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 100 poeng. Poengverdi er angitt ved hver oppgave.

* * *

For hver av de første 5 oppgavene skal du velge svaret blant disse algoritmene:

- BFS
- Bellman-Ford
- DFS
- Dag-Shortest-Path
- Dijkstra
- Faster-All-Pairs-Shortest-Paths
- Floyd-Warshall
- Slow-All-Pairs-Shortest-Paths

Merk: Sensor har anledning til å gi (evt. delvis) uttelling for andre svar enn det som er oppgitt, om det f.eks. dreier seg om små feil, eller alternative fullgode svar.

(5 p) 1. Hvilken algoritme på side 1 tillater ikke sykler?

DAG-Shortest-Path

(5 p) 2. Hvilken algoritme på side 1 tillater vilkårlige positive men ikke negative kantvekter?

Dijkstra

(5 p) 3. Hvilken algoritme på side 1 vil ikke nødvendigvis finne korteste vei i en rettet asyklisk graf der alle kantvekter er 1?

DFS

(5 p) 4. Hvilken algoritme på side 1 bør du bruke for å finne korteste vei fra alle til alle i en vilkårlig graf med positive kantvekter, dersom du har $\Theta(V^2)$ kanter?

Floyd-Warshall eller Dijkstra

Her gis altså full uttelling for Dijkstra, siden svaret er korrekt om man kjører den fra hver node (en rimelig lesning av oppgaveteksten), og om man bruker en usortert tabell eller Fibonacci-heap som prioritetskø. (Om man bruker en binær-heap, som var den opprinnelige tanken bak oppgaven, så vil det ikke lønne seg å bruke Dijkstra.) Strengt tatt vil nok Floyd-Warshall være mer effektiv i dette tilfellet uansett, men det var altså ikke poenget med oppgaven. Faster-All-Pairs-Shortest-Path gir 1 poeng.

(5 p) 5. Hvilken algoritme på side 1 bør du bruke for å finne en flytforøkende sti (*augmenting path*)?

BFS

Her gis også 3 poeng for DFS, siden den *kan* brukes, selv om den generelt ikke *bør* brukes.

(5 p) 6. I maskin-modellen til læreboka har heltall normalt $c \lg n$ bits. Hva er kravet til c?

$c \geq 1$

Om man skriver at n må kunne beskrives med $c \lg n$ bits, så gir det også full uttelling.

Enkelte har oppgitt at c må være et heltall større enn 0, evt. "et positivt heltall" (og dermed større eller lik 1). Dette er et strengere krav enn boka (som ikke krever at c må være et heltall), men dette svaret gis 4 poeng. $c > 1$ gis også 4 poeng. Om man bare har svart $c > 0$, gis det 1 poeng.

(5 p) 7. Du har funnet *best-case*-kjøretiden for en algoritme. Hvilken av $\mathcal{O}$, Ω eller Θ vil du bruke for å beskrive den?

Θ eller Ω

Denne oppgaven er ikke tydelig nok formulert, og tas dermed ut av sensur der det gir en høyere skalert poengsum. Slik oppgaven var ment, skulle man beskrive *best-case*-kjøretiden, som man hadde et uttrykk for, med asymptotisk notasjon. Det mest presise ville da være Θ. Flere tolket det derimot som at man skulle beskrive *algoritmen*. Gitt at man kun har *best-case*-kjøretiden, vil det mest presise da være Ω.

Enkelte har forstått oppgaven som et "lurespørsmål," basert på en misforståelse av ideen om at alle notasjoner kan brukes til å beskrive alle cases. Det stemmer generelt, men her er det likevel gitt hva som er mest gunstig/presist. Andre har svart at man ikke kan velge, siden det kommer an på om man også har funnet en øvre grense. Siden man ble bedt om å velge én av de tre, gis ikke dette poeng, men oppgaven tas ut i slike tilfeller, og vil dermed ikke trekke ned.

(5 p) 8. Hva er *worst-case*-kjøretiden til Merge-Sort?

$\Theta(n \log n)$

Her gis det også full uttelling for $\mathrm{O}(n \log n)$, selv om det er mindre presist. Andre logaritmer gir også full uttelling. Ikke-asymptotiske svar (som $n \log n$) gir 4 poeng.

(5 p) 9. Hva har BELLMAN-FORD oppdaget dersom den returnerer FALSE?

Negative sykler.

(5 p) 10. La $A = [3, 3, 1, 4, 4, 3, 1, 2, 3, 5]$. Hvordan ser $C[0\,..\,6]$ ut idet COUNTING-SORT$(A, B, 6)$ returnerer?

$[0, 0, 2, 3, 7, 9, 10]$

Tellesortering teller først forekomster, gjør så tellingene kumulative, og så dekrementerer disse under innsetting. Så tellingene til slutt tilsvarer altså hvor mange som er strengt mindre. F.eks. vil $C[3]$ hvor mange verdier som er mindre enn 3. Her gis også enkelte andre svar noe uttelling. For eksempel:

$[0, 0, 2, 3, 7, 9, 0]$	4 poeng	Avglemt kumulativ verdi for 6
$[0, 2, 3, 7, 9, 10, 10]$	4 poeng	Før innsetting
$[0, 2, 3, 7, 9, 10, 0]$	3 poeng	Før innsetting, avglemt 6
$[0, 2, 1, 4, 2, 1, 0]$	3 poeng	Ikke-kumulativ telletabell

Ett poeng mindre gis for hvert av alternativene om man kun har med de 6 første elementene (dvs., man har bare brukt $C[0\,...\,5]$).

(5 p) 11. La $A = [0, 9, 2, 8, 1, 5, 3, 4, 7, 6]$. Anta at PARTITION velger siste element som *pivot* (som i læreboka). Hvilken indeks returnerer da PARTITION$(A, 1, 10)$?

6 eller 7

PARTITION returnerer indeksen til pivot (altså 6) i sortert rekkefølge. Selv om det går an å resonnere seg frem til at det brukes 1 indeksering (pga. grensene som brukes) så gis det også full uttelling for det 0-indekserte svaret.

(5 p) 12. La $G = (V, E)$ være en graf med noder $V = \{0\,..\,9\}$ og med positive kantvekter w. La $D[0\,..\,9]$ være en tabell med avstandsestimater, der $D[u] = u.d$, for $u = 0\,..\,9$. Etter at DIJKSTRA$(G, w, 0)$ er utført er $D = [0, 2, 3, 5, 8, 6, 9, 1, 7, 4]$. I hvilken rekkefølge har nodene $0\,..\,9$ blitt valgt ut og besøkt?

$0, 7, 1, 2, 9, 3, 5, 8, 4, 6$ eller $1, 3, 4, 6, 9, 7, 10, 2, 8, 5$

Evt. uten startnoden:

$7, 1, 2, 9, 3, 5, 8, 4, 6$ eller $2, 3, 5, 8, 6, 9, 1, 7, 4$

Her er nøkkelen at DIJKSTRA besøker nodene i stigende avstandsrekkefølge. Meningen var at man skulle liste opp nodene i riktig rekkefølge, men om man i stedet oppgir nummeret i rekkefølgen for hver node så gir det også full uttelling.

(5 p) 13. La $C = \{a, b, \ldots, j\}$, der $a.freq = 11, b.freq = 12, \ldots, j.freq = 20$. Utfør HUFFMAN$(C)$. Anta, som i boka, at venstre barn er mindre enn høyre, og at venstre barn er 0. Hva blir Huffman-koden for g?

010

Her gis 4 poeng for 110.

(5 p) 14. La $A = [5, 0, 2, 7, 3, 9, 1, 6, 8, 4]$. Utfør BUILD-MAX-HEAP$(A)$. Hvordan ser A ut etterpå?

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

Om man har byttet om på to verdier, gis det 4 poeng. Her har enkelte brukt MAX-HEAP-INSERT gjentatte ganger. Det vil ha dårligere kjøretid, og er ikke det det er spurt om, men gir likevel 3 poeng. Resultatet blir da $[9, 8, 7, 6, 4, 2, 1, 0, 5, 3]$. Har man byttet om på to verdier her, gis det 2 poeng.

(5 p) 15. La $A = [0, 1, 2, 3, 6, 4, 9, 8, 5, 7]$. Utfør EXTRACT-MIN$(A)$. Hvordan ser A ut etterpå?

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

Har man byttet om på to verdier gis det 4 poeng.

Det første alternativet gjengir hele tabellen, mens det andre kun gjengir selve haugen. Det tredje alternativet er ikke helt korrekt, men gjenspeiler et trinn av HEAPSORT. Her kunne man ha fått trekk, men det har her blitt gitt full uttelling.

Her er det altså meningen at man skal bruke heap-versjonen av EXTRACT-MIN (s. 162–163). Siden det ikke eksplisitt sto HEAP-EXTRACT-MIN har enkelte lurt på om det kunne være en annen implementasjon. Den eneste andre implementasjonen i pensum er en usortert array. Om man har utført denne effektivt, og fått resultatet $[7, 1, 2, 3, 6, 4, 9, 8, 5]$, gis det 1 poeng.

(5 p) 16. La $T(0) = 0$ og $T(n) = T(n-1) + 2^n + n$ når $n > 0$. Løs rekurrensen. Bruk asymptotsk notasjon.

$T(n) = \Theta(2^n)$ Her gis også full uttelling for $O(2^n)$

(5 p) 17. La $T(0) = 0$ og $T(n) = \pi^2 T(n/\pi) + n^2$ når $n > 0$, der $\pi = 3.14159\ldots$.

Løs rekurrensen. Bruk asymptotisk notasjon.

$T(n) = \Theta(n^2 \lg n)$

Her gis også full uttelling for $O(n^2 \lg n)$. Bruk f.eks. masterteoremet, med $\log_\pi \pi^2 = 2$. Merk: Grensetilfellet $T(0) = 0$ er ikke egentlig til nytte, og kan ignoreres. Siden svaret er asymptotisk kan man uansett basere seg på den generelle antagelsen fra pensum om at $T(k) = \Theta(1)$ for en konstant k.

(5 p) 18. ~~Du har funnet en reduksjon med polynomisk kjøretid fra A til B, der A og B er beslutningsproblemer. En optimal algoritme for B har *worst-case*-kjøretid $\Theta(2^n)$. Kan du si noe om kjøretiden til algoritmer for A? Hvis ja, beskriv kjøretiden med asymptotisk notasjon.~~

Den tilsiktede tolkningen var at, ja, man *kan* si noe, nemlig at det finnes algoritmer med kjøretid $O(2^n)$ for A. Dette er analogt til det man kunne si om A dersom B var i P, for eksempel. (Man ville da vite at A kunne løses i polynomisk tid.)

Men oppgaveformuleringen er ikke tydelig nok. Man kan også tolke spørsmålet som at man må si noe om *alle* algoritmer for A, og her har vi ingenting å gå på (man kan jo lett lage algoritmer som er *verre*), så svaret må bli nei.

Enkelte har da også svart "Nei, fordi vi reduserer feil vei." Det var egentlig ment å være den misforståelsen oppgaven bl.a. skulle teste, men under den alternative tolkningen av oppgaven er dette faktisk korrekt. For om man hadde redusert fra B til A, så *kunne* man si noe om alle algoritmer for A, nemlig at de har en *worst-case* på $\Omega(2^n)$.

Antall poeng tildelt blir dermed mindre avhengig av hovedinnholdet i oppgaven, og mer basert på detaljer i svaret, på en såpass lite informativ måte at oppgaven tas ut av sensur.

(5 p) 19. Du har et minneområde på n bytes, og skal dele det opp i segmenter, som vist i figur 1 på side 6. Et segment med lengde k bytes har en *kostnad* på c_k, og du vil finne en oppdeling som er slik at den totale kostnaden $f(n)$ blir minst mulig. Skriv en rekursiv ligning som gir den optimale verdien for $f(n)$. Løsningen skal egne seg for memoisering. Du kan anta $f(0) = 0$.

$f(n) = \min_{1 \le i \le n} c_i + f(n-i)$ Essensielt ekvivalent med (15.2) i læreboka

Her er det også korrekt med en mer *divide-and-conquer*-aktig oppdeling, der man har *to* rekursive forekomster av f. F.eks.:

$f(n) = \min\{f(i) + f(n-i) : i = 1 \ldots \lfloor n/2 \rfloor\} \cup \{c_n\}$

Dette gis full uttelling. Andre varianter kan også gis full eller delvis uttelling, etter skjønn.

(5 p) 20. Som i *the 0-1 knapsack problem* har en tyv gjort innbrudd i en butikk og funnet n gjenstander. Hun gir hver gjenstand i en positiv eller negativ verdi v_i, basert på hvor tung den er og hvor mye hun kan selge den for. Hun ønsker å velge ut en delmengde S av gjenstander slik at $\sum_{i \in S} v_i$ blir størst mulig.

Men: Noen av gjenstandene er avhengige av andre. For eksempel kan hun ikke ta med det antikke sverdet uten å også ta med den antikke sverdsliren. Hun kan bare velge ut en delmengde der slike avhengigheter er ivaretatt: Hvis i er avhengig av j og hun vil ta med i så *må* hun også ta med j. (Se figur 2 for et eksempel.)

Beskriv en algoritme som løser problemet effektivt. Hold forklaringen så kort og enkel som mulig. Tegn gjerne en figur.

Løses som et *minimum cut*-problem, ved hjelp av f.eks. EDMONDS-KARP.

La gjenstander være noder, og legg til s og t.

Hvis i er avhengig av j, legg til en kant (i, j) med $c(i, j) = \infty$.

Hvis $v_i > 0$, legg til en kant (s, i) med $c(s, i) = v_i$.

Hvis $v_i < 0$, legg til en kant (i, t) med $c(i, t) = -v_i$.

Forklaring (kreves ikke av en fullgod løsning): Vi tenker altså at den utvalgte delmengden blir S i et minimalt (S, T)-snitt. Se læreboka, 720. Vi må da sørge for at et minimalt snitt gir oss det vi vil.

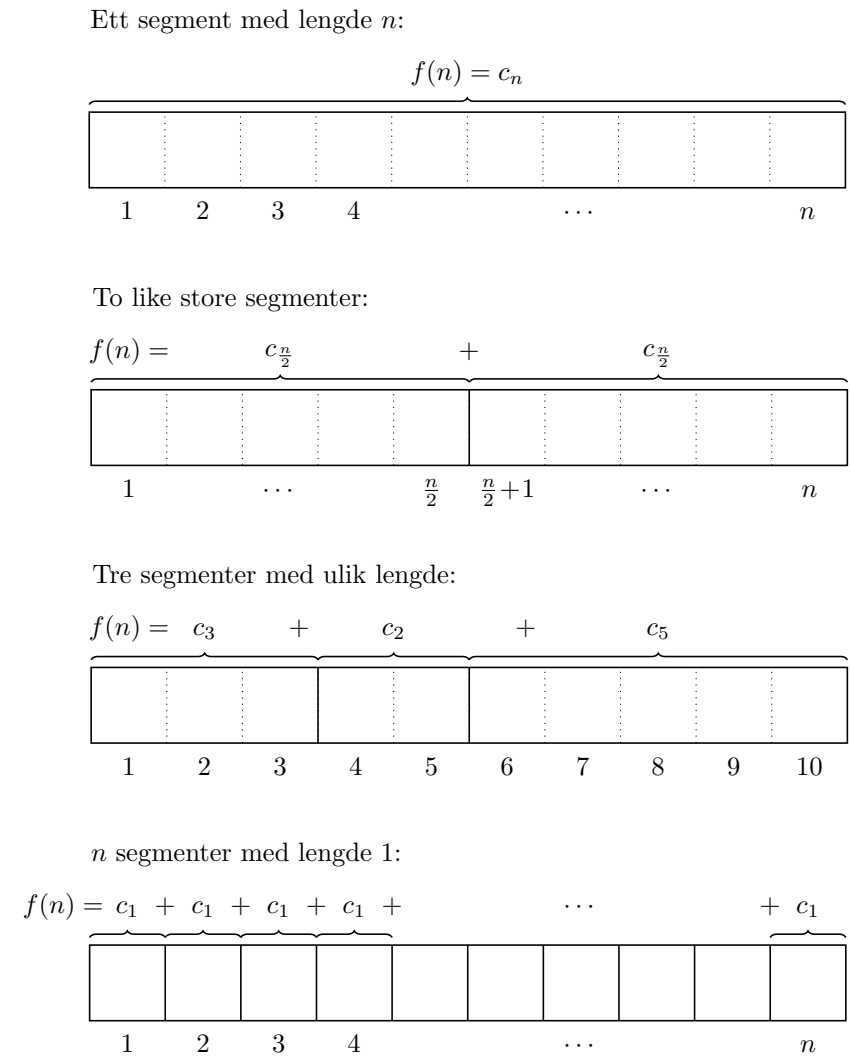
Hvis $i \in S$ og $j \in T$ og i er avhengig av j blir kostnaden uendelig, så dette vil vi automatisk unngå dersom det er mulig.

Hvis vi lar en verdifull gjenstand i være igjen i butikken (dvs., $i \in T$), så taper vi v_i. Hvis vi tar den med ($i \in S$), taper vi ingenting.

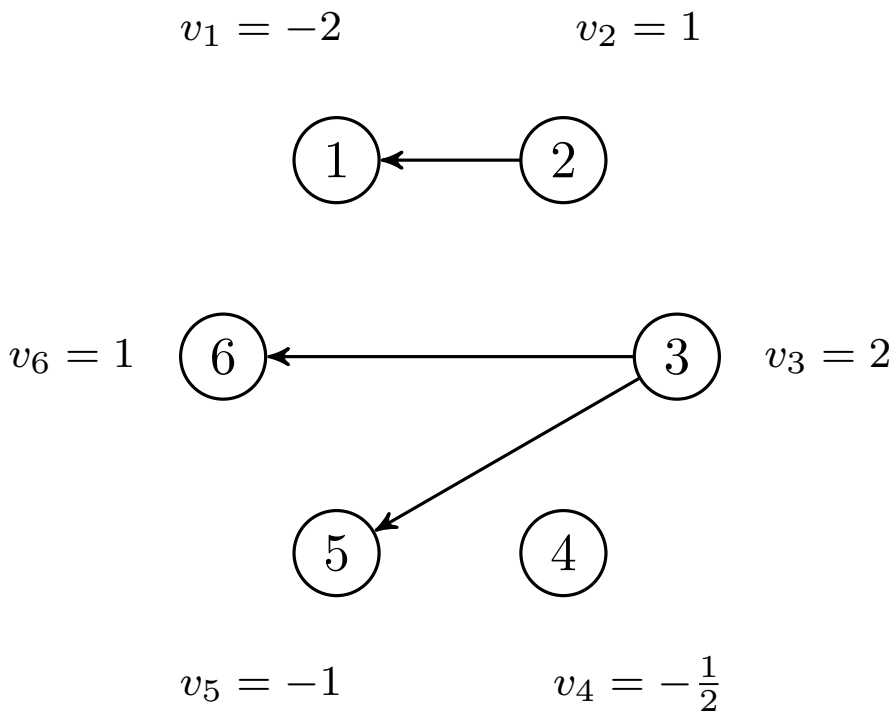
Hvis vi tar med oss en verdiløs gjenstand i (dvs., $i \in S$, der $v_i < 0$), så koster det oss $-v_i$. Dersom vi lar den være igjen ($i \in T$) så koster det oss ingenting.

Et minimalt snitt vil altså gi oss en lovlig delmengde (der avhengighetene respekteres), der vi taper minst mulig, dvs., der vi får med oss gjenstander med størst mulig totalverdi.

Noen har gitt en alternativ løsning der man først finner sterke komponenter (SCC-er), som må inkluderers eller ekskluderes som helhet, og får en verdi som er lik summen av de opprinnelige verdiene i komponenten. Man sitter da igjen med en asyklisk utgave av problemet. Her kunne man bruke dynamisk programmering (DP) for å finne et optimalt subsett. For de k første komponentene, i omvendt topologisk sortert rekkefølge, kan man finne en optimal verdi ved å vurdere om man vil ha med komponent k eller ikke. Hvis ikke, kan man bruke optimal verdi frem til $k - 1$. Om man vil ha med k, bruker man optimal verdi for $k - 1$, men legger også til alle komponenter som k er avhengig av. For å få til dette, må man også ha lagret hvilke objekter som allerede var med i den optimale løsningen for $k - 1$ (f.eks. i form av en bit-vektor av lengde k). Å få denne algoritmen helt korrekt er utfordrende, men hoved-ideene med å finne SCC-er, og deretter bruke DP er såpass god at man selv ved å skissere hovedtrekkene får 4 poeng. Selv varianter som ikke bruker DP til slutt, men enklere (og ikke helt korrekte) løsningsmetoder, kan få 3 eller 4 poeng.



Figur 1: Illustrasjon til oppgave 19. Minneområdet kan deles inn i alt fra ett til n segmenter av lik eller ulik lengde. Et segment med lengde k har kostnad c_k, og den totale løsningen har kostnad $f(n)$, som er summen av kostnadene til segmentene. Du skal prøve å finne en oppdeling slik at denne totalsummen av kostnader blir minst mulig.



Figur 2: Illustrasjon til oppgave 20. Innbrudstyven vil i dette eksemplet velge en delmengde av $\{1, \ldots, 6\}$. Hvert gjenstand har en pil til hver av gjenstandene den er avhengig av. For eksempel: Hvis du vil ta med gjenstand 2 så må du også ta med gjenstand 1. Totalt sett bidrar disse to gjenstandene med en verdi på -1, så det lønner seg ikke. Gjenstand 6 kan du ta med eller ikke uten hensyn til andre gjenstander. Den har en positiv verdi, så det lønner seg å ta den med. Hvis du vil ta med gjenstand 3 må du ta med gjenstand 5 og gjenstand 6. Selv om v_5 er negativ så vil dette lønne seg (heller enn å bare ta med gjenstand 6), siden $v_3 + v_5 > 0$. Gjenstand 4 er det ingen vits i å ta med seg.