Skriftlig Eksamen
Datastrukturer og Algoritmer (DM02)

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

Mandag den 31. januar 2000, kl. 9–13

Alle sædvanlige hjælpemidler (lærebøger, notater, etc.) samt brug af lommeregner er tilladt.

Eksamenssættet består af 4 opgaver på 7 nummererede sider (1–7). Fuld besvarelse er besvarelse af alle 4 opgaver.

De enkelte opgavers vægt ved bedømmelsen er angivet i procent. Der må gerne refereres til algoritmer og resultater fra lærebogen inklusive øvelsesopgaverne. Henvisninger til andre bøger (udover lærebogen) accepteres ikke som besvarelse af et spørgsmål.

Bemærk, at hvis der er et spørgsmål i en opgave, man ikke kan besvare, må man gerne (så vidt det er muligt) besvare de efterfølgende spørgsmål og blot antage, at man har en løsning til de foregående spørgsmål.

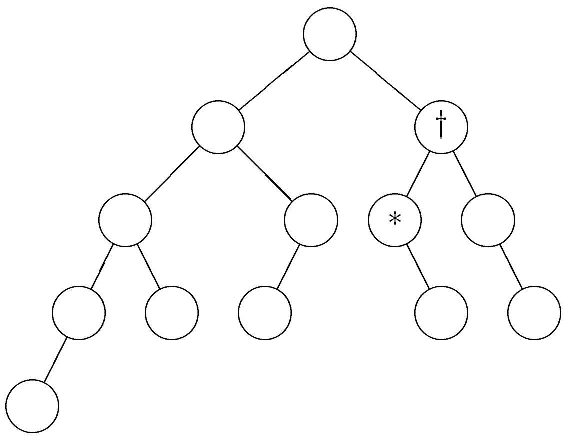
Opgave 1 (30%)

Denne opgave drejer sig om binære træer og en ny måde, hvorpå de kan holdes rimeligt balanceret.

I modsætning til Kingston tegner vi i denne opgave også bladene, når vi tegner træer. Bladene er de knuder, der ikke har nogen børn. Vi definerer, at et blad har højde 0, og som sædvanligt er højden af en intern knude defineret som én plus maximum af højderne af knudens børn. Højden af et træ er højden af træets rod. Et træ bestående af kun én knude har altså højde 0.

Af hensyn til næste definition siger vi, at et tomt træ (et træ med 0 knuder) har højde -1.

Et *højdebalanceret træ* defineres nu til at være et binært træ, hvor der for alle knuder gælder, at højdeforskellen på knudens venstre og højre undertræer er højst 1. Følgende er et eksempel på et højdebalanceret træ:



F.eks. har roden undertræer af højde 3 og 2, knuden mærket † har undertræer begge af højde 1, og knuden mærket $*$ har undertræer af højde -1 og 0.

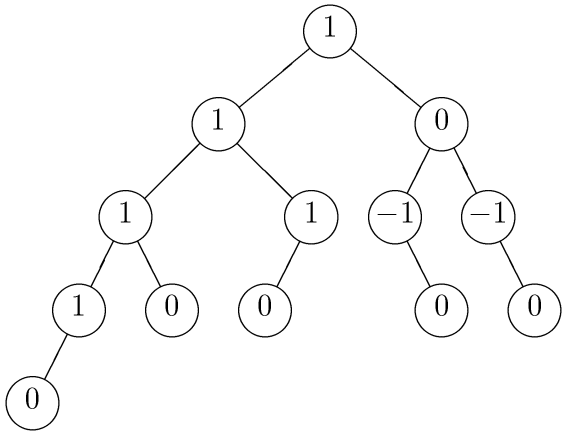
Spørgsmål a: Tegn for hver af højderne $h = 0, 1, 2, 3, 4$ et mindste højdebalanceret træ (dvs. færrest mulige knuder) med højde h. Der skal altså tegnes 5 uafhængige træer i dette spørgsmål. Angiv også antallet af knuder i hvert af de fem træer. □

Spørgsmål b: Vis ved induktion, at der er mindst $2^{\frac{h}{2}}$ knuder i et højdebalanceret træ af højde $h \geq 0$.

Vink: Hvad er det største h' (udtrykt ved h), for hvilket man kan garantere, at begge rodens undertræer har højde mindst h'? □

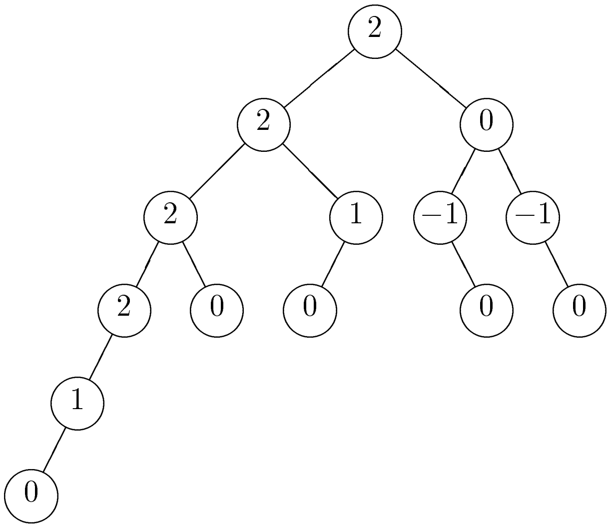
Spørgsmål c: Vis, at hvis et højdebalanceret træ af højde h indeholder n knuder, så gælder der $h \in O(\log n)$. □

Vi udstyrer nu knuderne med et felt, *diff*, hvor vi noterer værdien: *højden af venstre undertræ minus højden af højre undertræ*. Da træet er højdebalanceret skal disse værdier være -1, 0 eller 1. For eksemplet ovenfor ville vi få:

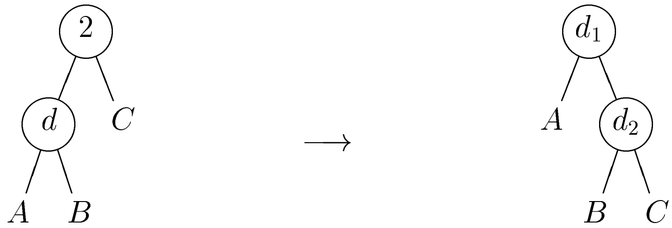


Vi er nu interesseret i at kunne føje nye knuder til et højdebalanceret træ. En ny knude tilføjes altid længst til venstre (en *venstretilføjelse*).

Hvis vi laver en venstretilføjelse til ovenstående, får vi følgende situation, hvor den "ulovlige" difference, 2, optræder flere steder:



For at bringe træet i orden igen kan man bruge følgende operation:



hvor A, B og C er vilkårlige højdebalancerede undertræer, og d er enten 0 eller 1.

Spørgsmål d: Hvad bliver værdierne (udtrykt i d) af diff-felterne i de to interne knuder til højre, d_1 og d_2, når denne operation er udført? $\square$

Spørgsmål e: Antag, at man laver en venstretilføjelse til et højdebalanceret træ med n knuder. Forklar, hvordan man i tid $O(\log n)$ kan sørge for, at træet bliver højdebalanceret. $\square$

Opgave 2 (25%)

Vi ser på tekststrenge, hvor bogstaverne 'a' til 'z' og cifrene '0' til '9' må indgå. Desuden ser vi på *mønstre*, der yderligere på indeholde tegnet '*', som kan stå for en vilkårlig streng (også den tomme). Et mønster `m` *matcher* en streng `s`, hvis man kan vælge strenge for hver '*' i mønsteret, så `m` (med disse valg indsat i stedet for '*') bliver lig med `s`.

Følgende PYTHON-funktion afgør om et mønster matcher en streng (de udskrevne værdier står angivet som kommentarer):

```
false, true = 0, 1

def Match(m, s, i, j):
    if i == len(m): return j == len(s)
    elif m[i] == "*":
        for k in range(j, len(s)+1):
            if Match(m, s, i+1, k): return true
        return false
    elif j == len(s): return false
    elif m[i] == s[j]: return Match(m, s, i+1, j+1)
    else: return false

print Match("hej * dig", "hej med dig", 0, 0)                 # 1
print Match("dm* er sj*vt", "dm02 er sjovt", 0, 0)            # 1
print Match("*111*222*", "111122223333444", 0, 0)             # 1
print Match("bare * jul", "gid det snart var jul", 0, 0)      # 0
print Match("*13*", "karakter", 0, 0)                         # 0
```

Spørgsmål a: Forklar, hvordan `Match` virker. □

Det viser sig, at `Match` bliver kaldt med de samme parametre gentagne gange, og at den får eksponentiel udførelsestid. Derfor vil vi lave en forbedring:

Spørgsmål b: Lav en ny version af `Match` vha. dynamisk programmering. □

Spørgsmål c: Hvad bliver tabelstørrelsen i dynamisk-programmerings-udgaven? Hvad bliver tidskompleksiteten? □

Opgave 3 (25%)

I denne opgave ser vi på venstreprofiler. Givet en ikke-tom liste af heltal, `L`, skal der konstrueres en liste, `P`, så `P[i]` er 1, hvis `L[i]` er større end eller lig med alle tal til venstre for denne position i listen `L`, og 0 ellers. Listen `P` kaldes en *venstreprofil*.

Som et eksempel, hvis `L` er listen `[2,6,4,8,5,42,42,6,17]`, så skal `P` være listen `[1,1,0,1,0,1,1,0,0]`. `P[0]` er selvfølgelig altid 1.

Følgende PYTHON-program løser problemet:

```
# Pre
P = len(L) * [None]
m = L[0]
P[0] = 1
i = 1
while i < len(L): # I
    if L[i] >= m:
        P[i] = 1
        m = L[i]
    else:
        P[i] = 0
    i = i + 1
# Post
```

Spørgsmål a: Bevis, at programmet terminerer. □

Spørgsmål b: Lad I være udsagnet:

$$i \in \mathbb{N} \ \wedge \ i \leq \text{len}(L) \ \wedge \ m = \max\{L[0], \ldots, L[i-1]\} \ \wedge$$
$$\forall j \in \{0, \ldots, i-1\}\colon \ P[j] = 1 \ \Leftrightarrow \ L[j] \geq \max\{L[0], \ldots, L[j-1]\}$$

Vis, at I er en invariant for **while**-løkken. □

Spørgsmål c: Antag, at pre- og postbetingelserne for programmet er følgende:

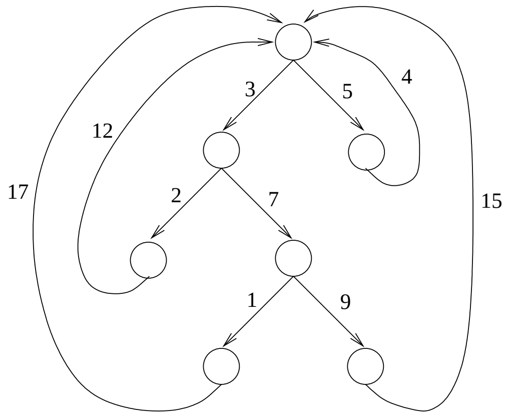
Pre: $\text{len}(L) \geq 1$

Post: $\forall j \in \{0, \ldots, \text{len}(L)-1\}\colon \ P[j] = 1 \ \Leftrightarrow \ L[j] \geq \max\{L[0], \ldots, L[j-1]\}$

Argumentér for, at programmet er korrekt. □

Opgave 4 (20%)

Nedenfor ses et eksempel på et *rodet træ*.



Mere præcist er et rodet træ en orienteret graf, der er konstrueret ved at tage udgangspunkt i et ikke-tomt binært træ, hvor alle knuder har 0 eller 2 børn. Derefter tilføjes så en kant fra hvert blad til roden. Endelig har kanterne ikke-negative vægte.

Mængden af rodede træer er altså en delmængde af alle orienterede grafer. Delmængden har en række særlige egenskaber; f.eks. er antallet af kanter mindre end $\frac{3}{2}n$, hvor n er antallet af knuder.

Vi interesserer os for følgende problem: Givet et rodet træ, en reference til roden og referencer til to knuder, a og b. Hvad er længden af den korteste vej fra a til b?

Spørgsmål a: Angiv på så simpel form som muligt den tidskompleksitet som Dijkstra's algoritme vil garantere for dette problem. Argumentér for dit svar. $\square$

Spørgsmål b: Forklar, hvordan man kan lave en algoritme, der løser problemet i tid $O(n)$. $\square$