TENTAMEN I
DATASTRUKTURER OCH ALGORITMER DAV B03

130115 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) Visa förhållandet mellan ett BST, ett komplett träd (complete tree), samt ett fullt träd (full tree) med hjälp av en Venn-diagram från mängdläran.
(b) Vad är det maximum belastingsfaktorn (load factor) i hashning med kvadratisk probing (quadratic probing) som kollisionshanteringsteknik som garanterar att man alltid kan hitta en plats?
(c) Vad är ”big-O”?
(d) Vad gör Warshalls algoritmen?
(e) Ge en definition av ett AVL-träd.
(f) Vilken grammatisk klass i natuliga språk motsvarar en relation i den entitet-relation modellen?
(g) Vilken algoritm kräver en DAG som förutsättning?
(h) Vilka datastrukturer är ordnade?
(i) Vilka algoritmer är $O(n^3)$?
(j) Ge en definition av en rekursiv funktion.

Totalt 5p

(2) Abstraktion

Diskutera ingående varför abstraktion är så viktigt inom datastrukturer. Finns några eventuella nackdelar till abstraktion? Vilka 3 definitioner av begreppet abstraktion har vi använt i denna kurs?

5p

(3) Sekvens

(a) Ange en rekursiv definition av en sekvens.

(1p)

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

(2p)

(c) Utifrån din definition i (a) ovan, skriv rekursiv pseudokod för att lägga till ett element i en skevens i stigande (sorterad) ordning. Anta att det redan finns en funktion som heter Cons som lägger till ett element i huvudet av listan.
Cons: element x lista ➔ lista.
Ange alla andra antaganden.
Ange ett exempel för att visa hur din kod fungerar.

(2p)

Totalt 5p

(4) Övriga frågor – svar ingående med exempel

(a) Vad är dubbelhashning?
(b) Vad använder man en stack till?
(c) Hur förvandlar man ett generellt träd till ett binärt träd?
(d) Hur fungerar add-funktionen i en heap?
(e) Beskriv en lösning till TSP-problemet. TSP = Travelling Salesman Problem.

Totalt 5p

(5) Graf – Dijkstra + SPT

Utöka Dijkstras algoritmen nedan för att kunna spara och visa SPT:et
(SPT: Shortest Path Tree – svenska KVT: Kortaste väg trädet).

```
Dijkstra ( )
{       S = {a}

        for ( i in 2..n) D[i] = C[a, i]          -- initialise D

        for (i in 1..(n-1)) {
                choose w in V - S such that D[w] is a minimum
                S = S + {w}
                foreach ( v in V-S) D[v] = min(D[v], D[w]+C[w,v])
                }
        }
```

(3p)

Tillämpa **din utökad algoritmen** på **den riktade grafen**,

(a, b, 13), (a, d, 12), (a, e, 20), (b, c, 60), (c, e, 50), (d, c, 30), (d, e, 40)

Börja med nod ”a”.
Visa varje steg i dina beräkningar.
Ange *alla* antaganden och visa *alla* beräkningar och mellanresultat
Rita **varje steg** i konstruktionen av SPT:et – dvs visa till och med de noder och kanter som läggs till men sedan tas bort.

(2p)

Totalt 5p

(6) Graf - Prims algorithm

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

(a-6-b, a-3-c, a-7-d, b-1-c, b-12-e, c-4-d, c-5-e, c-9-f, d-3-f, e-8-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