ORDINARIE TENTAMEN I
DATASTRUKTURER OCH ALGORITMER DVG B03

150112 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.

---

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

(a) Vad är ”big-O” för en funktion som skriver ut en adjacency matrix? Varför?
(b) Vad gör Dijkstras algoritm?
(c) Vad gör Floyds algoritm?
(d) Vad gör Warshalls algoritm?
(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) Ge ett kortfattat svar till följande uppgifter ((a)-(e)).

(a) Kan noden med det maximala värdet i ett vänsterbarn i ett BST (Binärt SökTräd) ha ett högerbarn? Förklara varför!
(b) Skriv rekursiv pseudokod till en funktion för att hitta det minimala värdet i ett högerbarn i ett BST (Binärt SökTräd).
(c) Ge en rekursiv definition av ett BT (Binärt Träd)
(d) Skriv en rekursiv sök (”find”) funktion för ett BT (Binärt Träd) – **OBS – ej BST!**
(e) Förklara hur Du skulle representera ett BT (Binärt Träd) med hjälp av en array. Vad är förhållandet mellan trädet och arrayen?

Totalt 5p

(3) Heap

Diskutera ingående hur koden till heap operationer (**se Bilaga A**) fungerar. Använd sekvensen 46, 13, 18, 33, 72, 9, 11, 44, 27, 15, 66 som ett exempel. Anta att det största värdet hamnar i roten. **Visa varje steg i Dina beräkningar.**

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) Diskussionsuppgift

Vad menas med **implentationsabstraktion**? Presentera för- och nackdelar till implementationsabstraktion. Ledar det fram till att man producerar bättre kod? **Diskutera ingående. Ge alla antagande.**

5p

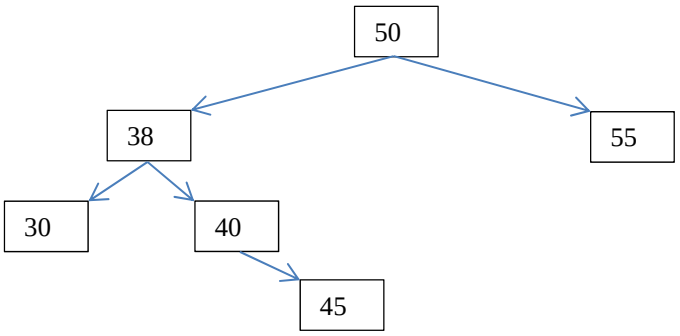
(6) AVL-Träd

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

3p



Totalt 5p

(7) Kruskals algoritm

Beskriv principerna bakom Kruskals algoritm. Vad blir resultatet av en tillämpning av algoritmen? Använd grafen nedan (oriktad) som exempel.

(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)

3p

Hur skulle man kunna anpassa idéer från Kruslals algoritmen för att ta fram en heuristik för att ge en lösning till det resande försäljare-problemet (Travelling Salesman Problem)?

2p

Totalt 5p

(8) Dijkstra + SPT (Shortest Path Tree)

Tillämpa **den givna Dijkstra_SPT algoritmen (nedan)** på **den riktade grafen**,

(a, b, 12), (a, d, 11), (a, e, 9), (b, c, 7), (c, e, 5), (d, c, 3), (d, e, 1)

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

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.

(3p)

Förklara **principerna** bakom **Dijkstras_SPT** algoritm.

(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]
                 }
        }
}
```

Bilaga A

Heap Algoritmer

```
Heapify(A, i)
  l = Left(i)
  r = Right(i)
  if l <= A.size and A[l] > A[i] then largest = l else largest = i
  if r <= A.size and A[r] > A[largest] then largest = r
  if largest != i then
        swap(A[i], A[largest])
        Heapify(A, largest)
        end if
  end Heapify

Build(A)
  for i = [A.size / 2] downto 1 do Heapify(A, i)
  end Build
```

```
Remove (H, r)
    let  A = H.array
    A[r] = A[A.size]
    A.size--
    Heapify(A, r)
     end Remove
```

```
Add (H, v)
        let A = H.array
        A.size++
        i = A.size
        while i > 1 and A[Parent(i)] < v do
                    A[i] = A[Parent(i)]
                     i = Parent(i)
              end while
        A[i] = v
end Add
```