Tentamen i Datastrukturer och algoritmer, fk, DVA246

Akademin för innovation, design och teknik

2023-08-17

Skrivtid: 14.30 – 18.30
Hjälpmedel: Inga hjälpmedel
Lärare: Daniel Hedin kan nås på telefon 15.00 - 16.00

Preliminära betygsgränser
Betyg 3: 50%
Betyg 4: 70%
Betyg 5: 90%
Max: 46p

Tentamen består av 9 uppgifter (observera att vissa av dessa består av deluppgifter)

Krav och allmänna råd:

- Läs igenom hela tentamen för att veta hur du skall disponera din tid
- Frågorna står inte i svårighetsordning
- Skriv tydligt vilken uppgift du svarar på
- Ordna svaren i rätt ordning med svaret på uppgift 1 först och svaret på uppgift 8 sist
- Om du är osäker på vad som avses och gör antaganden skriv ut vilka dessa antaganden är.
- Det går att få poäng för partiella lösningar givet att de är meningsfulla

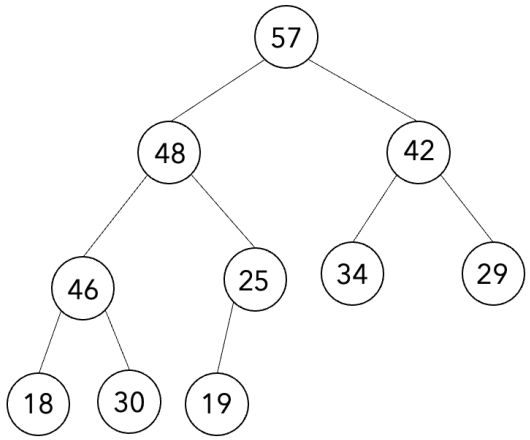
Uppgift 1: Korrekthet (6p)

Bevisa att följande algoritm som summerar alla tal i en array är korrekt. Givet att själva algoritmen är trivial är det noga att varje steg i ditt resonemang är formellt och korrekt och att ditt bevis följer den av kursboken givna strukturen.

```
int sum(int[] xs) {
      var total = 0;
      for (var i = 0; i < xs.Length; i++) {
            total += xs[i];
      }
      return total;
}
```

Uppgift 2: Heap (2+2p)

I båda deluppgifterna nedan ska du utgå från följande heap.



Deluppgift 1

Sätt in nyckel 55 i heapen ovan. Rita om heapen (visualiserad som en trädstruktur) till så som den ser ut efter att insättningen är färdig. Skriv endast nycklar i de noder som på något vis ändras på grund av insättningen av 55, de noder som inte ändras ritas ut utan innehåll.

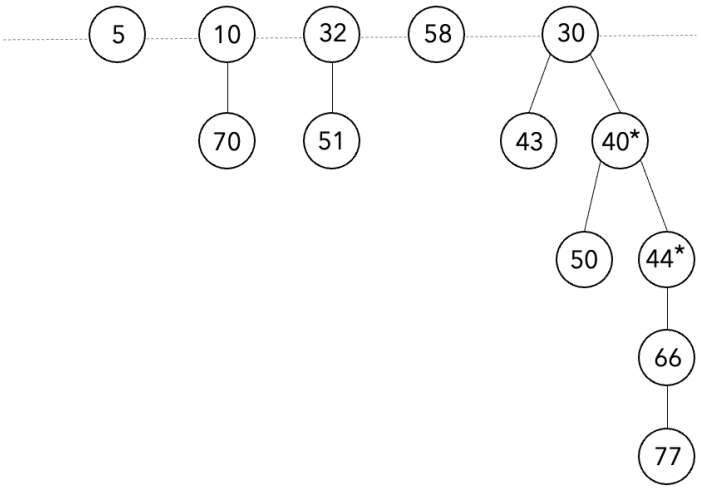
Deluppgift 2

Utgå från heapen ovan (innan deluppgift 1) och visa hur den ser ut representerad som en array i varje steg som gör någon förändring med heapen då operationen extractMax() genomförs.

Uppgift 3: Fibonacci-heap (4+3p)

Deluppgift 1

Antag Fibonacci-heapen nedan, noder med en * är markerade som vad vi i kursen kallar för ”loser”-noder. Utför extractMin() på heapen följt av decreaseKey(66, 4) som sänker nyckelvärdet 66 till 4. Rita hur heapen ser ut då den förändras i vardera steg (operation). Ytterligare steg får givetvis läggas till om du anser att det gör din lösning tydligare. Lägg till text för att förtydliga.



Deluppgift 2

Fibonacci-heapar är designade på ett sätt så att de aldrig blir överdrivet breda och grunda (i sin trädrepresentation). Vad i designen av fibonacci-heapar är det som leder till denna egenskap? Förklara.

Uppgift 4: Röd-svarta träd (3p)

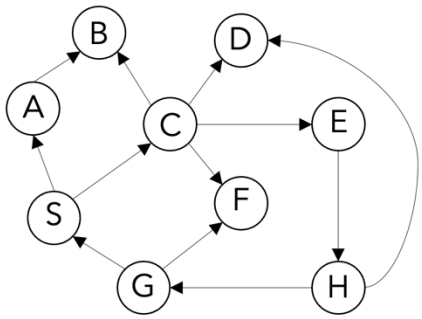
Nedan hittar du ett antal påståenden om röd-svarta träd. För vardera påstående ska du svara om påståendet är sant eller falskt. Om du svarar att ett påstående är falskt behöver du också ange en kort förklaring.

a) Röd-svarta träd är ett exempel på självbalanserande binära sökträd.
b) Noderna i ett röd-svart träd tilldelas en färg vid insättning och får sedan aldrig byta färg.
c) När en nod sätts in i ett röd-svart träd så sätts det alltid in längst till vänster på den lägsta nivån i trädet.
d) En av reglerna för röd-svarta träd är att en röd nod aldrig får ha röda barn.
e) I ett röd-svart träd skiljer djupen på två vägar genom ett delträd (från root till NIL-nod) med max 1 nod.
f) Roten i ett röd-svart träd är alltid svart.

Uppgift 5: Grafer (4+3p)

Deluppgift 1

Visa en topologisk sortering av grafen nedan. Börja i nod S. Vid val mellan vägar ska de väljas i alfabetisk ordning. Om det finns några forward-bågar så ska de tydligt markeras. Om det finns några back-bågar så ska de utelämnas från den topologiska sorteringen. Cross-bågar ska ritas ut men du behöver inte markera att de är just cross-bågar.
Tips: gör först en djupet-först sökning av grafen och logga nodernas tidsstämplar.



Deluppgift 2

Förklara på vilket sätt Dijkstra's algoritm tar hjälp av en prioritetskö. Samt beskriv varför vi inte kan garantera att den genererar ett korrekt single-source shortest path om grafen innehåller negativt viktade bågar (även om de inte skapar negativa cykler). Det går alldeles utmärkt att komplettera dina förklaringar med bilder och exempel.

Uppgift 6: Giriga algoritmer (4p)

Växlingsproblemet är det optimeringsproblem som uppstår när vi vill nå en viss summa givet mynt av olika valörer med så få mynt som möjligt.

När vi växlar till mynt använder vi normalt den giriga algoritmen där vi tar så många mynt som möjligt av den högsta valören som är möjlig för att sedan applicera samma algoritm på den resterande summan. Myntsystem där denna algoritm ger en optimal lösning kallas kanoniska myntsystem. Visa att myntsystemet med mynten 1, 4 och 9 inte är kanoniskt.

Uppgift 7: Optimal delstruktur (4p)

Bevisa att växlingsproblemet beskrivet ovan har optimal delstruktur.

Uppgift 8: Rekursionsträd (4+2p)

Ge en rekursiv algoritm för växlingsproblemet och visa rekursionsträdet för myntsystemet i uppgift 6 och summan 12. Algoritmen skall ta in en array som beskriver vilka myntvalörer som finns samt en summa och returnera en representation av hur många av varje mynt som krävs för att uppnå summan.

Missa inte sista uppgiften på nästa sida!

Uppgift 9: Dynamisk programmering (5p)

Ge en iterativ dynamisk algoritm som löser växlingsproblemet även för icke-kanoniska myntsystem. Algoritmen skall ta in en array som beskriver vilka myntvalörer som finns samt en summa och returnera en representation av hur många av varje mynt som krävs för att uppnå summan.