OMTENTAMEN I
DATASTRUKTURER OCH ALGORITMER DVG B03

160402 kl. 09:00 – 14:00

Ansvarig Lärare: Donald F. Ross

Hjälpmedel: Inga. Algoritmerna finns i de respektive uppgifterna eller i bilogarna.

***** OBS *****

Betygsgräns:	Kurs:	Max 60p, Med beröm godkänd 50p, Icke utan beröm godkänd 40p, Godkänd 30p (varav minimum 20p från tentan, 10p från labbarna)
	Tenta:	Max 40p, Med beröm godkänd 34p, Icke utan beröm godkänd 27p, Godkänd 20p
	Labbarna:	Max 20p, Med beröm godkänd 18p, Icke utan beröm godkänd 14p, Godkänd 10p

SKRIV TYDLIGT – LÄS UPPGIFTERNA NOGGRANT

Ange alla antaganden.

---

(1) Ge ett kortfattat svar till följande uppgifter ((a)-(j)).

(a) Vad skulle ett alternativ för Floyds algoritm vara?
(b) Vad är en rekursiv definition?
(c) Vad betyder ”collection abstraction”?
(d) Ge en definition av ett träd?
(e) Ge ett exempel av **en allmän djupet-först** traversering av ett BST?
(f) Vad är meningen med sortering?
(g) Vad är hanterliga (”tractable”) problem i komplexitetsteori?
(h) Nämn två metoder för att upptäcka cykler i grafer.
(i) Vad är ett minimal spanningträd (”minimal spanning tree”) (MST)?
(j) Nämn två exempel av algoritmer där djupet-först-sökning tillämpas?

Totalt 5p

(2) Ge ett kortfattat svar till följande uppgifter ((a)-(e)).

(a) Beskriv **principen** bakom **Dijkstras** algoritmen.

2p

(b) Beskriv en **abstrakt data typ (ADT)** som en **virtuell maskin**. Vilka fördelar finns med en sådan beskrivning?

2p

(c) Skriv **abstrakt rekursiv pseudokod** till en funktion för att ombalansera ett AVL-träd. Ange alla antagande och förklara alla hjälpfunktioner som behövs.

2p

(d) Skriv **abstrakt rekursiv pseudokod** till funktionen **T2Q** från labb 1. **T2Q** förvandlar ett binärt träd till en array.

2p

(e) Skriv den **rekursiva definitionen** av en sekvens och ett binärt-träd.

2p

Totalt 10p

(3) Hashning

Diskutera ingående hur hashning fungerar. Vilket är det största problemet? Hur löser man detta? Vilka aspekter skulle man ta hänsyn till? Ge exempel för varje fall Du beskriver.

5p

(4) Diskussionsuppgift

Man kan påstå att sekvensen är den viktigaste abstrakt datastrukturen (ADT). Skriv en utförlig beskrivning av en sekvens (2p) och ett detaljerad diskussion som stödjar detta påstående (3p).
Svara ingående. Ange alla antagande.

Total 5p

(5) Rekursion

Titta på koden för att ta bort ett värde från ett BST nedan.
Denna kod ska tillämpas på ett binärt sök träd som skapats genom att lägga till följande värden i denna ordning:-

10, 5, 30, 20, 15, 25, 13, 17, 16

Värdet 15 ska sedan tas bort.
Ange alla antagande.

1. Skriv (pseudo)koden till funktionen HDiff(treeref T)
   (1p)

2. Skriv (pseudo)koden till funktionerna LCmaxAsRoot(treeref T) samt RCminAsRoot(treeref T)
   (1p)

   Använd rekursion och samma programmeringsstil för Dina versioner av HDiff, LCmaxAsRoot och RCminAsRoot.

3. **Förklara stegvis och utförligt** hur koden, inklusiv Dina versioner av (i) HDiff (ii) LCmaxAsRoot och (iii) RCminAsRoot fungerar när den tilllämpas på detta träd.
   Visa varje rekursivt anrop till b_rem. Det första anropet är b_rem([10], 15) där [10] står för trädet med rotvärde 10. Dvs [x] står för trädet med rotvärde x.

   Rita trädet som returneras efter varje anrop i rader 2-7.
   (3p)

Total 5p

```
static treeref b_rem(treeref T, int v)
{
    1.  return is_empty(T)          ? T
    2.  : v < get_value(node(T))    ? cons(b_rem(LC(T), v), node(T), RC(T))
    3.  : v > get_value(node(T))    ? cons(LC(T), node(T), b_rem(RC(T), v))
    4.  : is_empty(LC(T))           ? RC(T)
    5.  : is_empty(RC(T))           ? LC(T)
    6.  : HDiff(T) > 0              ? LCmaxAsRoot(T)
    7.  :                             RCminAsRoot(T);
}
```

(6) 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 abstrakt rekursiv (pseudo)kod för att lägga till ett element i ett binärt träd. **Ange alla antagande.**

2p

Totalt 5p

(7) Prims

a) Tillämpa **Prims algoritm** (nedan) på den **oriktade** grafen:

(a-7-b, a-1-c, a-10-d, b-3-c, b-2-e, c-15-d, c-4-e, c-9-f, d-4-f, e-8-f).

Börja med nod a. Ange alla antaganden och visa alla beräkningar och mellanresultat. Vad representerar resultatet? **Ange varje steg i din beräkning.**

Rita delresultatet efter varje iteration.

(2p)

b) **Förklara principerna bakom Prims** algoritm.

(2p)

c) Visa hur Du skulle använda **principerna** bakom **Kruskals algoritm** för att bekräfta resultatet från Prims algoritm.

(1p)

Totalt 5p

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

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; }
   }
}
```