OMTENTAMEN I
DATASTRUKTURER OCH ALGORITMER DVG B03

150817 kl. 08:15 – 13:15

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.

Studenter som har läst kursen from HT2014 ska svara på samtliga uppgifter.

Studenter som har läst kursen from HT2006 tom HT2013 ska välja uppgifter värt 30 poäng.

Studenter som har läst kursen tom HT2005 ska svara på samtliga uppgifter.

---

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

(a) Vad är ”big-O” för Dijkstras algoritm?
(b) Vad är ”big-O” för Floyds algoritm?
(c) Vad är ”big-O” för lägga-till-operationen i hashning?
(d) Vad gör Warshalls algoritm?
(e) Vad gör en postorder traversering av ett binärt träd?
(f) Vad är en graf?
(g) Vad representerar kanterna i en graf?
(h) Vad är ett ”Free Tree”?
(i) Vad händer när man lägger till en kant till i ett ”Free Tree”?
(j) Vad är kvadratisk probning (quadratic probing)?

Totalt 5p

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

(a) Under vilka förutsättningar skulle SPT:et (Shortest Path Tree) och MST:et (Minimal Spanning Tree) vara identiska när man tillämpar Dijkstra-SPT och Prims algoritmer på samma oriktade graf?
(b) Skriv rekursiv pseudokod till en funktion för att hitta det maximala värdet i ett vänsterbarn i ett BST (Binärt SökTräd).
(c) Ge en rekursiv definition av ett BT (Binärt Träd).
(d) Vilka begränsningar måste man specificera för att förvandla ett binärt träd (BT) till ett binärt sökträd (BST) och ett BST till ett AVL-träd?
(e) Förklara vad en ADT (abstrakt datatyp) är.

Totalt 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 sekvens i stigande (sorterad) ordning. Anta att det redan finns en funktion som heter Cons som lägger till ett element i huvudet av sekvensen.
Cons: element x sekvens ➔ sekvens.
Ange alla andra antaganden.
Ange ett exempel för att visa hur din kod fungerar.

(2p)

Totalt 5p

(4) Rekursion

Skriv pseudokod till **två rekursiva funktioner** (alltså en funktion plus en hjälpfunktion) för att räkna fram antalet kanter (”edges”) i en graf. Ange alla antagande.

5p

(5) Topologisk sortering

Vid ett universitet har vissa kurser förkunskapskrav. I datavetenskap kräver kompilatorkonstruktion (DAV D02) programspråk (DAV C02) som förkunskap. Datastrukturer och algoritmer (DAV B03) är ett förkunskapskrav till programspråk, avancerad programmering i C++ (DAV C05), samt projektarbete i Java (DAV C08). Datastrukturer och algoritmer kräver diskret matematik (MAA B06) samt programutvecklingsmetodik (DAV A02). Operativsystem (DAV B01) kräver i sin tur programutvecklingsmetodik och datorsystemteknik (DAV A14) och är förkunskapskrav till C och UNIX (DAV C18), tillämpad datasäkerhet (DAV C17) samt realtidssystem (DAV C01). Objektorienterade designmetoder (DAV D11) kräver bägge avancerad programmering i C++ och software engineering (DAV C19).

Hur kan man **visa** att kompilatorkonstruktion och objektorienterade designmetoder kräver diskret matematik?

I vilken ordning ska en student som vill läsa på D-nivå ta alla de ovannämnda kurserna? Metoden som kan användas för att komma fram till en lösning heter topologisk sortering. En variant av topologisk sortering är följande algoritm:

Topological Sort

```
tsort(v) -- prints reverse topological order of a DAG from v
{       mark v visited
        for each w adjacent to v if w unvisited tsort(w)
        display(v)
        }
```

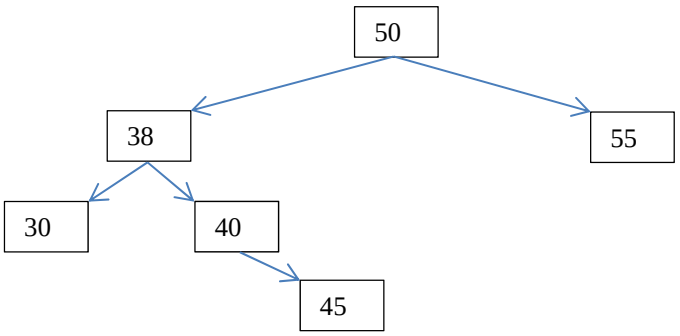
Vilken är den andra varianten? Tillämpa både varianter i din lösning till problemet ovan.

Vilka begränsningar måste man ta hänsyn till? **5p**

(6) AVL-Träd

Visa hur Du skulle ta fram de 4 rotationsfunktionerna från första principer. 2p

Förklara vad ”balansfaktorn” (balance factor) är och använd denna för att ta fram balansfunktionen från första principer. **Skriv balansfunktionen i pseudokod.** Tillämpa Din pseudokod på trädet nedan. **Visa varje steg i beräkningarna!** 3p



Totalt 5p

(7) Prims algoritm

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).

1. **Börja med nod ”a”**
2. **Visa varje steg i beräkningarna**
3. **Ange *alla* antaganden och visa *alla* beräkningar och mellanresultat!**

(3p)

Förklara hur Prims algoritm fungerar genom att rita bilder som representerar varje mellanresultat under algoritmensexekvering. Använd exemplet ovan.
Vad är principen bakom Prims algoritm?

(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

(8) Dijkstra + SPT (Shortest Path Tree)

Tillämpa **den givna Dijkstra_SPT algoritmen (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).

SPT = Shortest Path Tree - dvs kortaste väg trädet (KVT) från en nod till alla de andra.

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

(3p)

Förklara hur Dijkstra-SPT algoritm fungerar genom att rita bilder som representerar varje mellanresultat under algoritmensexekvering. Använd exemplet ovan.

(2p)

Totalt 5p

Dijkstras algoritm med en utökning för SPT

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

    for (i in V-S) {
      D[i]     = C[a, i]                    --- initialise D - (edge cost)
      E[i]     = a                          --- initialise E - SPT (edge)
                 L[i]     = C[a, i]         --- initialise L - SPT (cost)
    }

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