OMTENTAMEN I
DATASTRUKTURER OCH ALGORITMER DAV B03

130611 kl. 08:15 – 13:15

Ansvarig Lärare: Donald F. Ross

Hjälpmedel: Inga. Algoritmerna finns i de respektive uppgifterna.

***** OBS *****

Ni som har läst från och med HT 2006

Betygsgräns:

Kurs:	Max 60p, Med beröm godkänd 50p, Icke utan beröm godkänd 40p, Godkänd 30p **(varav minimum 15p från tentamen, 15p från labbarna)**
Tentamen:	Max 30p, betyg 5: 26p-30p, betyg 4: 21p-25p, betyg 3: 15p-20p
Labbarna:	Max 30p, betyg 5: 26p-30p, betyg 4: 21p-25p, betyg 3: 15p-20p

Ni som har läst tidigare än HT 2006

Betygsgräns:

Kurs:	Max 60p, Med beröm godkänd 50p, Icke utan beröm godkänd 40p, Godkänd 30p **(varav minimum 20p från tentamen, 10p från labbarna)**
Tentamen:	Max 40p, betyg 5: 34p-40p, betyg 4: 27p-33p, betyg 3: 20p-26p
Labbarna:	Max 20p, betyg 5: 18p-20p, betyg 4: 14p-17p, betyg 3: 10p-13p

SKRIV TYDLIGT – LÄS UPPGIFTERNA NOGGRANT

***** OBS *** Denna tentamen är kopierad på båda sidor *** OBS *****

---

(1) Ge ett kortfattat svar till följande uppgifter.

(a) Vad är ”big-O” för en funktion som skriver ut en adjacency matrix? Varför?
(b) Vad gör Dijkstras algorithm?
(c) Vad gör Floyds algorithm?
(d) Vad gör Warshalls algorithm?
(e) Vad gör Topologisk sortering?
(f) Vad är en heap?
(g) Vad är fördelen med hashning?
(h) Vad är en rekursiv funktion?
(i) Vad är ett AVL-träd?
(j) Vad är dubbel hashning?

Totalt 5p

(2) Sekvens

Diskutera ingående ADT:en sekvens, dess egenskaper och varför sekvensen är så viktigt inom datavetenskap. Vilket förhållande har sekvensen till ADT:erna ”träd” och ”graf”?

5p

(3) Träd

(a) Ange en rekursiv definition av ett träd.

(1p)

(b) Utifrån din definition i (a) ovan, skriv rekursiv pseudokod för att räkna antalet element i ett binärt träd.
Ange alla antaganden.
Ange ett exempel för att visa hur din kod fungerar.

(2p)

(c) Under kursens lopp hade vi diskuterat 2 sätt att ta bort ett element från ett BST. Vilka var dessa?
Ange alla antaganden.
Ange exempel för att visa hur en ”ta bort” funktion skulle fungerar.

(2p)

Totalt 5p

(4) Övriga frågor – svar ingående med exempel

(a) En student har skapat ett AVL-träd från följande sekvens:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
Rita slutversionen av trädet. 1p

(b) Hur gör man för att skriva ut ett binärträd i 2 dimensioner?
Vilka av trädets egenskaper använder man?
2p

(c) Förklara principerna bakom **Kruskals algorithm**. 2p

Totalt 5p

(5) Labbkod

(a) I graflabben har en student skrivit följande kod för att ta bort en kant (edge) från an adjacency lista. Förklara **ingående** hur koden fungerar. Använd gärna exempel. **Ange alla antagande.**
Vilka är förutsättningarna för att koden ska fungera?

```
void reme(char cs, char cd) {
  set_edges(b_findn(cs, G), b_reme(cd, get_edges(b_findn(cs, G))));
  }
```

2p

(b) I trädlabben har en student skrivit kod för att söka efter ett värde i ett BST (binärt sökträd). Sedan har studenten kommit på att denna kod kunde lätt anpassas för att söka efter ett värde i ett komplett träd. Dessa funktioner finns nedan. Vad har studenten skrivit för ”**xxx**” och ”**yyy**”? **Ange alla antagande.**

```
static int b_findb(treeref T, int v)
{
  return is_empty(T) ? 0
        : v <  get_value(node(T)) ? b_findb(LC(T), v)
        : v >  get_value(node(T)) ? b_findb(RC(T), v)
        : 1;
}

static int b_findc(treeref T, int v)
{
  return is_empty(T) ? 0
        : xxx ? 1
        : yyy;
}
```

1p

(c) Skriv (pseudo)kod för att lägga till ett element i ett binärt träd.
Ange alla antagande. **2p**

Totalt 5p

(6) Graf - Prims algorithm

Tillämpa **den givna Prims algoritm nedan** på **den oriktade grafen**,

(a-20-b, a-36-c, a-34-d, b-22-c, b-24-e, c-28-d, c-30-e, c-38-f, d-26-f, e-36-f).

Börja med nod ”a”.

Ange *alla* antaganden och visa *alla* beräkningar och mellanresultat

(3p)

Förklara **principerna** bakom Prims algoritm. Använd gärna exemplet ovan.

(2p)

```
Prim ( node v) -- v is the start node
{   U = {v}; for i in (V-U) { low-cost[i] = C[v,i]; closest[i] = v; }

    while (!is_empty (V-U) ) {
        i = first(V-U); min = low-cost[i]; k = i;
        for j in (V-U-k) if (low-cost[j] < min) {min = low-cost[j]; k = j; }
        display(k, closest[k]);
        U = U + k
        for j in (V-U) if ( C[k,j] < low-cost[j] ) )   {low-cost[j] = C[k,j]; closest[j] = k; }
        }
    }
```

Totalt 5p