EKSAMEN **Grundlæggende Algoritmer og Datastrukturer** **Fredag den 31. maj 2019, kl. 9.00–11.00**
Institut for Datalogi, Science and Technology, Aarhus Universitet
Antal sider i opgavesættet (incl. forsiden): 13
Tilladte medbragte hjælpemidler: **Ingen**

Studienummer : ____________________

Navn : ____

Vejledning og pointgivning

Dette eksamenssæt består af en mængde multiple-choice-opgaver.

Opgaverne besvares på opgaveformuleringen **som afleveres**.

For hver opgave er angivet opgavens andel af det samlede eksamenssæt.

Hvert delspørgsmål har præcist et rigtigt svar.

For hvert delspørgsmål må du vælge **max ét svar** ved at afkrydse den tilsvarende rubrik.

Et delspørgsmål bedømmes som følgende:

- Hvis du sætter kryds ved det rigtige svar, får du 1 point.
- Hvis du ikke sætter nogen krydser, får du 0 point.
- Hvis du sætter kryds ved et forkert svar, får du $-\frac{1}{k-1}$ point, hvor k er antal svarmuligheder.

For en opgave med vægt $v\,\%$ og med n delspørgsmål, hvor du opnår samlet s point, beregnes din besvarelse af opgaven som:

$$\frac{s}{n} \cdot v\,\%$$

Bemærk at det er muligt at få negative point for en opgave.

Opgave 1 (6 %)

I det følgende angiver $\log n$ 2-tals-logaritmen af n.

	Ja	Nej
$3n^2$ er $O(2n)$?	A	B
n^3 er $O(n^2 \log n)$?	A	B
n^2 er $O((\log n)^8)$?	A	B
n^3 er $O(8^{\log n})$?	A	B
$n^2 + n$ er $O(3n)$?	A	B
$\sqrt{n}$ er $O(2 \log n)$?	A	B
3^n er $O(n^3)$?	A	B
$n\sqrt{n}$ er $O(n^{2/3})$?	A	B
$2^{\log n}$ er $O((\log n)^2)$?	A	B
$n^{1/2}$ er $O(n^{1/3})$?	A	B
$\log(n^2)$ er $O(\log n)$?	A	B
$27n$ er $O(n/27)$?	A	B

Opgave 2 (4 %)

Givet et sorteret array $A[1..n]$ ($A[1] < A[2] < \cdots < A[n]$) og et element x, så ønsker vi at finde indexet ℓ således at $A[\ell] \leq x < A[\ell+1]$. Det antages at $A[1] \leq x < A[n]$. Hvilken af nedenstående algoritmer er korrekt (pilene angiver linierne der varierer i algoritmerne).

	A	B	C	D
	$\ell = 1,\ h = n+1$	$\ell = 1,\ h = n+1$	$\ell = 1,\ h = n+1$	$\ell = 1,\ h = n+1$
$\rightarrow$	**while** $\ell < h$	**while** $\ell + 1 < h$	**while** $\ell < h$	**while** $\ell + 1 < h$
	$m = \lfloor (h+\ell)/2 \rfloor$	$m = \lfloor (h+\ell)/2 \rfloor$	$m = \lfloor (h+\ell)/2 \rfloor$	$m = \lfloor (h+\ell)/2 \rfloor$
$\rightarrow$	**if** $A[m] > x$	**if** $A[m] > x$	**if** $A[m] \leq x$	**if** $A[m] \leq x$
	$\ell = m$	$\ell = m$	$\ell = m$	$\ell = m$
	else	**else**	**else**	**else**
	$h = m$	$h = m$	$h = m$	$h = m$

Opgave 3 (4 %)

Angiv worst-case tiden for HeapSort på et array med n *identiske* elementer.

$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n\sqrt{n})$	$\Theta(n^2)$
A	B	C	D	E

Opgave 4 (6 %)

Angiv for hver af nedenstående algoritmer udførselstiden som funktion af n i Θ-notation.

Algoritme Loop1(n)
  $i = 1$
  while $i \le n$
    $i = 2 * i$

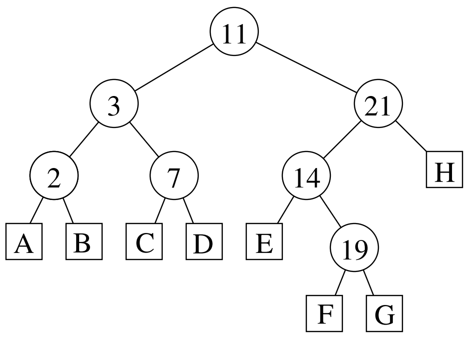
Algoritme Loop2(n)
  $i = 1$
  $s = 0$
  while $s \le n$
    $s = s + i$
    $i = i + 1$

Algoritme Loop3(n)
  $i = 1$
  while $i \le n$
    $j = i$
    while $j \le n$
      $j = 2 * j$
    $i = 2 * i$

Algoritme Loop4(n)
  $i = 1$
  while $i \le n$
    $s = 0$
    while $s \le i$
      $s = s + 1$
    $i = 2 * i$

	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n\sqrt{n})$	$\Theta(\sqrt{n})$	$\Theta((\log n)^2)$	$\Theta(n^3)$
Loop1	A	B	C	D	E	F	G	H
Loop2	A	B	C	D	E	F	G	H
Loop3	A	B	C	D	E	F	G	H
Loop4	A	B	C	D	E	F	G	H

Opgave 5 (4 %)



Angiv i hvilke blade A–H i ovenstående ubalancerede binære søgetræ elementerne 42, 10, 5, -1, og 15 skal indsættes (det antages at før hver indsættelse indeholder træet kun ovenstående syv elementer).

	A	B	C	D	E	F	G	H
`Insert(42)`	A	B	C	D	E	F	G	H
`Insert(10)`	A	B	C	D	E	F	G	H
`Insert(5)`	A	B	C	D	E	F	G	H
`Insert(-1)`	A	B	C	D	E	F	G	H
`Insert(15)`	A	B	C	D	E	F	G	H

Opgave 6 (4 %)

I følgende hashtabel af størrelse 11 er anvendt *dobbelt hashing* med hashfunktionerne $h_1(k) = 2k \bmod 11$ og $h_2(k) = 1 + (3k \bmod 10)$.

0	1	2	3	4	5	6	7	8	9	10
		12		13		3				5

Angiv positionerne de tre elementer 1, 3 og 10 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 3, 5, 12 og 13).

	0	1	2	3	4	5	6	7	8	9	10
`Insert`(1)	A	B	C	D	E	F	G	H	I	J	K
`Insert`(3)	A	B	C	D	E	F	G	H	I	J	K
`Insert`(10)	A	B	C	D	E	F	G	H	I	J	K

Opgave 7 (4 %)

Angiv den binære max-heap efter indsættelse af elementerne 3, 1, 4, 2, 6, 5, og 7 i den givne rækkefølge, startende med den tomme heap.

1	2	3	4	5	6	7	
7	6	5	2	1	3	4	A
7	4	6	1	2	3	5	B
7	5	6	1	2	3	4	C
7	6	5	4	3	2	1	D

Opgave 8 (4 %)

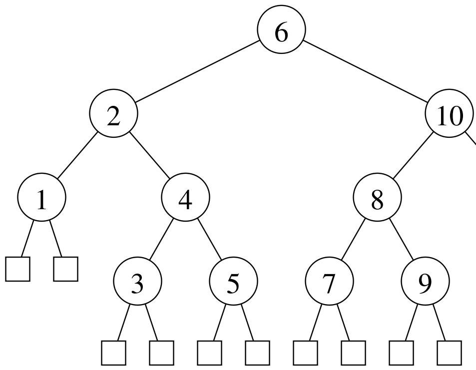
1	2	3	4	5	6	7
7	6	5	4	3	2	1

Angiv hvordan ovenstående binære max-heap ser ud efter Heap-Extract-Max.

1	2	3	4	5	6	
5	6	2	4	3	1	A
6	4	5	1	3	2	B
1	6	5	4	3	2	C
6	5	4	3	2	1	D

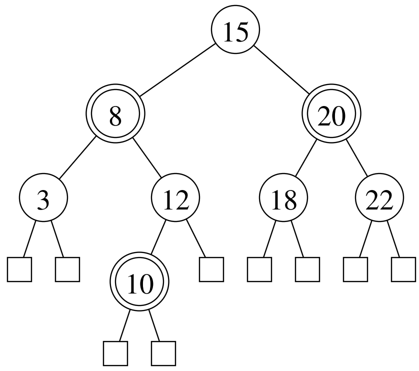
Opgave 9 (4 %)

For hver af nedenstående delmængder, angiv om nedenstående binære træ er et lovligt rød-sort træ hvis netop disse knuder farves røde



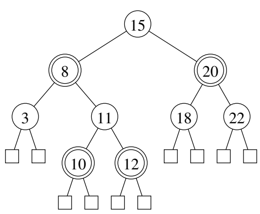
	Ja	Nej
$1, 2, 3, 5, 7, 9, 10$	A	B
$4, 6, 8$	A	B
$4, 8$	A	B
$4, 7, 9$	A	B
$3, 5, 6, 8$	A	B

Opgave 10 (4 %)

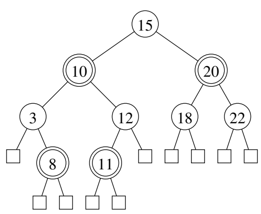


Angiv det resulterende rød-sorte træ når man indsætter 11 i ovenstående rød-sorte træ (dobbeltcirkler angiver røde knuder).

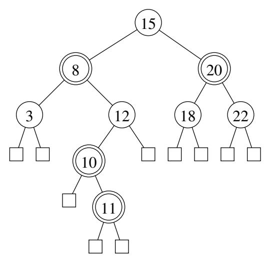
A



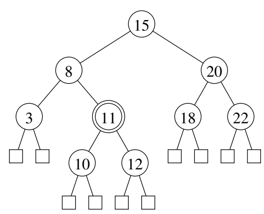
B



C



D

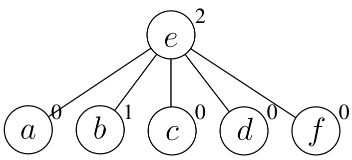


Opgave 11 (4 %)

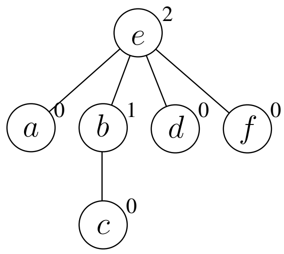
Angiv den resulterende union-find struktur efter nedenstående sekvens af operationer, når der anvendes union-by-rank og stikomprimering.

makeset(a)
makeset(b)
makeset(c)
makeset(d)
makeset(e)
makeset(f)
union(a,b)
union(a,c)
union(d,e)
union(b,e)
union(a,f)

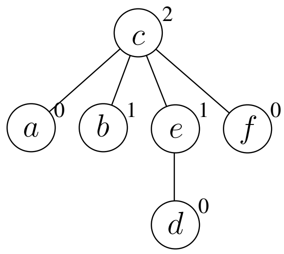
A



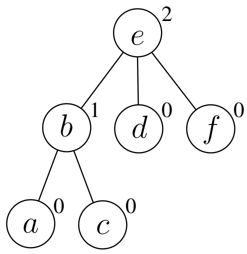
B



C



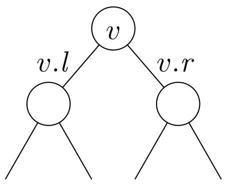
D



Opgave 12 (4 %)

Betragt et rød-sort træ hvor hver knude gemmer et par af heltal (*element*, *vægt*), og parrene er sorteret fra venstre-mod-højre efter stigende *element* værdi. For en knude v i træet lader vi $v.e$ og $v.w$ betegne parret (e, w) gemt i knuden. Desuden gemmer v værdien $v.W$ som er summen af vægtene i alle knuder i v's undertræ, og $v.\mathit{prefix}$ som er den maksimale sum af vægtene et *præfiks* af parrene i v's undertræ kan have (når parrene sorteres efter element værdi).

Angiv hvorledes $v.\mathit{prefix}$ kan beregnes når den tilsvarende information er kendt ved de to børn $v.l$ og $v.r$ (det kan antages at disse begge eksisterer).



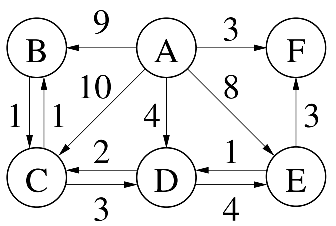
$$v.\mathit{prefix} = \begin{cases} \max(v.l.\mathit{prefix}, v.l.W + v.w, v.l.\mathit{prefix} + v.w + v.r.\mathit{prefix}) & \boxed{\text{A}} \\ \max(v.l.\mathit{prefix}, v.l.W + v.w, v.l.W + v.w + v.r.\mathit{prefix}) & \boxed{\text{B}} \\ \max(v.l.\mathit{prefix}, v.W + v.r.\mathit{prefix}) & \boxed{\text{C}} \\ \max(v.l.W, v.l.W + v.w, v.l.W + v.w + v.r.W) & \boxed{\text{D}} \end{cases}$$

Opgave 13 (4 %)

Hver af følgende rekursionsligninger har basistilfældet $T(1) = 1$. Angiv for hver ligning, hvad løsningen er.

	$\Theta(1)$	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^2 \log n)$	$\Theta(n^3)$
$T(n) = 1 + 2T(n/4)$	A	B	C	D	E	F	G	H
$T(n) = T(n/3) + 1$	A	B	C	D	E	F	G	H
$T(n) = T(n-1) + n$	A	B	C	D	E	F	G	H
$T(n) = 2 \cdot T(n/2) + n^2$	A	B	C	D	E	F	G	H

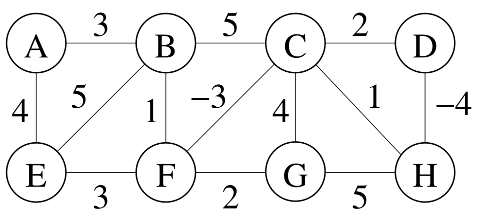
Opgave 14 (4 %)



Antag Dijkstras algoritme anvendes til at finde korteste **afstande fra A** til alle knuder i ovenstående graf. Angiv hvilken rækkefølge knuderne bliver taget ud af prioritetskøen i Dijkstra's algoritme.

A F D E B C	A F D E C B	A F D C B E	A F D C E B
A	B	C	D

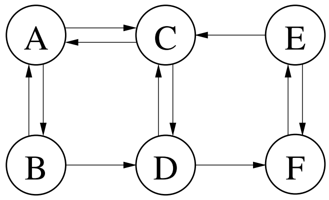
Opgave 15 (4 %)



Antag Prims algoritme anvendes til at finde et minimum udspændende træ for ovenstående graf, og algoritmen **starter i knuden A**. Angiv hvilken rækkefølge knuderne bliver inkluderet i det minimum udspændende træ (taget ud af prioritetskøen i Prims algoritme).

A B E F C H D G	A D H C F B G E	A B F C H D G E	A B F C H D E G
A	B	C	D

Opgave 16 (4 %)



Betragt et DFS-gennemløb af ovenstående graf, hvor DFS-gennemløbet **starter i knuden A**, hvor de udgående kanter til en knude besøges i alfabetisk rækkefølge.

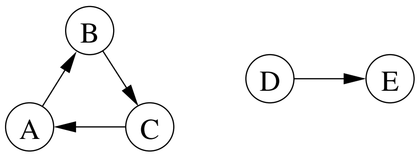
Angiv i hvilken rækkefølge knuderne får tildelt "**finishing time**".

A	B	C	D
C D E F B A	C E F D B A	A B D C F E	E F D C B A

Angiv for hver af nedenstående kanter kvilken type kanten bliver i DFS gennemløbet.

	Tree edge	Back edge	Cross edge	Forward edge
(A, C)	A	B	C	D
(E, C)	A	B	C	D
(D, F)	A	B	C	D

Opgave 17 (4 %)



Angiv for hver af nedenstående ordninger af knuderne i ovenstående graf om det er en lovlig topologisk sortering.

	Ja	Nej
A B C D E	A	B
A D B C E	A	B
D E A B C	A	B
A D E B C	A	B
A B D E C	A	B

Opgave 18 (4 %)

Givet et positive heltal x og y, så beregner nedenstående algoritme x^y.

Algoritme Power(x, y)
Inputbetingelse : Heltal $x \geq 1$ og $y \geq 1$
Outputkrav : $r = x^y$
Metode : $r \leftarrow 1$
  $\{I\}$ **while** $y \geq 1$ **do**
    if y ulige **then**
      $y \leftarrow y - 1$
      $r \leftarrow r * x$
    else
      $y \leftarrow y/2$
      $x \leftarrow x * x$

For hvert af nedenstående udsagn, angiv om de er en invariant I for algoritmen Power, hvor x_0 og y_0 angiver start værdierne for x og y.

	Ja	Nej
$r = x^y$	A	B
$r = x_0^{y_0}$	A	B
$r = x_0^{y_0 - y}$	A	B
$x_0^{y_0} = r \cdot x^y$	A	B
$x^y = r \cdot x_0^{y_0}$	A	B

Opgave 19 (4 %)

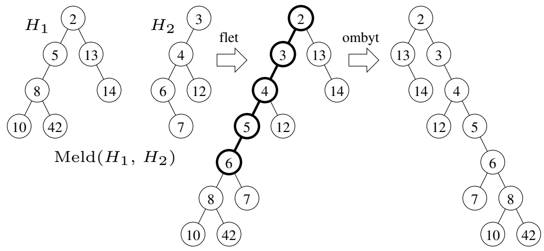
Strassen's algoritme til multipliaktion af kvadratiske $n \times n$ matricer er en del-og-kombiner algoritme. Angiv hvilken rekursionsligning der beskriver udførselstiden af Strassen's algoritme.

$T(n) \leq 7 \cdot T(n/2) + c \cdot n$	A
$T(n) \leq 7 \cdot T(n/4) + c \cdot n^2$	B
$T(n) \leq 7 \cdot T(n/2) + c \cdot n^2$	C
$T(n) \leq 7 \cdot T(n/2) + c \cdot n^3$	D

Opgave 20 (4 %)

I denne opgave betragter vi nedenstående implementation af minimum-prioritetskøer, hvor en prioritetskø er repræsenteret ved et binært træ, hvor hver knude gemmer præcist ét element og træet opfylder heap-orden, dvs. et element i en knude er altid større end eller lig med elementet i faderknuden. Træerne er ikke nødvendigvis balancerede. NULL angiver et tomt træ, H roden af et træ og Node(e, *left*, *right*) laver en ny knude med elementet e og venstre barn *left* og højre barn *right*. *Venstre-stien* i et træ består af roden, og de knuder man kan nå ved kun at gå til venstre startene i roden. Proceduren Meld fletter venstrestierne i to træer, og bytter om på venstre og højre børnene på alle de besøgte knuder (se eksempel).

proc Min(H)
  return H.e
proc DeleteMin(H)
  return Meld(H.left, H.right)
proc Insert(H, e)
  return Meld(H, Node(e, NULL, NULL))
proc Meld(H_1, H_2)
  if $H_1 = $ NULL **then return** H_2
  if $H_2 = $ NULL **then return** H_1
  if $H_1.e \leq H_2.e$ **then return** Node($H_1.e$, H_1.right, Meld(H_1.left, H_2))
  else return Node($H_2.e$, H_2.right, Meld(H_2.left, H_1))



Vi definer en knude x til at være *god* hvis dets venstre træ højst indeholder halvdelen af elementerne i x's undertræ, dvs. x.left = NULL eller $|x.\text{left}| \leq |x|/2$ hvor $|x|$ angiver antallet af knuder i undertræet rodet i x.

Hvor mange *gode knuder* kan der maksimalt være på venstre-stien i et træ med n knuder?

$O(1)$	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$
A	B	C	D

Med en passende potentialefunktion for et træ H repræsenterende en prioritetskø kan man argumentere for at alle ovenstående operationer tager amortiseret $O(\log n)$ tid, hvor n er størrelsen af den resulterende prioritetskø. Angiv for hver af nedenstående udsagn om dette er en sådan potentialefunktion Φ for et træ H.

	Ja	Nej
$\lvert H \rvert$	A	B
Længden af venstre-stien i H	A	B
Antallet af knuder på venstre-stien i H som *ikke* er gode	A	B
Antallet af knuder i H der *ikke* er gode	A	B
Antallet af knuder i H der er gode	A	B

Dynamisk programmering

De næste fire opgaver vedrører at løse *mønt opdelings* problemet ved hjælp af dynamisk programmering.

Givet en mængde af n positive heltal $C = \{c_1, c_2, \ldots, c_n\}$, som angiver forskellige møntværdier, hvor $c_1 = 1$, og et positivt heltal V, ønsker vi at finde det mindste antal mønter for at opnå værdien V. F.eks. for $C = \{1, 5, 7\}$ har vi at $V = 18$ kan beskrives som summen af de fire mønter $1+5+5+7 = 18$. Bemærk at en given møntværdi må indgå et vilkårligt antal gange.

For $V \geq 0$ lader vi $N(V)$ angive det mindste antal mønter, der skal til for at opnå værdien V, f.eks. $N(18) = 4$. $N(v)$ kan bestemmes ved følgende rekursionsformel.

$$N(v) = \begin{cases} 0 & \text{hvis } v = 0 \\ \min\{1 + N(v-c) \mid c \in C \ \wedge \ c \leq v\} & \text{ellers} \end{cases}$$

De følgende 4 opgaver består i at udfylde 4 blokke i følgende algoritmeskabelon.

Algoritme Coins(C, V)
$n = |C|$
Opret tom tabel T

for . . . ≪ **Opgave 21**: iterer over T ≫
≪ **Opgave 22**: beregn $T[v] = N(v)$ ≫
≪ **Opgave 23**: sæt *solution* til det minimale antal mønter ≫
≪ **Opgave 24**: Udskriv en løsning ≫

Opgave 21 (4 %)

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

Ja A	Ja A	Ja A	Ja A
Nej B	Nej B	Nej B	Nej B
for $i = 0$ **to** n	**for** $i = n$ **to** 0 **step** -1	**for** $i = 0$ **to** V	**for** $i = V$ **to** 0 **step** -1

Opgave 22 (4 %)

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

Ja $\boxed{\text{A}}$

Nej $\boxed{\text{B}}$

```
if i = 0 then
  T[i] = 0
else
  T[i] = i
  for j = 1 to n
    T[i] = min(T[i], 1 + T[i − c_j])
```

Ja $\boxed{\text{A}}$

Nej $\boxed{\text{B}}$

```
if i = 0 or c_i > i then
  T[i] = 0
else
  T[i] = 1 + T[i − c_i]
```

Ja $\boxed{\text{A}}$

Nej $\boxed{\text{B}}$

```
T[i] = i
for j = n to 1 step −1
  if c_j ≤ i and T[i] > 1 + T[i − c_j] then
    T[i] = 1 + T[i − c_j]
```

Opgave 23 (4 %)

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

Ja $\boxed{\text{A}}$

Nej $\boxed{\text{B}}$

```
solution = T[V]
for i = V − 1 to 0 step −1
  if T[i] < solution then
    solution = T[i]
```

Ja $\boxed{\text{A}}$

Nej $\boxed{\text{B}}$

```
solution = T[V]
```

Ja $\boxed{\text{A}}$

Nej $\boxed{\text{B}}$

```
solution = T[n]
```

Opgave 24 (4 %)

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

Ja $\boxed{\text{A}}$

Nej $\boxed{\text{B}}$

```
i = V
while i > 0
  j = T[i]
  print i − j
  i = j
```

Ja $\boxed{\text{A}}$

Nej $\boxed{\text{B}}$

```
i = V
while i > 0
  k = T[i]
  print c_k
  i = i − c_k
```

Ja $\boxed{\text{A}}$

Nej $\boxed{\text{B}}$

```
i = V
while i > 0
  k = 1
  for j = 1 to n
    if c_j ≤ i and T[i − c_j] < T[i − c_k] then
      k = j
  print c_k
  i = i − c_k
```