Høgskolen i Østfold

Ny/utsatt

EKSAMEN

Emnekode: **ITF20006**	Emne: **Algoritmer og datastrukturer**
Dato: **6. januar 2016**	Eksamenstid: **09:00 – 13:00**
Hjelpemidler: **Alle trykte og skrevne**	Faglærer: **Jan Høiberg**

Om eksamensoppgavene:

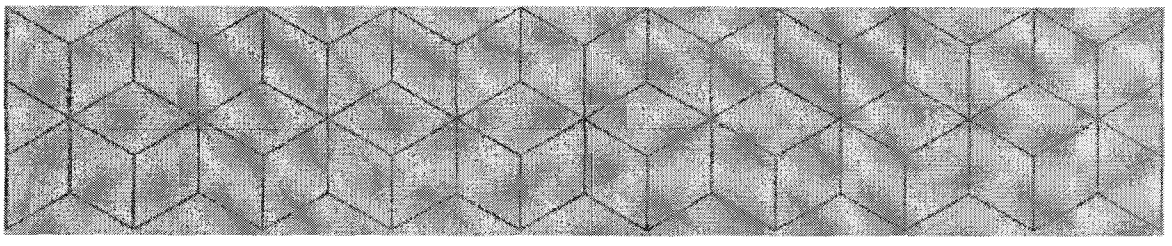
Oppgavesettet består av 6 sider, inkludert denne forsiden. Kontrollér at oppgaven er komplett før du begynner å besvare spørsmålene.

Oppgavesettet består av 4 oppgaver med i alt 15 deloppgaver. Innen hver oppgave vektes alle deloppgaver likt. Les oppgavetekstene nøye før du begynner på besvarelsen.

Legg vekt på å skrive en ryddig og lett forståelig besvarelse.

Sensurfrist: **27. januar 2017**

Karakterene er tilgjengelige for studenter på studentweb senest 2 virkedager etter oppgitt sensurfrist. www.hiof.no/studentweb



Oppgave 1: Algoritmeanalyse (25%)

I deloppgavene a) – e) nedenfor er det gitt i alt fem metoder. Gjør følgende for hver metode:

- Beskriv kort hva metoden utfører og evt. hva slags verdi den returnerer når den kjøres.
- Angi metodens arbeidsmengde med *O*-notasjon. Gi en kort begrunnelse for svaret.

a)
```
int metode_a(int A[], int x)
{
   int antall = 0, n = A.length;
   for (int i = 0; i < n; i++)
   {
      if (A[i] == x)
     antall++;
   }
   return antall;
}
```

b) Her skal du angi virkemåte og arbeidsmengde når parameteren `i` har verdien null (0):

```
int metode_b(int A[], int x, int i)
{
   int n = A.length;
   if (i < n)
   {
      if (A[i] == x)
         return 1 + metode_b(A, x, i+1);
      return metode_b(A, x, i+1);
   }
   else
      return 0;
}
```

c) Merk at metoden her kaller `metode_b` fra forrige deloppgave:

```
boolean metode_c(int A[], int x)
{
    int antall = metode_b(A, x, 0);
    return antall == 0 ? false : true;
}
```

(Oppgave 1 fortsetter på neste side)

d)

```
void metode_d(int A[])
{
   int tmp, n = A.length;
   for (int i = 0; i < n; i++)
   {
       for (int j = n - 1; j > i; j--)
          if (A[j] < A[j - 1])
          {
             tmp = A[j];
             A[j] = A[j - 1];
             A[j - 1] = tmp;
          }
   }
}
```

e) Merk at metoden her kaller `metode_d` fra forrige deloppgave:

```
boolean metode_e(int A[], int x)
{
   int n = A.length;
   int min = 0, max = n-1, mid = 0;
   metode_d(A);
   while (max >= min)
   {
      mid = (min + max) / 2;
      if (A[mid] == x)
         return true;
      if (x < A[mid])
         max = mid - 1;
      else
         min = mid + 1;
   }
   return false;
}
```

f) De to metodene som er angitt i deloppgave c) og deloppgave e) ovenfor, løser det samme problemet. De er begge lite effektive, spesielt gjelder dette for `metode_e`.

Foreslå en enkel algoritme kan brukes til å løse det samme problemet mer effektivt. Det er tilstrekkelig at du bare angir navnet/betegnelsen på algoritmen.

(Slutt på oppgave 1)

Oppgave 2: Kø (20%)

Denne oppgaven dreier seg om vanlige (First-In-First-Out) køer.

a) En kø implementeres med bruk av en array, slik at det første elementet i køen alltid ligger lagret på indeks 0 (null) i arrayen. Det andre elementet i køen ligger lagret på indeks 1, det tredje elementet på indeks 2 osv.

Hva er da arbeidsmengden, uttrykt med O-notasjon, for de to operasjonene `enqueue` og `dequeue` ? Gi en kort begrunnelse for svaret ditt.

b) En kø implementeres med bruk av en array, slik at det siste elementet i køen alltid ligger lagret på indeks 0 (null) i arrayen. Det nest siste elementet i køen ligger lagret på indeks 1, elementet foran det neste siste på indeks 2 osv.

Hva er da arbeidsmengden, utrykt med O-notasjon, for de to operasjonene `enqueue` og `dequeue` ? Gi en kort begrunnelse for svaret ditt.

c) Hvorfor brukes vanligvis ikke de to implementasjonene av kø som er beskrevet i deloppgavene a) og b) ovenfor? Beskriv kort to alternative implementasjoner av kø som begge er mer effektive.

(Slutt på på oppgave 2)

Oppgave 3: Datastrukturer (30%)

a) Beskriv *kort* fordeler og ulemper med følgende datastrukturer når det gjelder søking, innsetting og fjerning av data:

1. Lenket liste
2. Hashtabell
3. Binært søketre
4. Array

b) En hashtabell med hashlengde 11 skal brukes til å lagre heltall. Tabellen er i utgangspunktet tom. Følgende hashfunksjon brukes:

```
hash(x) = x % 10
```

Det brukes åpen adressering med lineær probing ved kollisjon. Vis hvordan hashtabellen ser ut etter innsetting av følgende verdier i denne rekkefølgen:

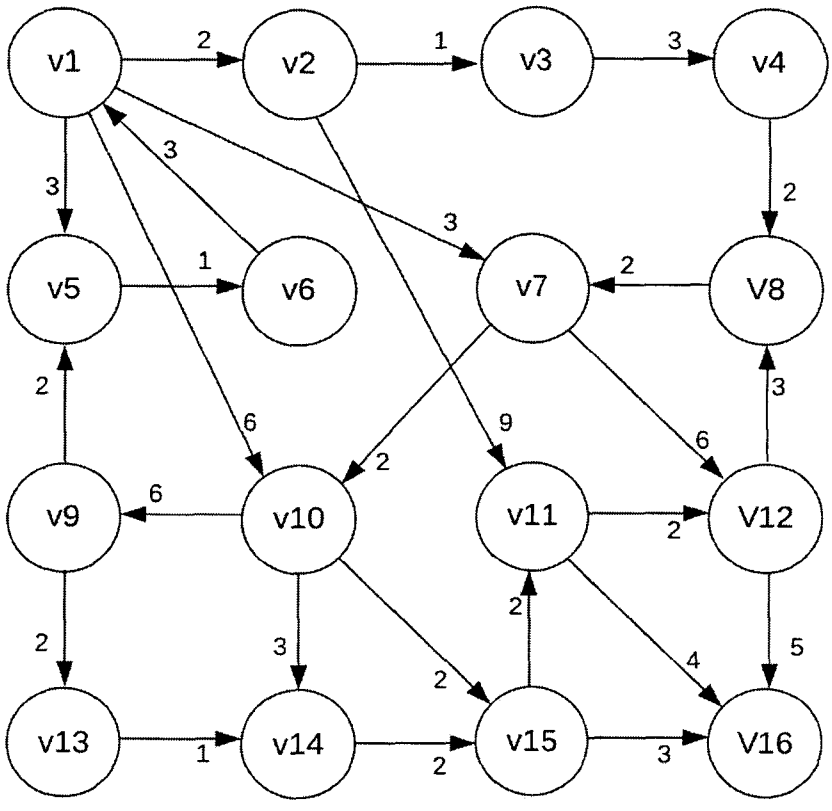
6 , 5 , 28 , 17 , 18 , 115 , 16 , 7 , 71 , 60

c) I et komplett binært tre med 20 noder der roten er definert til å være på nivå 0, hvor mange noder er det på nivå 4?

(Slutt på opppgave 3)

Oppgave 4: Grafer (25%)

Følgende graf er gitt:



Figur 1 - En rettet vektet graf

a) Hvor lang er den korteste veien fra node `v1` til node `v16` ? Hvilke noder ligger langs denne korteste veien?

b) Sett opp en oversiktlig tabell som viser stegene som gjøres i Dijkstra's algoritme for denne grafen med start i node `v1`, frem til vi har funnet korteste vei til node `v16`. Hver rad i tabellen skal vise tilstanden etter en iterasjon og skal angi:

- Hvilke noder som vi kjenner *garantert* korteste vei til, og hvor lang denne veien er.
- Hvilke andre noder som er oppsøkt og som vi kjenner en *mulig* korteste vei til, og hvor lang denne veien er.
- Hvilke noder som ennå ikke er oppsøkt.

c) Anta at figur 1 er et representativt utsnitt av en mye større graf som består av 20000 noder. Hvordan ville du lagret dataene om denne store grafen? Begrunn svaret.

(Slutt på opppgave 4)