

## Avsluttende eksamen i IT1105/TDT4120 Algoritmer og datastrukturer

<b>Eksamensdato</b>	Torsdag 6. desember
<b>Eksamenstid</b>	1500–1900
<b>Sensurdato</b>	Søndag 6. januar
<b>Språk/målform</b>	Bokmål
<b>Kontakt under eksamen</b>	Magnus Lie Hetland (tlf. 91851949)
<b>Tillatte hjelpemidler</b>	Alle trykte/håndskrevne; bestemt, enkel kalkulator

Vennligst les hele oppgavesettet før du begynner, disponer tiden og forbered evt. spørsmål til faglærer kommer til eksamenslokalet. Gjør antagelser der det er nødvendig. **Skriv kort og konsist.** Lange forklaringer og utledninger som ikke direkte besvarer oppgaven tillegges liten eller ingen vekt. Skriv fortrinnsvis på anvist plass (dvs. uten å legge ved ark).

### Svarskjema for oppgave 1

(Se oppgavetekst side 2. Galt = 0 poeng, ubesvart = 1 poeng, riktig = 4 poeng.)

	Ja	Nei	Kort begrunnelse
<b>a</b>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<b>b</b>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<b>c</b>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<b>d</b>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<b>e</b>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<b>f</b>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<b>g</b>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<b>h</b>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

## Svarskjema for oppgave 2

(Se oppgavetekst side 3. Galt = 0 poeng, ubesvart = ¼ poeng, riktig = 1 poeng.)

Rekursive kall	Størrelse, delproblem	Arbeid i hvert kall	Rekurrens	Kjøretid
Ett	Redusert med 1	Konstant	$T(n) =$	$\Theta( \quad )$
Ett	Halvert	Konstant	$T(n) =$	$\Theta( \quad )$
Ett	Redusert med 1	Lineært	$T(n) =$	$\Theta( \quad )$
Ett	Halvert	Lineært	$T(n) =$	$\Theta( \quad )$
To	Redusert med 1	Konstant	$T(n) =$	$\Theta( \quad )$
To	Halvert	Konstant	$T(n) =$	$\Theta( \quad )$
To	Redusert med 1	Lineært	$T(n) =$	$\Theta( \quad )$
To	Halvert	Lineært	$T(n) =$	$\Theta( \quad )$

Oppgave 3 til 5 besvares i ruter i oppgaveteksten.

## Oppgave 1 (32%)

For hvert av delspørsmålene nedenfor, kryss av for «Ja» eller «Nei», og gi en kort, stikkordspreget begrunnelse for svaret ditt. Bruk svarskjema på side 1.

**Merk:** Besvarelser for denne oppgaven honoreres som følger: Galt svar eller gal begrunnelse for en deloppgave gir 0 poeng, ubesvarte deloppgaver gir 1 poeng og riktig besvarte deloppgaver (med riktige begrunnelser), 4 poeng. En vurdering av egne kunnskaper blir altså premiert – hvis du er mindre enn 25% sikker på svaret ditt (inkl. begrunnelse) lønner det seg å la deloppgaven stå ubesvart.

- Det er lettere å finne ut-kantgraden til en node hvis vi bruker nabomatrise-representasjon enn hvis vi bruker nabolister.
- Hvordan vi implementerer Quicksort har mye å si for hvor sårbar den er for ugunstige datasett.
- Anta at du sorterer  $n$  reelle tall for så å hente ut 10 tall med indekser spredd så jevnt som mulig fra 1 til  $n$ . Kunne du ha funnet de samme elementene uten å sortere, i lineær tid?
- På grunn av UNION/FIND-SET-mekanismen kan Kruskals algoritme finne minimale spennetrær også for usammenhengende grafer.
- Den grunnleggende metoden i pensum for å finne maksimal flyt har i verste tilfelle eksponentiell kjøretid.
- Enkelte NP-komplette problemer dreier seg om å finne en optimal permutasjon av  $n$  elementer. Det er mulig at noen av disse problemene har en

worst-case-kjøretid på  $\Omega(n!)$

- g. En algoritme tar inn heltallet  $n$  som eneste input og bruker binærsøk for å finne en heltallsverdi i intervallet  $[1, n]$  som er optimal (i en eller annen forstand). Anta at hver operasjon i binærsøket tar konstant tid, og at binærsøket dominerer den totale kjøretiden. Algoritmen har lineær kjøretid.
- h. Å bygge en heap har samme forventede, asymptotiske kjøretid som å bygge et søketre.

### Oppgave 2 (8%)

I svarskjemaet på side 2 ser du en oversikt over kjøretiden til et sett med rekursive algoritmer. Fyll ut det som mangler. Anta  $T(0) = T(1) = \Theta(1)$ .

**Merk:** Besvarelser for denne oppgaven honoreres som følger, for hver rad som helhet: Galt svar gir 0 poeng, manglende svar gir  $\frac{1}{4}$  poeng og riktige svar (på begge spørsmål), 1 poeng. En vurdering av egne kunnskaper blir altså premiert – hvis du er mindre enn 25% sikker på svaret ditt for en rad lønner det seg å la deloppgaven stå ubesvart.

### Oppgave 3 (27%)

Anta at du skal implementere en datastruktur som lar deg holde rede på to ting: (1) Rekkefølgen elementer legges til og plukkes ut i, og (2) hvorvidt et element finnes i strukturen. Du ønsker altså en struktur som både kan fungere som en FIFO-kø og en mengde.

- a. Hvordan vil du implementere en slik struktur? Legg vekt på asymptotisk kjøretid. Hva blir kjøretiden for de ulike operasjonene? (Bruk  $\Theta$ -notasjon og begrunn svaret).

Svar (12%):

Anta at du har et kart av en øygruppe i form av et rutenett der hver rute enten er hav/sjø eller land. (Anta at alle kantrutene er hav.) Du ønsker å skrive en algoritme med lineær kjøretid som teller hvor mange øyer som finnes på kartet.

**Merk:** Hvert selvstendige, sammenhengende segment med land telles som én øy. Mao. skal innlandsøyer også telles.

b. Beskriv kort en slik algoritme.

Svar (8%):

Du har laget en rekursiv algoritme som finner både maksimum og minimum i en tabell. Den gjør dette ved å rekursivt finne maksimum og minimum i hver halvdel, og så velge maksimum og minimum fra de to delsvarene. Anta at størrelsen på tabellen er  $n = 2^k$ , for et heltall  $k$ .

c. Hvor mange sammenligninger vil algoritmen din bruke? Begrunn svaret.

**Hint:** Svaret blir ikke  $n-1$  eller  $2n-2$ .

Svar (7%):

#### Oppgave 4 (7%)

Betrakt følgende algoritme.

---

```
DOLOREM(ipsum, quia, dolor)
  amet ← ipsum[dolor]
  consecetur ← quia-1
  for adipisci ← quia to dolor-1
    if ipsum[adipisci] ≤ amet:
      consecetur ← consecetur + 1
      ipsum[adipisci] ↔ ipsum[consecetur]
  ipsum[consecetur+1] ↔ ipsum[dolor]
  return consecetur+1

NUNQUAM(ipsum, quia, dolor, velit)
  sit ← DOLOREM(ipsum, quia, dolor, quia)
  if sit < velit:
    NUNQUAM(ipsum, velit, sit+1, dolor)
  else if sit > velit:
    NUNQUAM(ipsum, velit, quia, sit-1)
```

---

- a. Hva gjør algoritmen NUNQUAM? (Det vil si, hva beregner den – hva er hensikten med den? En oppsummering av hvilke trinn den utfører vil gi liten eller ingen uttelling.) Anta at *ipsum* er en 1-indeksert sekvens. Gjør andre naturlige antagelser om hvilke argumenter NUNQUAM kalles med.

Svar (7%):

### Oppgave 5 (26%)

Anta at du har et lokalt datanettverk der alle maskinene er koblet sammen, direkte eller indirekte. Kablene du har brukt har ulik pris (i hele kroner) og kvalitet (et reelt tall større eller lik 0 og mindre enn 10). Det er ikke nødvendigvis noen direkte sammenheng mellom pris og kvalitet.

Du skal velge ut et sett kabler som skal selges, og stiller følgende krav til løsningen:

- Alle maskiner må fremdeles være sammenkoblede, direkte eller indirekte etter at de valgte kablene er fjernet.
- Alle gjenværende koblinger må holde en viss minimumskvalitet,  $k$ .

Gitt disse to kravene skal du finne et sett kabler som gir deg en høyest mulig salgssum.

- a. Beskriv kort en effektiv algoritme som løser problemet generelt.

Svar (12%):

Etter å ha tenkt deg om kommer du med et tilleggskrav:

- Hvis det er flere optimale kabelsett skal du velge slik at de gjenværende kablene har en høyest mulig total kvalitet.

Du skal altså fremdeles finne et sett som gir høyest mulig utsalgspris, som ikke bryter opp nettet, og som lar hver gjenværende kabel ha en kvalitet på minst  $k$ . Å la det gjenværende nettet ha høyest mulig total kvalitet kommer som et tilleggskrav.

- b.** Beskriv kort en effektiv algoritme som løser problemet generelt.

Svar (7%):

Du har bestemt deg for å bruke operativsystem fra en bestemt kommersiell leverandør. Du har kjøpt én lisens (som gjelder for én maskin) for hver utgave av deres nyeste operativsystem. Målet er å få hver av datamaskinene dine til å kjøre ett av et sett med ulike serverprogrammer du har skrevet. Problemet er at (1) ikke alle operativsystem-versjonene kan kjøre på alle maskinene og (2) ikke alle serverprogrammene dine kan kjøre på alle operativsystem-versjonene.

- c.** Beskriv kort en effektiv algoritme som avgjør om det er mulig å få kjørt alle server-programmene (anta at det ikke er mulig å kjøre to serverprogrammer på samme maskin).

Svar (7%):