Tentamen i Datastrukturer och algoritmer, fk, DVA246

Akademin för innovation, design och teknik

2024-01-11

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: 47p

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 är korrekt, dvs. att den returnerar det största elementet i heltalsarrayen *A*.

```
Max(A)
n = A.length
v = A[0]
for i = 1 to n - 1
    if A[i] > v
        v = A[i]
return v
```

Anta att *A* innehåller minst ett element och indexeras från 0.

I ditt bevis ska du definiera en lämplig loop-invariant och visa att:

1. Loop-invarianten gäller före den första iterationen (*initialization*).
2. *Om* loop-invarianten gäller i början på iteration *i* = *j*, där *j* ∈ { 0, 1, …, *n*-1 }, *så* gäller den även i början på iteration *i* = *j* + 1 (*maintenance*).
3. Efter att loopen har terminerat (du kan betrakta detta som iteration *i* = *n*) följer algoritmens korrekthet direkt av loop-invarianten (*termination*).

Uppgift 2: Heap (3p)

Det vi har pratat om i kursen är binära heapar (varje nod har maximalt 2 barn i en trädvisualisering av heapen). Vi behöver dock inte begränsa oss till just binära heapar utan kan arbeta med d-ary heapar, alltså heapar där varje nod har maximalt d barn. En sådan d-ary heap är en ternary heap där varje nog har maximalt 3 barn. I en ternary heap hittar vi de tre barnen till noden på index i på position (i*3)+1, (i*3)+2 respektive (i*3)+3 för en nollindexerad array. Föräldern till noden på index i hittar vi på position $\lfloor (i-1)/3 \rfloor$.

Rita upp hur följande ternary heap skulle visualiseras som ett träd
[92, 82, 79, 76, 46, 1, 49, 44, 61].

Är det en max-heap, en min-heap eller bryts heap-reglerna?

Uppgift 3: Fibonacci-heap (4+1+2p)

Deluppgift A

Antag en från början tom Fibonacci-heap. Utför följande sekvens av operationer (uppifrån och ner, en i taget) och visa hur den resulterande heapen ser ut. Heapen är en min-heap. I sekvensen nedan så står det var du ska visa hur fibonacci-heapen ser ut samt hur du ska namnge bilden.

Insert(27)
Insert(17)
Insert(19)
Insert(20)
Insert(24)
Insert(12)
Insert(11)
Insert(10)
Insert(14)
Insert(18)
ExtractMin()
Visa hur fibonacci-heapen ser ut här (Fibonacci-heap 1)
DecreaseKey(19, 7)
Visa hur fibonacci-heapen ser ut här (Fibonacci-heap 2)
ExtractMin()
DecreaseKey(20, 10)
Visa hur fibonacci-heapen ser ut här (Fibonacci-heap 3)

När noder läggs till i root-listan så läggs de längst till höger. Sammanslagning görs från höger till vänster

Deluppgift B

Antag att 139 element sätts in i en från början tom fibonacci heap (inga decreaseKey eller extractMin görs), hur många träd finns det i heapen efter att insättningarna gjorts? Motivera/förklara ditt svar med max 2 meningar.

Deluppgift C

Vilken degree har det resulterande trädet om två träd av degree k slås ihop i en fibonacci heap?

Uppgift 4: Röd-svarta träd (4+2p)

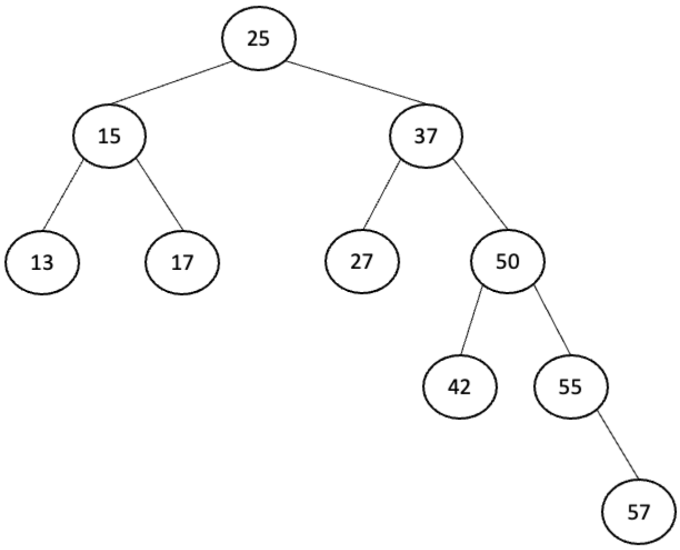
Deluppgift 1

Förklara vad ett röd-svart träd är, hur det fungerar samt vad det är bra för. Förklaringen bedöms baserat på dess korrekthet och utförlighet. Max 10 meningar.

Deluppgift 2

Denna deluppgift tittar endast på själva rotationen.

Visa vilken eller vilka rotationer som behöver göras i följande träd:



Uppgift 5: Grafer (2+3p)

Deluppgift 1

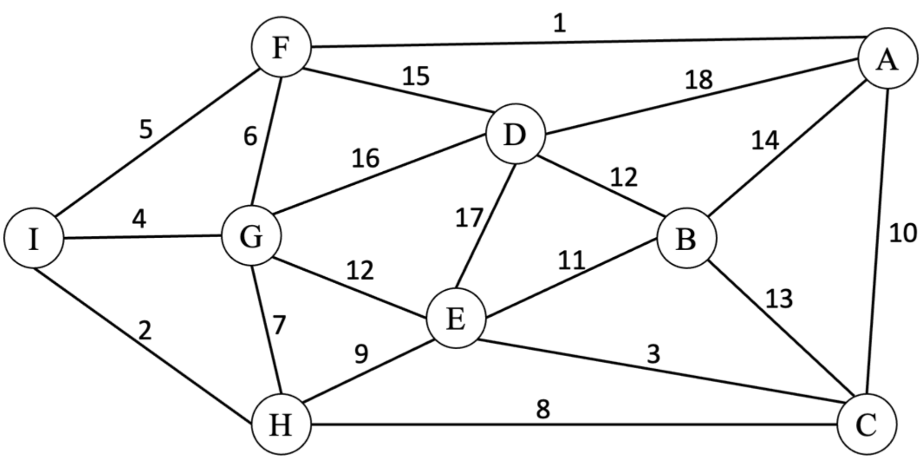
Vilka av nedan påståenden om grafer är sanna. Varje felaktigt angivet svar kommer att kvittas mot korrekt angivet svar. Minsta totalpoäng för deluppgiften är 0p.

a) Dijkstra's algoritm är greedy men Bellman-ford är inte det.
b) Ett spanningträd innehåller alla noder i grafen men endast V-1 bågar (där V är antalet noder).
c) En bredden-först sökningen ger oss den kortaste vägen (i antal bågar) från startnoden till varje annan nod.
d) Ett spanningträd kan innehålla cykler.
e) Djupet-först sökning kan användas för att göra en topologisk sortering.
f) Varken Kruskal's eller Prim's algoritm klarar av att hantera negativa vikter.
g) Om en graf innehåller en negativ cykel så kan vi inte hitta ett shortest path träd i grafen.
h) Det kan aldrig existera mer än ett shortest path träd i en graf.
i) En riktad graf måste också alltid vara viktad.

Deluppgift 2

Visa hur ett Minimum Spanning Tree framtaget med hjälp av Kruskal's algoritm ur nedanstående graf ser ut.

Skriv också i vilken ordning du lagt till de olika bågarna till spanningträdet (Om du t.ex. har en båge mellan noderna X och Y med vikt 5 som du lägger till så skriver du "X – Y: 5").



Uppgift 6: Giriga algoritmer (4p)

Skapa ett Huffman-träd för strängen abracadabra!

Uppgift 7: Optimal delstruktur (2+2p)

Förklara vad optimal delstruktur betyder och namnge två typer av algoritmer vi har talat om som fungerar enbart för problem med optimal delstruktur.

Uppgift 8: Rekursionsträd (4+2p)

Ge en rekurrensformeln för en naiv implementation av Fibonacci och använd rekursionsträd för att uppskatta en rimlig asymptotisk övre begräsning av algoritmen.

Uppgift 9: Dynamisk programmering (4+2p)

Anta att $c_1 < c_2 < \ldots c_n$ där $c_1 = 1$ är valörer på mynt. Skapa en rekursiv algoritm som beräknar det minsta antal mynt som behövs för att nå en viss summa t. Du kan anta att det finns godtyckligt många mynt av varje valör. Skulle din algoritms exekveringstid förbättras av dynamisk programmering? Argumentera för eller emot.