Tentamen - Datastrukturer, algoritmer och programkonstruktion.

DVA104

Akademin för innovation, design och teknik

Måndag 2017-08-14

Skrivtid: 14.30-19.30
Hjälpmedel: Handskrivna anteckningar (obegränsad mängd) samt ordbok/lexikon
Lärare: Caroline Uppsäll (är inte anträffbar under tentamenstillfället)

Betygsgränser

Betyg 3: 19p
Betyg 4: 26p
Betyg 5: 31p
Max: 35p

Allmänt

- Kod skriven i tentauppgifterna är skriven i C-kod.
- På uppgifter där du ska skriva kod ska koden skrivas i C.
- Markera tydligt vilken uppgift ditt svar avser.
- 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!
- Inga bonuspoäng är aktuella vid omtentamen.
- Tips: Läs igenom hela tentan innan du börjar skriva för att veta hur du ska disponera din tid.

Lycka till!

Uppgift 1: *(7p)*

1) Nämn två saker, förutom det rekursiva fallet, som behövs för att en rekursiv funktion ska vara korrekt. *(1p)*

2) 64 studenter är antagna till en utbildning. Om studenternas namn är lagrade i en array, hur många studenter måste då algoritmen linjärsökning titta på för att leta reda på en specifik student (i värsta fall). Motivera ditt svar med **en** mening **och/eller** illustration – *för poäng krävs både korrekt svar och en korrekt motivering. (1p)*
    1) Max 32
    2) Max 64
    3) Max 7
    4) Max 1

3) Vad innebär det att en algoritm har komplexiteten $O(n^2)$? *(1p)*

4) En skillnad mellan en kö och en stack är…? *(1p) – motivera ditt svar med en mening.*
    1) En stack kräver en länkad lista, en kö gör inte det
    2) En kö kräver en länkad lista, en stack gör inte det
    3) En stack använder båda ändarna på strukturen, en kö gör inte det
    4) En kö använder båda ändarna på strukturen, en stack gör inte det

5) Vad innebär det att en algoritm är in-place? Ge exempel på en sorteringsalgoritm som inte är in-place. *(1p)*

6) Vilken av nedanstående strukturer är mest effektiv för implementationen av ett set? *(1p) – glöm inte att motivera ditt svar.*
    1) Länkad Lista
    2) Array
    3) Cirkulär kö
    4) Binärt sökträd

7) I arrayimplementationen av en stack, vilka av operationerna nedan kräver linjär tidskomplexitet (i värsta fallet)? *(1p) – Glöm inte att motivera dina svar (Felaktigt angivet svar ger poängavdrag på deluppgiften).*
    1) is_empty
    2) peek
    3) pop
    4) push (antag att det finns plats kvar i stacken)
    5) Ingen av ovanstående operationer kräver linjär tidskomplexitet

Uppgift 2: *(2p)*

Skriv en rekursiv funktion som tar en heltalsvariabel x som inparameter. Funktionen ska skriva ut x utropstecken. Du får inte använda några loopar eller andra variabler än x.

Uppgift 3: *(6p)*

a) Skapa en nod för en enkellänkad lista, datat ska bestå av ett heltal. *(1p)*
b) Skapa nu en lista med pekare för både head och tail. Head ska hela tiden peka på första noden i listan och tail på den sista. Från början är listan tom. *(1p)*
c) Skriv en funktion som tar bort en nod från listan. Funktionen ska ta datat som ska tas bort som inparameter, leta upp noden innehållande datat och ta bort den från listan. Övriga noder ska vara oförändrade. Den uppdaterade listan ska på något sätt returneras. Du kan anta att listan som operationen appliceras på är en korrekt lista. *(4p)*

Uppgift 4: *(8p)*

a) Rita ett fullt binärt sökträd med minst 9 noder (innehållande heltal) (*1p)*
b) Du ska nu jobba med trädet du ritade i deluppgift a (har du inte ritat ett korrekt fullt binärt sökträd i a kommer deluppgiften rättas med det trädet du ritat). (3p)
    1) Ange vilka noder som är löv i trädet
    2) Skriv ut trädet i postorder (LR)
    3) Ta bort rootnoden och rita om trädet (det ska fortfarande vara ett binärt sökträd).

c) Antag implementationen:

```
struct treeNode
{
      int data;
      struct treeNode* left;
      struct treeNode* right;
};

typedef struct treeNode* BSTree;
```

Skriv en funktion som summerar och returnerar innehållet i alla noder med två barn *(4p)*

Uppgift 5: *(3p)*

För vardera av funktionerna nedan. Ange vilken komplexitetsklass de ska placeras i (med avseende på n).

De komplexitetsklasser du har till ditt förfogande är följande (ej sorterade i ordning): $O(2^n)$, $O(n)$, $O(\log_2 n)$, $O(1)$, $O(n^3)$, $O(n \log_2 n)$, $O(n^2)$

a)

```
void funkA(int n, int x)
{
    for(int k = 0; k < x; ++k)
    {
        if(n > 50)
        {
            for(int i = 0; i < n; ++i)
                for(int j = 0; j < i; ++j)
                    printf("x = %d\n", x);
        }
    }
}
```

b)

```
int funkB(int n, int m)
{
    if(n < 1)
        return m;
    else if(n > 10)
        return silly(n/2, m);
    else
        return silly(n-2, m);
}
```

c)

```
void funkC(int n)
{

    for(int j = 1000; j > 0; j = j/2)
        printf("j = %d\n", j);
    for(int m = 0; m < 5000; m++)
        printf("m = %d\n", m);
}
```

Uppgift 6: *(4p)*

Antag att du har en fil uppbyggd på samma format som exemplet nedan. Din uppgift är att lagra datat i filen i en struktur så att du effektivt kan söka i den (sökningen ska ligga så nära O(1) som möjligt). Sökningen sker baserat på ID och allt data i filen ska finnas med i strukturen. Varje data i filen består av ett ID (heltal), en timestamp (heltal) och ett state (UP eller DOWN). Samma ID kan förekomma flera gånger i filen med olika timestamps och samma eller olika state, det ska då finnas med flera gånger i din struktur.

Beskriv med bilder och text hur du skulle lagra datat, vilken struktur skulle du använda och hur skulle den se ut. Beskriv också varför din tänka lösning är effektiv för sökning. Du behöver inte beskriva hur du läser datat från filen utan endast i vilken typ av struktur det ska lagras för effektiv sökning. Tänk på att du kan kombinera strukturer.

Exempel på filen

ID	Timestamp	State
101	2341	UP
439	12351	UP
509	2375	UP
89	4325	UP
439	54232	DOWN
89	5261	DOWN
439	54238	UP
101	3254	DOWN

Uppgift 7: *(2p)*

Följande array skickas som indata till en sorteringsalgoritm

5	9	7	2	4	8	3

Vid något tillfälle under sorteringen ser arrayen ut så här:

2	5	7	9	4	8	3

Vilken algoritm har använts? Insertion Sort (insättningssortering), Selection Sort (Urvalssortering) eller Bubble Sort? Motivera ditt svar – *för poäng krävs både korrekt svar och en korrekt motivering. (2p)*

Uppgift 8: *(3p)*

Du ska nu jobba med QuickSort och Selection Sort (Urvalssortering). För vardera av dessa, vilken är bästa fallets komplexitet om du vet att…

a) Indata redan är sorterat och första elementet väljs som pivot
b) Indatat är helt osorterat och mittersta elementet väljs som pivot
c) Indatat är en linjär sekvens innehållande n kopior av samma data och ett slumpvis element väljs som pivot

Du ska för varje case ovan och **för vardera** av algoritmerna motivera/förklara med **en** mening – *för poäng krävs både korrekt svar och en korrekt motivering. (3p)*