

i Information

- Försök på alla uppgifter! Uppgifterna är inte ordnade på något speciellt sätt.
- Det är ditt ansvar att övertyga oss att du besitter den kunskap som efterfrågas.
- Det är viktigt att du löser den *givna* uppgiften!
- På sista "frågan" i denna tentamen finns plats att skriva kommentarer om du vill förtydliga något kring dina svar på någon fråga. "Frågan" ger inga poäng.
- Lärarna kommer **inte** att besöka tentasalen under provets gång. Det går dock att nå en av lärarna (Niclas) **under första timman**.
- Endast de som är underkända på tidigare tentamina får skriva denna tentamen. Plussning (dvs att skriva om efter du blivit godkänd i syfte att få högre betyg) är inte tillåtet.

Betyg:

Maximal poäng är 100.

- För betyg **3** (godkänt) krävs **50** poäng.
- För betyg **4** krävs **65** poäng.
- För betyg **5** krävs **80** poäng.

Lycka till!

Ola, Niclas, Vicenç

1 Terminologi (10p)

Nedan kommer 10 sant/falskt frågor. Rätt svar ger 1p, felaktigt svar -1p och inget svar alls ger 0p. Du kan få 0-10p på denna uppgift (alltså inte minusresultat). Med datatyp avses en abstrakt datatyp med definition enligt boken.

En ordnad datatyp kan vara osorterad.

☐ Sant



☐ Falskt

Kö är en heterogen datatyp.

☐ Sant

☐ Falskt



En datatyp vars objekt inte består av element som i sin tur är datatypsobjekt kallas konstruerad datatyp. Tex heltal.

☐ Sant

☐ Falskt



En konkret datatyp kan vara implementerad.

☐ Sant



☐ Falskt

Man kan prata om en lista som innehåller tre element och ett värde.

☐ Sant



☐ Falskt

En heterogen datatyp är en sammansatt datatyp där elementen är av olika datatyper.

☐ Sant



☐ Falskt

En datatyp som är konstruerad av en implementerad datatyp är implementerad.

☐ Sant



☐ Falskt

Mängd är en oordnad datatyp.

☐ Sant



☐ Falskt

En datatyp som finns tillgänglig i en given maskinvara eller programspråk kallas abstrakt datatyp.

☐ Sant

☐ Falskt



Enkel datatyp är en term som används när man beskriver, diskuterar eller använder en datatyp utan att ta hänsyn till om eller hur den är realiserad i ett programspråk/hårdvara.

☐ Sant

☐ Falskt



2 Huffman (10p)

a) Givet ett Huffman-träd för strängen DISTANCE:

Hur lång blir den *längsta* sekvensen bitar som behövs för att koda ett tecken? (3)

Hur lång blir den *kortaste* sekvensen bitar som behövs för att koda ett tecken? (3)

b) Konstruera ett Huffman-träd för följande text:

DUBBAR ÄR BRA ATT HA (bortse från mellanslagen).

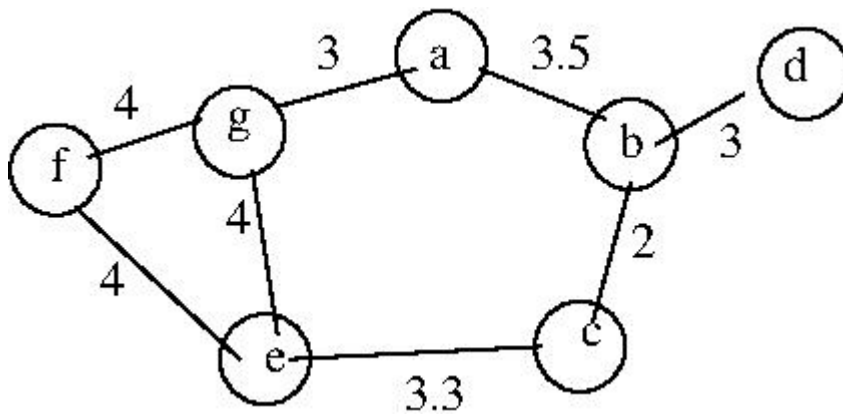
Följ dessa regler:

- Sortera löven i storleksordning, med största värdet till höger och minsta till vänster.
- Ordningen på två noder som slås ihop bestäms i första hand av deras vikt och i andra hand av bokstavsordningen (a först, ö sist).
 - Om det finns fler än två val med samma vikt när du ska slå samman noder skall du använda de två som har lägst höjd.
- Etiketten är 0 på vänster-grenar och 1 på höger-grenar.

Ange för var och en av bokstäverna vilken kod de får:

D: (0000) U: (0100) B: (011) A: (11)

R: (10) Ä: (0101) T: (001) H: (0001)

3(a) **Prims algoritm (12p)**

Antag att Prims algoritm appliceras på ovanstående graf, med början i nod d .

I steg 1 är det korrekt att välja följande båge:

- ☐ (c,e)
 ☐ (e,f)
 ☐ (e,g)
 ☐ (b,c)
 ☐ (f,g)
 ☐ (g,a)
 ☒ (b,d)

☐ (a,b)

I steg 2 är det korrekt att välja följande båge:

- ☐ (f,g)
 ☐ (a,b)
 ☐ (b,d)
 ☒ (b,c)
 ☐ (g,a)
 ☐ (e,g)
 ☐ (c,e)

☐ (e,f)

I steg 3 är det korrekt att välja följande båge:

(c,e)



(a,b)



(b,d)



(e,g)



(e,f)



(b,c)



(f,g)



(g,a)



I steg 4 är det korrekt att välja följande båge:

(e,g)



(e,f)



(a,b)



(c,e)



(g,a)



(b,c)



(f,g)



(b,d)



I steg 5 är det korrekt att välja följande båge:

(a,b)



(b,d)



(e,f)



(c,e)



(e,g)



(g,a)



(b,c)



(f,g)



I steg 6 är det korrekt att välja följande båge:

(a,b)

☐

(f,g)



(b,d)

☐

(e,g)

☐

(g,a)

☐

(c,e)

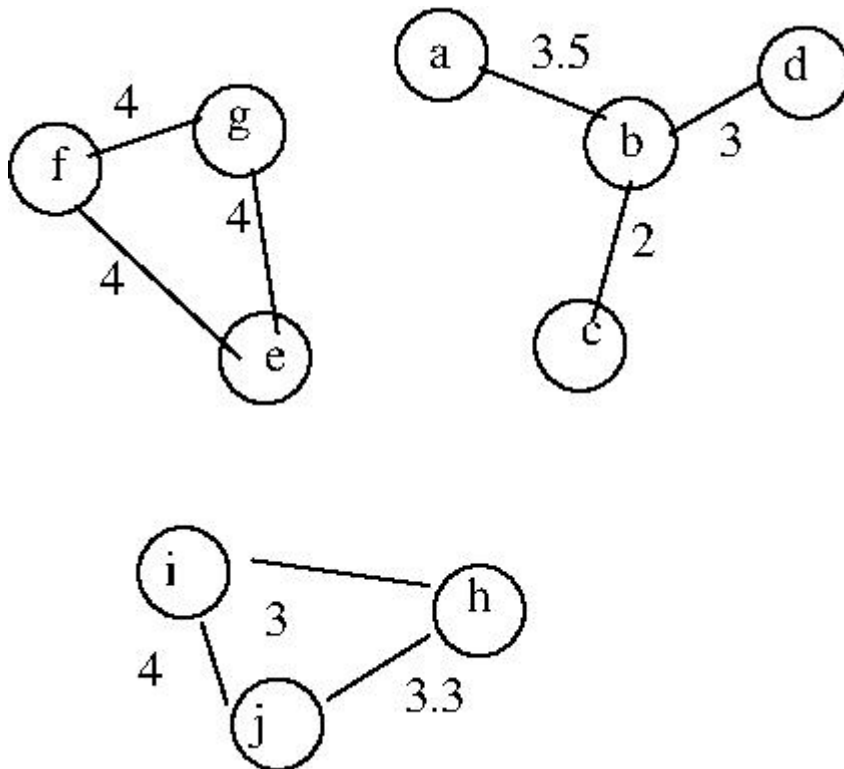
☐

(e,f)



(b,c)

☐

3(b) **Prims och Kruskals algoritm (3p)**

Antag att Prims och Kruskals algoritm appliceras på ovanstående graf med 3 sammanhängande komponenter. Den algoritm som behöver en startnod startar i noden *d*. Resultaten av algoritmerna är ett antal träd.


Hur många noder ingår totalt i det/de träd som blir resultatet av Prims algoritm? (4)

Hur många noder ingår totalt i det/de träd som blir resultatet av Kruskals algoritm? (10)

4(a) Sortering av nästan sorterad sekvens (5p)

Antag att vi har en sekvens som är **nästan sorterad**. Vilken sorteringsalgoritm ska vi använda?

Välj ett alternativ:

- ☐ Quicksort är bättre än mergesort om längden på sekvensen är ett primtal.
- ☐ Att använda quicksort är bättre än att använda mergesort.
- ☒ Att använda mergesort är bättre än att använda quicksort. 
- ☐ Mergesort är bättre än quicksort om längden på sekvensen är ett primtal.
- ☐ Mergesort och quicksort är lika bra, så det spelar ingen roll vilken vi använder.

4(b) Sortering och stabilitet (10p)

Vilka av följande sorteringsalgoritmer är stabila (om någon)? Du får pluspoäng för korrekt svar, minuspoäng för felaktigt svar. Du kan få 0-10p på frågan.

Välj ett eller flera alternativ:

- | | |
|---|---|
| <input type="checkbox"/> Merge sort | ✓ |
| <input type="checkbox"/> Quicksort | |
| <input type="checkbox"/> Insertion sort | ✓ |
| <input type="checkbox"/> Bubble sort | ✓ |
| <input type="checkbox"/> Heapsort | |

5 Operationskategorier graf (15p)

Låt det abstrakta datatypen Graf ha följande informella specifiikation av gränsytan:

- **Empty()** returnerar en tom graf.
- **Insert-node(n, g)** sätter in noden n i grafen g och returnerar den modifierade grafen.
- **Insert-edge(e, g)** sätter in bågen $e=(n_1, n_2)$ från noden n_1 till noden n_2 i grafen g och returnerar den modifierade grafen. Noderna n_1 och n_2 förutsätts ingå i grafen.
- **Isempty(g)** returnerar True of grafen är tom.
- **Has-no-edges(g)** returnerar True om grafen g saknar kanter.
- **Choose-node(g)** returnerar en godtycklig nod i grafen g, under förutsättning att grafen inte är tom.
- **Neighbours(n, g)** returnerar en mängd av kanter som startar i noden n i grafen g.
- **Delete-node(n, g)** tar bort noden n ur grafen g och returnerar den modifierade grafen. Noden n får inte ingå i någon båge i grafen.
- **Delete-edge(e, g)** tar bort bågen e ur grafen g och returnerar den modifierade grafen.

Ange vilken operationskategori varje operator tillhör. Om operatören kan anses tillhöra flera kategorier, välj den längst till höger i tabellen nedan.

	Konstruktor	Inspektor	Modifikator	Navigator	Komparator
Empty()	<input type="radio"/> ✓	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Insert-node(n,g)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> ✓	<input type="radio"/>	<input type="radio"/>
Insert-edge(e,g)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> ✓	<input type="radio"/>	<input type="radio"/>
Isempty(g)	<input type="radio"/>	<input type="radio"/> ✓	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Has-no-edges(g)	<input type="radio"/>	<input type="radio"/> ✓	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Choose-node(g)	<input type="radio"/>	<input type="radio"/> ✓	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Neighbours(n, g)	<input type="radio"/>	<input type="radio"/> ✓	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Delete-node(n,g)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> ✓	<input type="radio"/>	<input type="radio"/>
Delete-edge(e,g)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> ✓	<input type="radio"/>	<input type="radio"/>

6(a) Pseudokod instickssortering (12p)

Använd gränssnittet till Riktad lista till att skriva en funktion i pseudokod som utför instickssortering av en Riktad lista s . Algoritmen ska returnera en sorterad lista t . In-listan s ska lämnas orörd. Algoritmen ska ha följande huvud:

Algorithm InsertionSort(s : Directed list)

Det är möjligt att *modularisera* lösningen, dvs. att identifiera en eller flera delalgoritmer/funktioner och beskriva dem separat. Ge dom i så fall lämpliga namn och kommentarer. Speciellt viktigt är kommentarer som beskriver inparametrar och returvärden. Tänk på att funktioner i pseudokod kan returnera flera parametrar.

Det finns en funktion **IsRelatedTo(a, b)** som returnerar True om värdet a är *relaterat till* värdet b , dvs. att a kommer *före* b i sorteringsordningen. Använd funktionen i din lösning.

Skriv pseudokod för **InsertionSort(s)** i fältet längst ner.

Det som kommer att bedömas är

- att pseudokoden löser uppgiften under de givna förutsättningarna,
- att koden är korrekt indenterad,
- att koden är rimligt kommenterad,
- om koden är modulariserad, och
- om koden har optimal komplexitet ($g(n)$ är viktigast).

Skriv din kod här:

1	
---	--

```

Algorithm FindPosition(v: Value, s: Directed list)
// Return the position in the list s where v should be inserted to
// match the sorting order.

p ← First(s)
while not Isend(p, s) do
    if IsRelatedTo(v, Inspect(p, s)) then
        // We found the position
        return p

    // Otherwise, advance in the list
    p ← Next(p, s)

// We've reached the end.
// v is not related to any element in s and should be inserted last
return p

Algorithm InsertionSort(s : Directed list)

// Start with empty output list
t ← Empty()

// Iterate over input list
p ← First(s)
while not Isend(p, s) do
    // Get the value
    v ← Inspect(p, s)
    // Find its correct position
    q ← FindPosition(v, t)
    // Insert it there
    (t, q) ← Insert(v, q, t)
    // Advance in input list
    p ← Next(p, s)

// Return the sorted list
return t

```

6(b) Frågor instickssortering (8p)

Denna fråga gäller din lösning på förra frågan:

Är din lösning stabil?

☐ Ja



☐ Nej

Nedanstående frågor gäller Instickssortering i allmänhet:

Vilken komplexitet har Instickssortering i medelfallet?

☐ $O(n)$

☐ $O(\log n)$

☐ $O(n \log n)$

☐ $O(n^2)$



Vilken komplexitet har Instickssortering på en redan sorterad lista?

☐ $O(\log n)$

☐ $O(n^2)$



☐ $O(n \log n)$

☐ $O(n)$

Vilken komplexitet har Instickssortering på en omvänt sorterad lista?

☐ $O(\log n)$

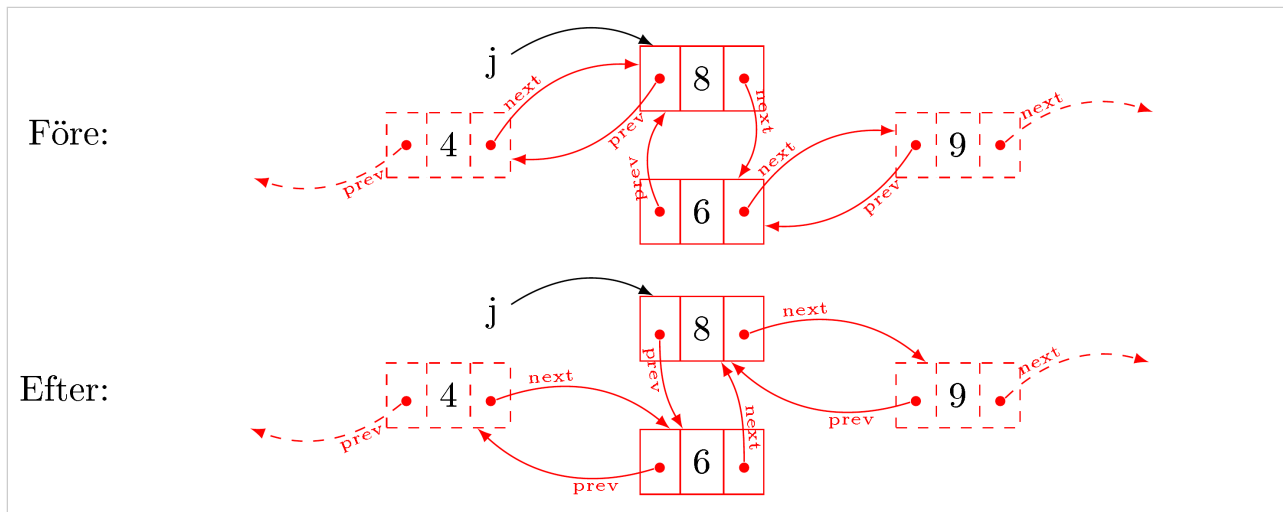
☐ $O(n \log n)$

☐ $O(n^2)$



☐ $O(n)$

7 Länkade strukturer (15p)



Vi vill byta plats på två element i en dubbel-länkad lista genom att uppdatera referenserna/pekarna i 2-cellerna som utgör elementen. De två elementen som ska byta plats är det som variabeln `j` refererar till och dess efterträdare. Det önskade före/efter-tillståndet illustreras i figuren ovan.

Koden nedan är ett försök att implementera detta. Om `e` är en 2-cell eller en referens till en sådan så refererar `e.next` till framåt-länken och `e.prev` till bakåtlänken. Tecknen "`<-`" står för tilldelning.

```
1. j.prev.next <- j.next
2. j.next.prev <- j.prev
3. j.next.next.prev <- j.prev
4. j.next <- j.next.next
5. j.prev <- j.prev.next
6. j.prev.next.next <- j
```

Tyvärr innehåller koden två fel: Dels innehåller en av kodraderna fel text, dels har två kodrader bytt plats. Din uppgift är att identifiera och korrigera felen så att koden fungerar.

a) Vilken del av koden innehåller textfelet? (Rad 1, vänsterledet, Rad 1, högerledet, Rad 2, vänsterledet, Rad 2, högerledet, Rad 3, vänsterledet, **Rad 3, högerledet**, Rad 4, vänsterledet, Rad 4, högerledet, Rad 5, vänsterledet, Rad 5, högerledet, Rad 6, vänsterledet, Rad 6, högerledet)

b) Vad ska den felaktiga texten bytas ut mot? Om flera alternativ är tänkbara, välj det kortaste.

(**j**, `j.next`, `j.prev`, `j.next.next`, `j.next.prev`, `j.prev.next`, `j.prev.prev`, `j.next.next.prev`, `j.prev.next.next`)

c) Efter att den felaktiga texten bytts ut, vilka två rader måste då byta plats för att koden ska fungera? (1 och 2, 2 och 3, 3 och 4, 4 och 5, **5 och 6**)

8 Förtydliganden (inga poäng)

Om du anser att du behöver förtydliga ditt svar på någon/några frågor kan du skriva det här.
Annars lämnar du fältet tomt.

Teckenf... ▾ | **B** *I* U x_2 x^2 | $\frac{1}{x}$ | | | | Ω | Σ |

Ord: 0