Tentamen - Datastrukturer, algoritmer och programkonstruktion.

DVA104

Akademin för innovation, design och teknik

Måndag 2014-06-02

Skrivtid: 14.10 – 19.30
Hjälpmedel: Lexikon/Ordbok
Lärare: Caroline Uppsäll - 021-101456

Betygsgränser

Betyg 3:	26p (varav minst 7p P-uppgifter)
Betyg 4:	37p
Betyg 5:	45p
Max:	**50p (varav 14p är P-uppgifter)**

- Skriv endast en uppgift per blad
- Skriv bara på ena sidan av pappret.
- Referera inte mellan olika svar.
- Om du är osäker på vad som avses i någon fråga, skriv då vad du gör för antagande.
- Oläsliga/Oförståeliga svar rättas inte.
- Kommentera din kod!
- Uppgifter som är markerade som P-uppgifter måste lösas med kod i Ada (C eller C#). Övriga uppgifter behöver inte lösas med kod.
- Tips: Läs igenom hela tentan innan du börjar skriva för att veta hur du ska disponera din tid.
- Endast bonuspoäng som intjänats under instansen VT14 period 2 kan användas på denna tentamen.
- Intjänade bonuspoäng räknas till totalpoängen och inte till P-uppgifterna

Lycka till!

Uppgift 0

Läs instruktionerna på förstasidan noggrant. Följ dessa genom hela tentamen!

Uppgift 1 *(11p)*

a) Varför är det viktigt att hålla isär en ADTs implementation och interface? (1p)
b) Vad innebär det att ett binärt träd är fullt? (1p)
c) Nämn fyra olika saker som måste finnas i en funktion för att den ska vara rekursiv samt varför dessa fyra är viktiga. (2p)
d) Vilka förutsättningar finns på mängden för att man ska kunna applicera binärsökning på den? (1p)
e) När är länkad lista respektive array att föredra, annat än beroende på om vi vet storleken på den mängd som ska lagras sekventiellt eller inte? (1p)
f) Vad innebär det att en algoritm är stabil? (1p)
g) Ge ett exempel på en sorteringsalgoritm som är in-place och som är naturlig. (1p)
h) Varför vill du kunna mäta en algoritms komplexitet? (1p)
i) Vad innebär det att en kö arbetar enligt principen FIFO? (1p)
j) Vad innebär det att en kö är cirkulär? (1p)

Uppgift 2 *(3p)* P-uppgift

Skriv en funktion som rekursivt beräknar värdet av n!. Funktionen ska ta emot talet n och returnera det efterfrågade resultatet.

Definition: n! = 1 * 2 * 3 * .. * n

Uppgift 3 *(4p)* P-uppgift

Antag att du har en nod (skriven i pseudokod):

```
Node
   Integer data
   Node next
   Node previous
```

Samt en länkad lista (skriven i pseudokod), där variabeln counter håller reda på hur många noder som finns i listan:

```
LinkedList
   Node head
   Node tail
   Integer counter
```

Vi kan anta att länkar som inte har en specifik nod att peka på är satta till NULL samt att en tom lista har counter 0.

Skriv funktionen som lägger till en nod på en angiven position i den länkade listan. Om positionen ska anges som ett heltal eller som en pekare till en nod i listan väljer du själv.

Funktionen ska inte returnera någonting. Den nya noden (som du kan anta är skapad korrekt) samt positionen ska tas in som parametrar till funktionen, du kan också behöva ytterligare inparametrar.

Uppgift 4 *(2p)*

Ange tidskomplexitet (komplexitetsklass) på O-notation i värsta fallet i avseende på n för följande två funktioner. – Förklara.

a) (1p)

```
foo(Integer arr1[], Integer arr2[], Integer n)
    Integer i := 0
    While i < n do
        arr1[i] := 0
        i := i + 1
    End loop
    Integer j := 0
    While j < 2*n do
        arr2[j] := 1
        j := j + 1
    End loop
```

b) (1p)

```
foo(Integer arr1[], Integer arr2[], Integer n)
    Integer i := 0, j := 0
    While i < n do
        arr1[i] := i*i
        While j < n*n do
            arr2[j] := i*j
            j := j + 1
        End loop
        i := i + 1
    End loop
```

Uppgift 5 *(5p)*

Omvänd polsk notation, eller postfix notation, är en metod som gör det möjligt att skriva aritmetiska uttryck (alltså räkneuttryck) utan att använda parenteser. Detta görs genom att skriva operatorn (så som +, - , *, /) efter operanderna.

Exempel:

Vanligt räkneuttryck	Postfix
a + b	ab+
a + b * c	abc*+
(a + b) * c	ab+c*
(a – (b + c)) * (c – d)	abc+-cd-*

a) För att beräkna aritmetiska uttryck i postfix notation (omvänd polsk notation) används datastrukturen stack. Visa och beskriv hur du med hjälp av en stack beräknar det postfixa uttrycket: abc+-cd-*. Du kan anta att uttrycket är lagrat i t.ex. en sträng. Du behöver inte beskriva hur du typkonverterar. (3p)
b) Hur kan vi via stacken i din lösning i a) veta om vi stött på ett felaktigt eller korrekt angivet postfixt uttryck? (2p)

Uppgift 6 *(4p)*

a) Visa med hjälp av bilder hur följande sekvens av data sorteras med Mergesort (3p)

5	8	3	4	9	1	6	7

b) Förklara i vilket/vilka fall algoritmen Quicksort är mindre effektiv än Mergesort. – Motivera (1p)

Uppgift 7 *(4p)*

Välj en sorteringsalgoritm som inte använder principen Divide & Conquer.

a) Beskriv i ord och/eller bild hur algoritmen sorterar en mängd sekventiell mängd. (1p)
P-uppgift b) Skriv koden för den valda algoritmen. (3p)

Uppgift 8 *(5p)*

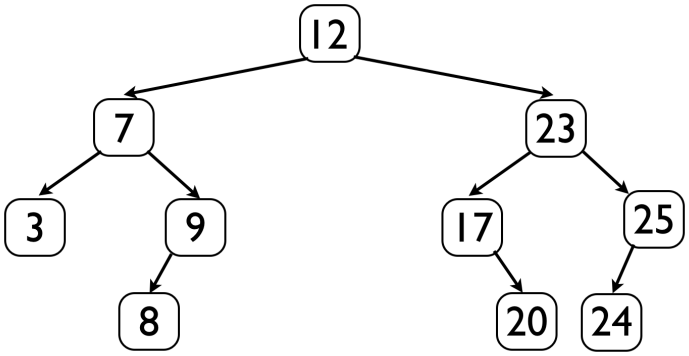
Antag att du har en hashtabell med plats för 8 element- Hashfunktionen är $h(x) = (x \bmod 8)$

Modulus ger resten vid heltalsdivision, ex: 5 mod 8 = 5, 8 mod 8 = 0, 15 mod 8 = 7, 16 mod 8 = 0

a) Antag att tabellen använder chained-strategin (länkad lista) för att hantera krockar. Visa hur hashtabellen ser ut när följande element satts in i ordning: (1,5p)
   3 13 9 27 8 32 5
b) Antag nu istället att tabellen använder öppen adressering för att hantera krockar. Visa hur hashtabellen ser ut när samma sekvens som i a) sätts in (i samma ordning). (1,5p)
c) Om man vill söka efter ett data i en stor mängd. När bör man använda hashtabell och när behöver man använda binärt sökträd? – Diskutera. (2p)

Uppgift 9 *(12p)*

Antag följande träd:



a) Är trädet ovan ett balanserat binärt sökträd? - Motivera (1,5p)
b) Skriv ut trädet i alla tre traverseringsordningarna vi gått igenom i kursen. Namnge också vilken utskrift som är vilken traversering. (1,5p)
c) Vilken är komplexiteten för insättning i ett balanserat samt oblanserat binärt sökträd i bästa respektive sämsta fall? – Motivera. (1p)

P-uppgift d) Antag följande träd och nod:

```
BTree
     Node root

Node
     Integer data
     Node leftChild
     Node rightChild
```

Skriv funktionen som söker efter ett visst data i trädet. Du kan anta att trädet är sorterat och att länkar som inte pekar på en nod är satta till NULL. Funktionen ska returnera en pekare till den eftersökta noden om den finns i trädet eller NULL om datat inte finns. Datat (heltal) som ska eftersökas ska skickas in som parameter till funktionen. Du väljer själv om du vill göra en rekursiv eller en iterativ funktion. (4p)

e) Beskriv vilka olika fall det finns för borttagning av en nod i ett sorterat binärt träd samt vad man måste tänka på i de olika fallen och hur borttagningen går till. (2p)
f) Beskriv två olika sätt att balansera ett binärt sökträd på. Det räcker inte att endast ange namnet på balansering utan en beskrivning av hur balanseringen går till måste ges. (2p)