvnode)?

3
(Høyden her er kun avhengig av antallet verdier, og er $\lfloor \lg n \rfloor = \lfloor \lg 10 \rfloor = 3$.)

(5 p) 11. I en sortert tabell med n elementer forekommer et element x én gang, mens alle andre elementer forekommer to ganger. Beskriv en effektiv algoritme for å finne ut hva x er. Hva blir kjøretiden?

Bruk en form for binærsøk. Se på de tre midterste, og se hvilke av disse som er like, om noen. Videre rekursjon bestemmes av det.
$T(n) = \Theta(\lg n)$.

(5 p) 12. Forklar svært kort hva det vil si at en sorteringsalgoritme er *stabil*.

Den bytter ikke om på like elementer.

(5 p) 13. Du kjører DFS på en rettet graf, og får klassifisert alle kantene. Hvordan kan du bruke denne klassifiseringen til å avgjøre om grafen har en sykel?

Se om det finnes bakoverkanter (*back edges*).

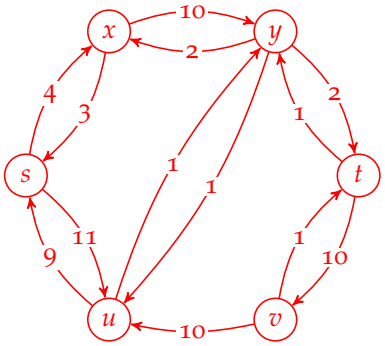
(5 p) 14. Du har en binær matrise som angir om det går en veistrekning direkte mellom ulike par av byer. Du vil lage en ny matrise som angir om det går en direkte *eller indirekte* vei mellom hvert par, det vil si, hvis du får lov til å kjøre innom andre byer. Anta at det er snakk om n byer. Hvordan vil du løse problemet hvis det er $\Theta(n)$ direkte veistrekninger? Hva blir kjøretiden?

BFS eller DFS fra alle noder for å se hvem vi når. $T(n) = O(n^2)$.

(5 p) 15. Hvordan vil du løse problemet i oppgave 14 om det er $\Theta(n^2)$ direkte veistrekninger? Hva blir kjøretiden?

Bruk Transitive-Closure. $T(n) = \Theta(n^3)$. Evt. bruk samme løsning som i forrige oppgave.
Man kan evt. bruke Floyd-Warshall, og se om avstandene blir endelige.

(5 p) 16. Tegn residualnettverket til flytnettverket som er tegnet i figur 1 (se s. 4).



Merk: Flytnettverket i figur 1 er feil! (Det går mer flyt inn til x enn ut av x, som ikke er tillatt). Å påpeke dette gir 1 bonuspoeng (men oppgaven gir fortsatt maksimalt 5 poeng). $f(s, x)$ var ment å være 2. Om man endrer dette og forklarer hvorfor, så vil et korrekt flytnettverk for det korrigerte tilfellet fortsatt gi full uttelling.

(10 p) 17. Din venn Lurvik har funnet opp en ny algoritme som skal lage spenntrær for sammenhengende, uvektede, urettede grafer. Hun starter med en vilkårlig kant. Hun går så gjennom alle kantene igjen og igjen. I hver iterasjon legger hun til kanter som har nøyaktig én nabokant i spenntreet. (Kantene legges til én og én. For hver kant som legges til må man også ta hensyn til kantene som har blitt lagt til tidligere i samme iterasjon.) Er algoritmen korrekt? Forklar svært kort. Hvis grafen har n noder og m kanter, hva blir kjøretiden i verste tilfelle?

(**Merk:** Grafen er *uvektet*, så det er *ikke* snakk om å finne et *minimalt* spenntre, men et *vilkårlig* et.)

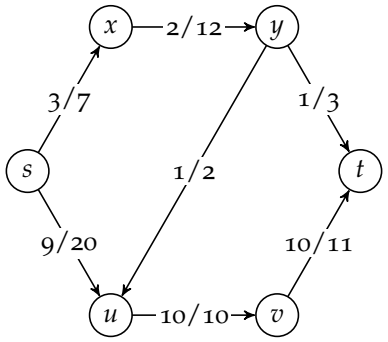


Figure 1: Flytnettverk til oppgave 16.

Algoritmen er feil, noe som enkelt vises ved et moteksempel (som f.eks. ⋖). Algoritmen vil kun fungere dersom grafen har en Hamilton-sti og vi tilfeldigvis finner den.

Om man tolker oppgaven som at kanten har nøyaktig én nabonode som er tilkoblet treet med én eller flere kanter, så er algoritmen korrekt: Den vil aldri danne sykler, og alle noder vil etter hvert bli koblet til treet. (Kan evt. vises ved induksjon.) Denne tolkningen er ikke korrekt, men ble oppgitt til noen som spurte på eksamen, og gis dermed full uttelling, så lenge svaret forklares.

Hver iterasjon tar $\Theta(m)$ tid. Alle stier fra start-kanten øker med 1 i hver iterasjon, så det kan maksimalt være $\Theta(n)$ iterasjoner. Kjøretiden blir altså $\Theta(mn)$ i verste tilfelle.

(10 p) 18. Anta at klassene P og NP alt er definerte (dvs., du trenger ikke gi definisjoner for dem). Skissér svært kort definisjonen av NPC.

Problemene i NP som er slik at alle problemer i NP kan reduseres til dem i polynomisk tid.

(Her har flere tolket «skissér» som «tegn», heller å beskrive kortfattet. En tegning som gir uttrykk for definisjonen gir full uttelling, men det holder ikke med et Venn-diagram med NP og NP-hard, siden NP-hard ikke er oppgitt som definert i oppgaven.)

(10 p) 19. Du har en rad $v_1, \ldots, v_n$ med mynter, der n er et partall. Du og en motspiller skal, annenhver gang, forsyne dere med enten den første eller siste mynten av dem som er igjen. Beskriv en algoritme som finner ut hvor mye du kan være *garantert* å vinne, dersom du begynner.

Bruk dynamisk programmering. F.eks. lag en memoisert, rekursiv algoritme som prøver å fjerne både første og siste element, både for deg og motstanderen. I dine runder, velg maksimum av de to resultatene (siden du kan velge mellom dem); i motstanderens, velg minimum (for sikkerhets skyld, siden du ikke vet hva motstanderen vil velge).

Merk: Det er ikke sagt noe om at myntene har samme verdi, og dette er heller ikke en rimelig antagelse, siden gevinsten da aldri vil variere for en gitt n.

(10 p) 20. Du har oppgitt en graf der hver node er *rød* eller *grønn*, og det er ingen kanter mellom noder av samme farge. En grønn node har 0, 1 eller 2 naboer, mens de røde kan ha vilkårlig mange. Hver grønne node har også en positiv vekt. Du skal slette grønne noder, men den resulterende grafen må fortsatt ha stier mellom alle røde noder. Du ønsker å slette et sett med så høy totalvekt som mulig. Beskriv en algoritme som løser problemet, og forklar kort hvorfor svaret ditt blir riktig.

Slett først alle med 0 eller 1 nabo, siden de ikke påvirker stiene mellom røde. Bytt ut gjenværende grønne nodene med kanter. Du kan bare slette kanter til du sitter igjen med et spenntre. I stedet for å slette maksimal vektsum kan du bevare minimal vektsum, og det løser du ved å finne et minimalt spenntre, f.eks. med Prim eller Kruskal.