Institut for Matematik og Datalogi
Syddansk Universitet, Odense

2. november 2003
finie02@imada.sdu.dk

DM02 – Opgaver fra ugeseddel 8

Øvelsesopgaver 21/10, 22/10 og 23/10

1. Januar 2003, opg. 3

Spørgsmål a
Udfyld tabellen med sandhedsværdierne for $U(k, s)$ givet:

$$n = 3, S = 7, C_1 = \{1, 2, 4\}, C_2 = \{2, 3, 5\}, C_3 = \{3, 4\}$$

	0	1	2	3	4	5	6	7
0	S	F	F	F	F	F	F	F
1	F	S	S	F	S	F	F	F
2	F	F	F	S	S	S	S	S
3	F	F	F	F	F	F	S	S

Husk at den tomme sum er 0.

Spørgsmål b
Forklar hvordan beregner sit resultat.

Tegn et rekursionstræ for et givent eksempel og forklar med ord hvordan algoritmen går igennem tabellen.

Herunder er angivet hvilken rækkefølge felter bliver beregnet i for eksemplet FINDU(C,3,7), med samme C mængder som i opgave a.

	0	1	2	3	4	5	6	7
0	5	4						
1			3					
2					2			
3								1

Spørgsmål c
Lav en ny version af vha. dynamisk programmering, så delresultater kun beregnes én gang.

```
FindU(C, k, s)
if U[k, s] ≠ NIL
   then return U[k, s]
if k < s
   then return FALSE
if k = 0
   then if s = 0
           then return TRUE
           else return FALSE
   else ok ← FALSE
        i ← 0
        while i < length[C[k]] and ¬ ok
           do y ← C[k, i]
              if y ≤ s
                 then ok ← FindU(C, k − 1, s − y)
              i ← i + 1
U[k, s] ← ok
return ok
```

Spørgsmål d
Hvad bliver tabelstørrelsen i dynamisk programmering udgaven?
Tabelstørrelsen er $k \cdot S$, som vist i spørgsmål a og b.

Hvad bliver worst-case tidskompleksiteten?
I værste tilfælde kan man ikke udtage et tal fra hver mængde så summen bliver S, så algoritmen får udregnet alle de mulige talkombinationer. Der er en beregning for hvert element i C_i for feltet (i, s) i tabellen. Tabellen har størrelsen $k \cdot S$ og antallet af elementer i C_i er højst S (angivet i opgavedefinitionen).
Hermen får den samlede worstcase kompleksitet:

$$W(t) \in O(k \cdot S^2)$$

2. Cormen 10.1-4
   Omskriv Enqueue og Dequeue så de tjekker for underflow og overflow af køen.

```
Enqueue(Q, x)
if head[Q] = (tail[Q] + 1) mod length[Q]
   then error "queue overflow"
Q[tail[Q]] ← x
if tail[Q] = length[Q]
   then tail[Q] ← 1
   else tail[Q] ← tail[Q] + 1
```

DEQUEUE(Q)

if $head[Q] = tail[Q]$
  then error "queue underflow"
$x \leftarrow Q[head[Q]]$
if $head[Q] = length[Q]$
  then $head[Q] \leftarrow 1$
  else $head[Q] \leftarrow head[Q] + 1$
return x

3. Cormen 10.2-2
   Implementer en stak vha. en enkelt-hægtet liste L. Operationerne PUSH og POP skal stadig køre i $O(1)$.

PUSH(L, x)

$next[x] \leftarrow head[L]$
$head[L] \leftarrow x$

POP(L)

if $head[L] =$ NIL
  then error "stack underflow"
$x \leftarrow head[L]$
$head[L] \leftarrow next[x]$
return x

4. Cormen 10.2-3
   Implementer en kø ved at bruge en enkelt-hægtet liste L. Operationerne ENQUEUE og DEQUEUE skal stadig tage $O(1)$ tid.

ENQUEUE(Q, x)

if $head[Q] =$ NIL
  then $head[Q] \leftarrow x$
  else $next[last[Q]] \leftarrow x$
$next[x] \leftarrow$ NIL
$last[Q] \leftarrow x$

DEQUEUE(Q)

if $head[L] =$ NIL
  then error "queue underflow"
$x \leftarrow head[L]$
$head[L] = next[x]$
return x

5. Cormen 10.4-2
   Skriv en $O(n)$ rekursiv procedure som, givet et binært træ med n knuder, udskriver key for hver knude i træet.

   Kaldes med PRINT-KEYS($root[T]$)

```
PRINT-KEYS(x)
if x = NIL
   then return
PRINT-KEYS(left[x])
print key[x]
PRINT-KEYS(right[x])
```

6. Cormen 22.1-1
   (udeladt med vilje)

7. Cormen 22.1-6
   Vis at man kan bestemme om en graf indeholder en universal sink i tiden $O(V)$ givet en adjacensmatrice for G.

   Definition
   En universal sink er en knude med in-degree $|V| - 1$ og out-degree 0.

   Bevis selv følgende lemmaer, og udled derefter algoritmen:

   Lemma 1
   En universal sink kan ikke have en sti til sig selv.

   Lemma 2
   En universal sink kan nås fra alle andre knuder.

   Lemma 3
   Der kan kun være én universal sink i en graf.

   Lemma 4
   Hvis der er en dead-end d i en graf, dvs. at out-degree er 0, og d ikke selv er universal sink, så er der ingen universal sink i grafen.

   Lemma 5
   Hvis vi følger en sti i grafen, uden at lave kredse (dvs. gå tilbage til en knude vi allerede har besøgt) vil vi ende i en universal sink hvis en sådan eksisterer i grafen.

   Lemma 6
   Hvis en universal sink findes i grafen, kan vi finde den ved at følge en sti på højst V skridt

   Lemma 7
   Vi kan tjekke om en bestemt knude er universal sink i tid $O(V)$

   Note:
   Til et eksamensspørgsmål er det ikke krævet at I systematisk beviser hver enkel del af idéen bag en algoritme, men I skal altid have en velformulerende beskrivelse og argumentation for jeres løsning.