# TDT4120 Algoritmer og datastrukturer

Eksamen, 15. desember 2025, 15:00–19:00

| | |
|---|---|
| **Faglig kontakt** | Magnus Lie Hetland |
| **Hjelpemiddelkode** | E |

## Løsningsforslag

Løsningsforslagene i rødt nedenfor er *eksempler* på svar som vil gi uttelling. Det vil ofte være helt akseptabelt med mange andre, beslektede svar, spesielt der det bes om en forklaring eller lignende. Om du svarte noe litt annet, betyr ikke det nødvendigvis at du svarte feil!

**Merk:** Det å forstå hva det spørres etter *er en del av oppgaven*. Enkelte oppgaver kan være konstruert slik at det er nettopp dette som er utfordringen, så om du ikke får dem til, vil de kunne virke uklare eller tvetydige.

5 % **1** Hva er kjøretiden til prosedyren for å bygge en maks-haug, BUILD-MAX-HEAP?

> $\Theta(n)$.
>
> Her gis også full uttelling for $O(n)$.

5 % **2** Din venn Lurvik har klart å vise at *to* av følgende beskrivelser av kjøretiden til en algoritme er korrekte, *men han husker ikke hvilke to*:

1. $T(n) = O(n^3)$
2. $T(n) = \Omega(n^3)$
3. $T(n) = \Theta(n^3)$

Sjefen hans har bedt ham velge ut én av beskrivelsene, og nå har Lurvik bedt deg om hjelp. Hvilken av de tre velger du (nummer 1, 2 eller 3)? Forklar kort hvorfor dette er et godt valg.

> Nummer 3, fordi $\Theta$-notasjon er mest informativt. Om Lurvik har vist to av disse, må han enten ha vist at nummer 3 stemmer, eller at både 1 og 2 stemmer, som impliserer nummer 3.

5 % **3** Hva er det som gjør at COUNTING-SORT har lavere asymptotisk kjøretid enn f.eks. MERGE-SORT? Forklar kort.

At vi kun sorterer heltall i området 0 til $k$, for en konstant $k$.

Som vanlig, aksepteres alle forklaringer som får frem hovedpoenget.

5 % **4** Lurvik og Smartnes har en heftig diskusjon. Smartnes mener man kan ha en algoritme med ulik asymptotisk kjøretid i beste og verste tilfelle, der den gjennomsnittlige kjøretiden likevel er lik én av de to. Lurvik mener dette er umulig, siden gjennomsnittet $y$ av $x$ og $z$ alltid vil ligge imellom dem, altså $x < y < z$.

Bruk et eksempel fra pensum for å vise at Smartnes har rett. Forklar kort hvorfor det er mulig.

Smartnes mener altså at *average-case* kan være lik enten *best-case* eller *worst-case*, selv om *best-case* og *worst-case* er forskjellige.

Insertion-Sort har kjøretid $\Theta(n)$ i beste tilfelle og $\Theta(n^2)$ i verste tilfelle. Gjennomsnittlig kjøretid er $\Theta(n^2)$. Grunnen til at dette er mulig er at den asymptotiske notasjonen skjuler forskjeller i konstantfaktorene.

Som vanlig, godtas alle eksempler og forklaringer som får frem hovedpoenget. F.eks. kan man bruke Quicksort, Randomized-Quicksort, Randomized-Select eller søk i tilfeldige søketrær.

5 % **5** Dette er den rekursive formuleringen av lengden til den lengste felles delsekvensen (*longest common subsequence*, LCS) for prefiksene $X_i$ og $Y_j$ av X og Y.

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0\,, \\ c[i-1,j-1]+1 & \text{if } i,j > 0 \text{ and } x_i = y_j\,, \\ \max\{c[i,j-1], c[i-1,j]\} & \text{if } i,j > 0 \text{ and } x_i \neq y_j\,. \end{cases}$$

Hva er det som mangler? Forklar kort.

Hvis de to siste bokstavene er like, vil vi aldri tape på å ta dem med i løsningen, som øker lengden med 1. I tillegg kommer svaret for $X_{i-1}$ og $Y_{j-1}$.

5 % **6** Felles for følgende algoritmer er at hver node $v$ har et bestemt felt av typen $v.x$ (men med et annet navn), som representerer en del av løsningen:

- DFS-Visit (altså dybde-først-søk fra én node)
- BFS
- Dag-Shortest-Paths
- Dijkstra
- Bellman-Ford

- Prim

Hvilket felt er det snakk om (dvs., hva skal stå i stedet for «$x$» i «$v.x$»)? Hva representerer det? Forklar kort.

Det er altså snakk om det samme feltet i alle algoritmene.

> Det er snakk *forgjenger*-feltet, $\pi$, som angir foreldrenoden i treet som konstrueres (traverserings-tre, korteste-vei-tre eller minimalt spenntre).

5 % **7** I skog-implementasjonen av disjunkte mengder (*disjoint-set forests*), som brukt bl.a. i Kruskal, hva skjer med foreldrepekerne når man bruker Find-Set?

> Alle foreldrepekerne til nodene langs stien opp til rota flyttes til å peke direkte på rota (såkalt stikomprimering, eller *path compression*).

5 % **8** Dette er definisjonen av restkapasiteten (*residual capacity*) $c_f(u,v)$ i et flytnett (*flow network*) med flyt $f$:

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in \mathrm{E}\,, \\ f(v,u) & \text{if } (v,u) \in \mathrm{E}\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Hva er det som mangler? Forklar kort.

> Vi kan øke flyten fra $f(u,v)$ til $c(u,v)$ langs $(u,v)$, eller vi kan *oppheve* flyten $f(v,u)$ ved å sende den tilbake fra $u$ til $v$.

5 % **9** Du studerer et problem P, og vil sammenligne det med et annet, kjent problem Q ved å finne en effektiv reduksjon. Hvilken vei vil du redusere for å vise hva? Forklar kort.

> Om Q er vanskelig, vil vi prøve å redusere fra Q til P, for å vise at P også er vanskelig. Om Q er lett, vil vi redusere fra P til Q, for å vise at P også er lett.
>
> Her kan man godt bruke polynomisk kjøretid eller P vs. NPC for å beskrive vanskegrad, men det er ikke nødvendig.

5 % **10** Hvordan kan antall sterke komponenter (*strongly connected components*) i en rettet graf endres hvis man legger til en ny kant? Forklar kort.

Om man har $n$ noder, kan man gå fra $n$ sterke komponenter til 1. Det skjer f.eks. om grafen er en rettet sti, og man legger til kanten som skal til for å gjøre den til en rettet sykel.

Dette er oppgave 20.5-1 fra læreboka.

5 % **11** Hvorfor gir ikke DIJKSTRA nødvendigvis riktig svar dersom grafen vi bruker den på inneholder kanter med negativ vekt? Forklar kort.

Her må forklaringen være mer presis enn f.eks. at «valgene algoritmen gjør blir gale». Hvorfor blir de gale? Bruk gjerne et eksempel, om du har behov for det.

Vi risikerer da at noden med lavest avstandsestimat fortsatt kan få *enda lavere* avstandsestimat, etter at vi har besøkt andre noder.

Som vanlig, godtas alle forklaringer som får frem hovedpoenget.

5 % **12** Hvor mye øker $m$ om du utfører følgende prosedyre?

```
1  for i = 1 to n
2      m = m + 1
3      for j = 1 to n
4          m = m + 1
5          for k = 1 to n
6              m = m + 1
```

Oppgi svaret i $\Theta$-notasjon, som funksjon av $n$. Forklar kort.

$\Theta(n^3)$. Linje 2, 4 og 6 øker $m$ med henholdsvis $n$, $n^2$ og $n^3$.

5 % **13** Hvor mye øker $m$ om du utfører følgende prosedyre?

```
1  k = 1
2  for i = 1 to n
3      m = m + k
4      k = k + k
```

Oppgi svaret i $\Theta$-notasjon, som funksjon av $n$. Forklar kort.

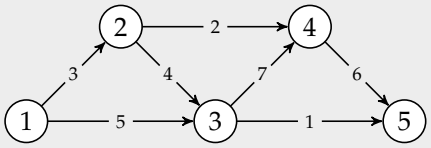
$\Theta(2^n)$. Linje 4 dobler $k$ for hver iterasjon, så linje 3 øker $m$ med $2^i$. Totalt økes $m$ med $\sum_{i=1}^{n} 2^n = 2^{n+1} - 2 = \Theta(2^n)$.

5 % **14** Løs følgende rekurrens eksakt:

$$T(n) = T(n-1) + 2n - 1$$

$$T(0) = 0$$

**Figur 1**




Oppgi svaret uten asymptotisk notasjon. Forklar kort.

$$
\begin{aligned}
T(n) &= (2n-1) + T(n-1) \\
&= (2n-1) + \big(2(n-1)-1\big) + T(n-2) \\
&= (2n-1) + \big(2(n-1)-1\big) + \big(2(n-2)-1\big) + \cdots + \big(2\cdot 1 - 1\big) \\
&= 2\big(n + (n-1) + \cdots + 2 + 1\big) - n \\
&= 2\big(1 + 2 + \cdots + n\big) - n \\
&= 2\big(n(n+1)/2\big) - n = n(n+1) - n = n^2
\end{aligned}
$$

Her godtas også enklere utregninger eller forklaringer.

5 % **15** Hva blir $T(n)$, om du løser følgende rekursive ligningssett?

$T(n) = 2S(n/2) + (n \lg n)^2$

$S(n) = 8T(n/2) + n^2$

Oppgi svaret i $\Theta$-notasjon. Forklar kort.

Bruk vanlige antagelser for algoritmiske rekurrenser.

$$
\begin{aligned}
T(n) &= 2\big(8T((n/2)/2) + n^2\big) + (n \lg n)^2 \\
&= 16T(n/4) + n^2 \lg^2 n + n^2
\end{aligned}
$$

Løses f.eks. med masterteoremet, som gir:

$T(n) = \Theta(n^2 \lg^3 n)$

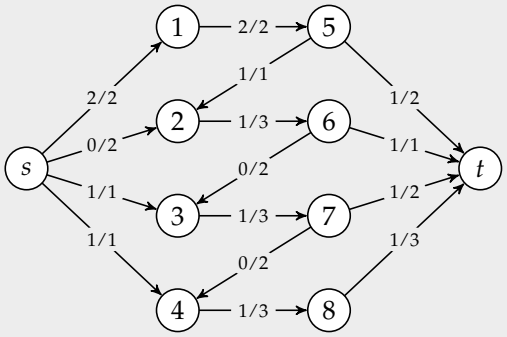
5 % **16** Utfør den første av de $|V| - 1$ iterasjonene til Bellman-Ford på grafen i figur 1, under følgende betingelser:

1. Bruk node 1 som startnode.
2. Anta at 8 er brukt i stedet for $\infty$ under initialiseringen.
3. Der du kan velge mellom kanter, velg den med lavest vekt.

Hva blir $v.d$ for hver node $v = 1, \ldots, 5$ etterpå? Forklar kort.

Oppgi verdien for hver node, i rekkefølge.

**Figur 2**




0, 3, 5, 8, 8

Vi oppdaterer (med RELAX) langs kantene med vekt $1, \ldots, 7$, i rekkefølge.

Av disse er det kun kantene $(1,2)$, $(2,3)$ og $(1,3)$, med vekt 3, 4 og 5, som har noen effekt.

5 % **17** Figur 2 viser et flytnett (*flow network*) med flyt. Utfør én iterasjon av EDMONDS-KARP på flytnettet for å finne en forøkende sti (*augmenting path*). Oppgi nodene i den resulterende stien, i rekkefølge. Forklar kort.

$s, 2, 5, t$

Det finnes flere forøkende stier, men EDMONDS-KARP velger den korteste.

Vi kan følge kanten $(5,2)$ baklengs, fordi det går flyt i den. Om man ikke tar høyde for det, sitter man igjen med de forøkende stiene $\langle s, 2, 6, 3, 7, t\rangle$ og $\langle s, 2, 6, 3, 7, 4, 8, t\rangle$. Om man oppgir den korteste av disse (den første), gir det 3 poeng. Om man oppgir den lengste, gir det 2 poeng.

Det samme uttelling også om man utelater $s$ og $t$ fra svaret.

5 % **18** Nøklene (*keys*) i et søketre er vanligvis fra en ordnet mengde (f.eks. tall eller tekststrenger). Din venn Gløgsund har laget et søketre der hver node i stedet inneholder et *rektangel*. Når hun skal bygge treet, tar hun inn et sett S med ikke-overlappende rektangler, som legges i løvnodene. For hver indre node konstruerer hun et nytt rektangel: det minste som inneholder alle barnas rektangler.

Når hun skal søke i treet, tar hun inn et punkt, og sjekker rekursivt nedover i treet hvilke rektangler som inneholder punktet, og returnerer det av disse som ligger i en løvnode.

Gjør følgende antagelser:

- Rektanglene fordeles tilfeldig på løvnodene.
- Treet er perfekt balansert.

Hva blir kjøretiden for et søk, som funksjon av $n = |S|$? Gi både en øvre og nedre grense, dvs., bruk enten $\Theta$-notasjon eller både O- og $\Omega$-notasjon.

> $O(n)$, $\Omega(\lg n)$. I beste tilfelle, vil søket kun følge én sti fra rot til løvnode, og siden treet er balansert, vil denne ha lengde $\Theta(\lg n)$. I verste tilfelle, kan rektanglene fordeles slik at vi får et lineært antall overlappende rektangler, som alle må besøkes.

5 % **19** Du utvikler et kodebibliotek for tekstprosessering, og skal implementere funksjonen `split(s, x)`, som deler strengen `s` ved alle forekomster av tegnet `x`. For eksempel vil

```
split("abc;de;fghi;jk", ";")
```

returnere

```
["abc", "de", "fghi", "jk"].
```

Du kan finne indeksene til `x` i `s` med en løkke. For en gitt slik indeks, kan du dele strengen i to mindre strenger, som er kopier av hver sin «halvdel».

Om strengen du deler inneholder $n$ tegn, må du kopiere $n - 1$ tegn. Konstruer og beskriv en algoritme som finner det minste totale antallet slike kopieringer som trengs. Hva blir kjøretiden? Forklar.

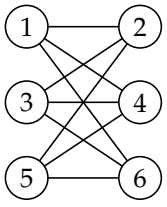
> For hvert splittpunkt, løs problemet høyre og venstre side for seg. Memoiser løsningen, f.eks. ved å ha en tabell $A[1:m, 1:m]$ som indekseres med start- og slutt-indeksene til segmentet man ser på. Denne tabellen kan også bygget opp fra bunnen (*bottom-up*).
>
> Det gis ingen uttelling i seg selv for å nevne dynamisk programmering.
>
> Oppgaven baserer seg på oppgave 14-9 i læreboka.

5 % **20** For to grafer G og H er en *grafhomomorfi* $f : G \to H$ en funksjon $f$ fra G.V til H.V, som bevarer naboskap. For alle $u, v \in$ G.V:

$$(u, v) \in \text{G.E} \implies (f(u), f(v)) \in \text{H.E}$$

Det vil si, hvis $(u, v)$ er en kant i G, så er $(f(u), f(v))$ en kant i H. Å avgjøre om det eksisterer en homomorfi, gitt G og H, er NP-komplett generelt, men kan i visse tilfeller gjøres i polynomisk tid.

Anta at vi bestemmer at H alltid skal være følgende graf, kjent som $K_{3,3}$:



Input er nå bare G, og vi skal avgjøre om det finnes en homomorfi $f : G \to K_{3,3}$.

Enten vis at denne begrensede versjonen av problemet fortsatt er NP-komplett, eller vis hvordan problemet kan løses (dvs., avgjøres) i polynomisk tid. Kan du trekke noen generelle konklusjoner fra svaret ditt? Forklar.

Om vi har funnet en $f : G \to K_{3,3}$, vil den fortsatt være gyldig om vi slår sammen verdier på høyre eller venstre side, så vi kan begrense oss til å bruke node 1 og 2.

Det er det samme som å si at det finnes en homomorfi videre til grafen $K_2$, som består av bare disse to nodene, med en kant imellom seg, og at $f$ finnes hvis og bare hvis det finnes en homomorfi $G \to K_2$.

Prøv å konstruere $h$ ved å traversere G, og å la $f(v) = 1$ eller $f(v) = 2$ for hver node $v \in G.V$, avhengig av hvilke verdier naboene alt har fått. Om vi mislykkes, må grafen ha en odde sykel, og det vil ikke finnes noen homomorfi.

Denne metoden fungerer fordi $K_{3,3}$ er bipartitt. Generelt, hvis H er bipartitt, kan vi i lineær tid avgjøre om $f : G \to H$ eksisterer, ved å avgjøre om G er bipartitt (tofargbar).