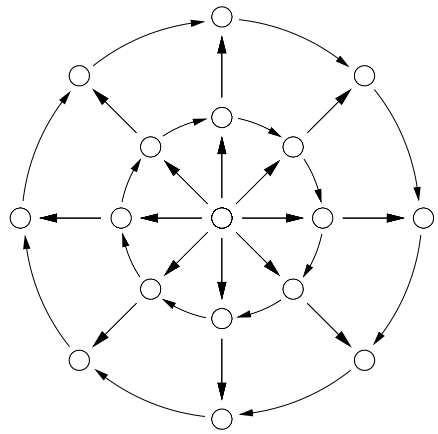
Opgave 1 (20%)

En *hjulgraf* er en orienteret graf af følgende type:



Mere præcist består en hjulgraf af en centerknude c, hvorfra der udgår k orienterede "eger", hver bestående af s kanter. Alle egers i'te knude (for $i = 2, 3, \ldots, s+1$) er desuden forbundet til en orienteret kreds. Orienteringen af alle kanter er som indikeret af figuren ovenfor. På figuren er $k = 8$ og $s = 2$.

Spørgsmål a: Angiv antallet m af kanter i en hjulgraf som funktion af antallet n af knuder. □

Vi lader nu kanterne i hjulgrafen være vægtede med positive heltal. Vi ønsker at finde længderne af de korteste veje fra centerknuden c til alle andre knuder i hjulgrafen.

Spørgsmål b: Angiv udførelsestiden (udtrykt ved n), hvis vi bruger Dijkstras algoritme. □

Spørgsmål c: Beskriv (i ord) en $O(n)$ algoritme, som finder de korteste veje fra centerknuden c til alle andre knuder i en hjulgraf. Du skal både gøre rede for algoritmens kompleksitet, og for at den finder det ønskede.

[Vink: Tag en kreds ad gangen, og start undersøgelsen af hver kreds ved en knude, som i en passende forstand er minimal.]

□

Opgave 2 (25%)

I denne opgave betragtes den sædvanlige ADT *dictionary* (jf. Goodrich og Tamassia, side 248), som ønskes udvidet med en operation

$$\textsc{count}(k)\text{: Angiv antallet af elementer i } D\text{, hvis nøgle er større end } k.$$

Spørgsmål a: Vis hvorledes en sådan udvidet dictionary kan implementeres, så operationerne findElement, insertItem, remove og count alle får logaritmiske udførelsestider.

[Vink: Udvid et rød-sort træ med passende ekstra information i knuderne.] □

En *inversion* i et array S er som bekendt et par af elementer $S[i]$, $S[k]$, hvor $i < k$ og $S[i] > S[k]$.

Spørgsmål b: Beregn antallet af inversioner, hvis array'ets indhold i rækkefølge er

$$[5, 2, 7, 1, 9, 4, 6]$$

□

Spørgsmål c: Beskriv en $O(n \log n)$ algoritme, der under brug af den udvidede dictionary fra spørgsmål **a** finder antallet af inversioner i et array S. Du skal både gøre rede for algoritmens kompleksitet, og for at den finder det ønskede. □

Opgave 3 (30%)

Denne opgave handler om at give penge tilbage med et mindst muligt antal mønter. De tilgængelige møntstørrelser kaldes tilsammen et *møntsæt*, og beskrives ved en liste $(m_1, m_2, \ldots, m_K)$ af heltal i stigende orden, hvor K er antallet af forskellige møntstørrelser. Vi antager, at $m_1 = 1$ (for at være sikker på at kunne give alle beløb tilbage), og at alle tallene er forskellige. Eksempler på møntsæt er $(1, 5, 10, 50)$ og $(1, 7, 9, 13)$.

For et givet møntsæt kan en samling mønter beskrives ved en vektor $(v_1, v_2, \ldots, v_K)$ af ikke-negative heltal, hvor v_i angiver, hvor mange mønter vi har af størrelse m_i. Det samlede antal mønter er naturligvis $\sum_{i=1}^{K} v_i$. Hvis

$$\sum_{i=1}^{K} v_i m_i = B,$$

d.v.s. hvis mønterne tilsammen repræsenterer beløbet B, kalder vi vektoren en B-vektor. En B-vektor er *optimal*, hvis ingen anden B-vektor bruger færre mønter. For eksempel er $(0, 3, 2, 0)$ og $(0, 1, 3, 0)$ begge 35-vektorer for det første møntsæt ovenfor, men kun den sidste vektor er optimal.

Givet et møntsæt og et beløb B, ønsker vi at finde en optimal B-vektor.

En oplagt algoritme af grådig type er beskrevet ved nedenstående pseudo-kode, hvor **div** og **mod** betegner henholdsvis heltalsdivision og rest ved heltalsdivision:

```
OptimalSum(B)
  for i = K down to 1 do
    v_i = B div m_i;
    B = B mod m_i;
```

Man kan vise (men det kræves ikke her), at for visse møntsæt er denne algoritme korrekt (f.eks. hvis møntsættet opfylder, at m_i går op i m_{i+1} for $i = 1, 2, \ldots, K-1$). Dette gælder dog ikke for alle møntsæt:

Spørgsmål a: Vis med et eksempel, at denne algoritme ikke er korrekt for alle møntsæt. D.v.s. find et møntsæt og et beløb, hvor algoritmen svarer forkert. □

For et givet møntsæt $(m_1, m_2, \ldots, m_K)$ lader vi $a[k, b]$ betegne det mindste *antal* mønter i nogen b-vektor som kun anvender møntstørrelserne $m_1, m_2, \ldots, m_k$. Her er k og b heltal, hvor $1 \leq k \leq K$ og $0 \leq b$.

Spørgsmål b: Gør rede for, at $a[k,b]$ kan beskrives ved følgende rekursionsformel:

$$a[k,b] = \begin{cases} b\,, & k = 1 \\ \min\limits_{0 \le i \le b \,\mathbf{div}\, m_k} \{\, i + a[k-1, b - i \cdot m_k] \,\}\,, & k > 1 \end{cases}$$

□

Spørgsmål c: Beskriv en algoritme baseret på dynamisk programmering, som for et givet møntsæt $(m_1, m_2, \ldots, m_K)$ og beløb B finder antallet af mønter i en optimal B-vektor i tid $O(KB^2)$. Der kræves et argument for, at algoritmen har den angivne kompleksitet. □

Spørgsmål d: Gør rede for, hvordan algoritmen kan udvides til ud over det mindste antal mønter også at finde en optimal B-vektor. □

Opgave 4 (25%)

Heltalskvadratroden af et tal $a \geq 1$ er som bekendt det positive heltal x, for hvilket

$$x^2 \leq a < (x+1)^2.$$

Herons formel er en klassisk metode til at beregne kvadratrødder iterativt. I denne opgave ser vi på heltalsversionen af Herons formel (i det følgende betegner vi således heltalsdivision med ":").

Algoritme: Heron
Input: a, positivt heltal
Output: x, hvor $x^2 \leq a < (x+1)^2$
Metode: $x \leftarrow a$
$\{ a < (x+1)^2 \ \wedge \ x \geq 1 \}$
while $a < x * x$ **do**
  $x \leftarrow (x + a : x) : 2$

Spørgsmål a: Angiv hvilke bevisbyrder, der skal eftervises i et gyldighedsbevis for algoritmen. □

Spørgsmål b: Eftervis bevisbyrderne fra spørgsmål **a**.
[Vink: Observér at $\frac{1}{2}(x + \frac{a}{x}) - 1 \leq (x + a : x) : 2 \leq \frac{1}{2}(x + \frac{a}{x})$.] □

Spørgsmål c: Argumentér for, at algoritmen er korrekt. □

Skriftlig Eksamen
Algoritmer og Datastrukturer (dADS)

Datalogisk Institut
Aarhus Universitet

Fredag den 4. august 2000, kl. 09–13

Opgave 1 (20%)

Lad A betegne et array af længde n, indeholdende heltal. Hvis der ikke findes to indekser i, j med $A[i] = A[j]$ (og $i \neq j$), da siges A at være *uden dubletter*.

Spørgsmål a: Antag, at du har én af følgende til rådighed:

i) En DICTIONARY.

ii) En PRIORITY QUEUE.

Forklar i begge tilfælde, hvorledes man under brug af strukturen kan afgøre, hvorvidt A er uden dubletter. □

Spørgsmål b: Brug svaret på spørgsmål **a** til at angive en algoritme med worst-case udførelsestid $O(n \log n)$, der afgør hvorvidt A er uden dubletter. □

Spørgsmål c: Som spørgsmål **b**, men algoritmen skal nu have forventet udførelsestid $O(n)$. □

Opgave 2 (25%)

En delstreng af en streng x er en *sammenhængende* række tegn fra x. Hvis en delstreng af x også er en delstreng af y, siges den at være en fælles delstreng. Hvis der om en fælles delstreng for x og y gælder, at ingen anden fælles delstreng er længere, kaldes den en *længste fælles delstreng* for x og y.

> Eksempel: længste fælles delstreng af `stegepanden` og `legepladsen` er `egep`.

En delstreng, der ender helt til højre i strengen x, kaldes et suffix af x. Givet to strenge x og y, defineres *længste fælles suffix* som det længste suffix af x, der også er et suffix af y. I eksemplet ovenfor er dette lig `en`.

I resten af denne opgave er $x = x_1 x_2 \ldots x_n$ og $y = y_1 y_2 \ldots y_m$ to givne strenge af længde henholdsvis n og m. Med $S[i, j]$ betegner vi *længden* af længste fælles suffix af strengene $x_1 x_2 \ldots x_i$ og $y_1 y_2 \ldots y_j$, for $1 \leq i \leq n$ og $1 \leq j \leq m$.

Spørgsmål a: Antag, at værdierne $S[i, j]$ er tabellagt for alle i, j, og at opslag kan foretages i konstant tid. Gør rede for, hvorledes man kan finde en længste fælles delstreng af x og y i tid $O(nm)$. □

Spørgsmål b: Angiv en rekursionsformel for $S[i, j]$. □

Spørgsmål c: Beskriv en algoritme baseret på dynamisk programmering, som i tid $O(nm)$ finder en længste fælles delstreng af x og y. Der kræves et argument for, at algoritmen har den angivne kompleksitet. □

Opgave 3 (25%)

Vi ønsker at konstruere en abstrakt datatype, DIAGRAM. Denne skal realisere *pindediagrammer*, i hvilke der til ethvert heltal $p \geq 0$ knyttes en *højde*, som er et ikke-negativt heltal. Figuren nedenfor viser et eksempel.

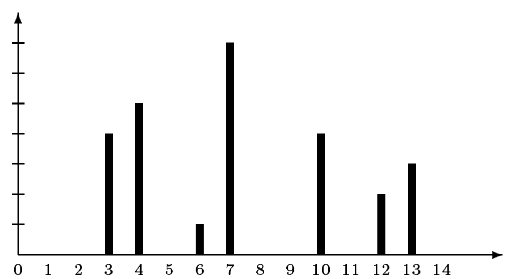


DIAGRAM skal have følgende metoder:

`Diagram()`:	Konstruktor. Opretter et pindediagram, hvor alle pinde har højde 0.
`change(int` p`, int` k`)`:	Lægger k til højden af pind p (k kan være negativ, men højden må ikke komme under 0).
`height(int` p`)`:	Returnerer højden af pind p.
`totalSum()`:	Returnerer summen af højderne af alle pinde.

I det følgende betegner n antallet af pinde, som ikke har højde nul.

Spørgsmål a: Beskriv en implementation af den abstrakte datatype DIAGRAM, så `Diagram` får udførelsestid $O(1)$, `change` og `height` får udførelsestid $O(\log n)$, og `totalSum` får udførelsestid $O(1)$. Der kræves ikke egentlig kode, men derimod en beskrivelse i ord af de væsentlige implementationsdetaljer, samt et argument for at den ønskede effektivitet opnås. □

Vi ønsker nu at tilføje yderligere en metode til den abstrakte datatype DIAGRAM:

`intervalSum(int` p_1`, int` p_2`)`:	Returnerer summen af højderne af alle pinde p, som opfylder $p_1 \leq p \leq p_2$.

Spørgsmål b: Beskriv, hvorledes implementationen fra det foregående spørgsmål kan modificeres, så `intervalSum` får udførelsestid $O(\log n)$. □

Opgave 4 (30%)

Heltalslogaritmen med basis b ($b \geq 2$) af et positivt heltal n er det ikke-negative heltal x, for hvilket

$$b^x \leq n < b^{x+1}.$$

Følgende algoritme påstås at beregne heltalslogaritmen:

Algoritme: Heltalslogaritme

Input: n, positivt heltal

Output: x, heltalslogaritmen af n

Metode: $x \leftarrow 1$; $B \leftarrow b$;
$\{\ (B = b^x)\ \wedge\ (b^{x-1} \leq n)\ \}$
while $B \leq n$ **do**
  $x \leftarrow x + 1$;
  $B \leftarrow B \cdot b$;
$x \leftarrow x - 1$;

Spørgsmål a: Angiv hvilke bevisbyrder, der skal eftervises i et gyldighedsbevis for algoritmen. □

Spørgsmål b: Eftervis bevisbyrderne fra spørgsmål **a**. □

Spørgsmål c: Argumentér for, at algoritmen er korrekt. □

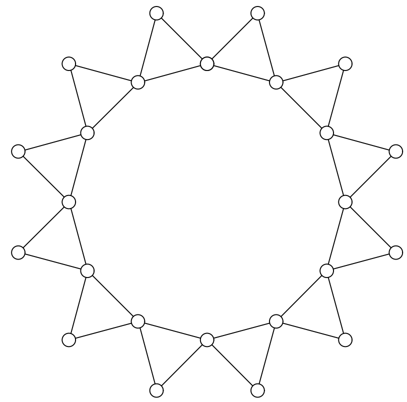
Skriftlig Eksamen
Algoritmer og Datastrukturer (dADS)

Datalogisk Institut
Aarhus Universitet

Mandag den 28. maj 2001, kl. 09–13

Opgave 1 (30%)

En *solsikke-graf* med omkreds k er en uorienteret graf bestående af k trekanter sat sammen i en kreds. For $k = 12$ ser grafen sådan ud:



Spørgsmål a: Angiv sammenhængen mellem antallet m af kanter og antallet n af knuder i en solsikke-graf. $\square$

Vi lader nu kanterne i solsikke-grafen være vægtede med ikke-negative tal. Vi ønsker at finde dels korteste veje fra en given spids i grafen til alle andre knuder, dels et minimum udspændende træ for grafen.

Spørgsmål b: Argumentér for, at både Dijkstras algoritme for korteste veje og Kruskals algoritme for minimum udspændende træ bruger $\Omega(n \log n)$ tid på en solsikke-graf. $\square$

Spørgsmål c: Beskriv en $O(n)$ algoritme, som givet en reference til en spids i grafen (d.v.s. en knude af grad to) finder længden af de korteste veje til alle andre knuder. Beskriv ligeledes en $O(n)$ algoritme, som finder et minimum udspændende træ for en vægtet solsikke-graf. Du skal argumentere for algoritmernes kompleksitet, og for at de finder det ønskede. Antag at grafen som sædvanligt er givet ved en *adjacency list* representation. $\square$

Man kan uden bevis bruge, at der gælder følgende sætning om minimum udspændende træer:

> *Lad $G = (V, E)$ være en vægtet uorienteret graf, og lad $K \subseteq E$ være en kantmængde, som indeholder et minimum udspændende træ for G. Hvis K indeholder en cykel C og hvis e er en kant i C af størst vægt, da vil $K - \{e\}$ også indeholde et minimum udspændende træ for G.*

Opgave 2 (25%)

I forbindelse med fremstilling af tabeller for matematiske standardfunktioner (logaritmer, trigonometriske funktioner m.v.) har man traditionelt benyttet såkaldte *itererede differenser*. Ideen i denne metode er, at man først approksimerer den funktion, der skal tabellægges, med et polynomium, og dernæst udnytter, at man kan udregne værdien af et polynomium på *ækvidistante* punkter v.hj.a. en iterativ metode, der kun anvender additioner.

Følgende algoritme påstås at beregne værdierne af andengradspolynomiet

$$p(x) = ax^2 + bx + c$$

i punkterne $0, 1, 2, \ldots, n$.

Algoritme: ID(a, b, c, n)

Input: a, b, c: koefficienter i polynomiet $p(x)$
$n \geq 0$: tabelstørrelsen

Output: T: $T[i] = ai^2 + bi + c$ for $0 \leq i \leq n$

Metode: $T[0] \leftarrow c$; $t \leftarrow a + b$; $k \leftarrow 0$
$\{I\}$
while $k < n$ **do**
  $T[k+1] \leftarrow T[k] + t$;
  $t \leftarrow t + a + a$;
  $k \leftarrow k + 1$

Her er I udsagnet

$$(T[i] = ai^2 + bi + c \text{ for } 0 \leq i \leq k) \ \wedge \ (k \leq n) \ \wedge \ (t = a(2k+1) + b).$$

Spørgsmål a: Angiv hvilke bevisbyrder, der skal eftervises i et gyldighedsbevis for algoritmen. □

Spørgsmål b: Eftervis bevisbyrderne fra spørgsmål **a**. □

Spørgsmål c: Argumentér for, at algoritmen er korrekt. □

Opgave 3 (20%)

Lad $x = x_1x_2\ldots x_n$, $y = y_1y_2\ldots y_m$ og $z = z_1z_2\ldots z_{n+m}$ være tre strenge af længde henholdsvis n, m og $n+m$. Vi kalder z et *flet* af x og y, hvis x og y findes som to disjunkte delsekvenser i z, og disse tilsammen udgør hele z.

> Eksempler: `gulerod` er et flet af `uro` og `gled`, og `dalgatorastritukturmerer` er et flet af `algoritmer` og `datastrukturer`.

For $0 \leq i \leq n$ og $0 \leq j \leq m$ lader vi $F[i,j]$ være en boolsk værdi, der angiver, hvorvidt strengen $z_1z_2\ldots z_{i+j}$ er et flet af strengene $x_1x_2\ldots x_i$ og $y_1y_2\ldots y_j$. Her defineres $x_1x_2\ldots x_i$ som den tomme streng, når $i = 0$ (og tilsvarende for y og z).

$F[i,j]$ kan beskrives ved følgende rekursionsformel:

$$F[i,j] = \begin{cases} X_{ij} \ \vee \ Y_{ij}\,, & i,j \geq 1 \\ X_{ij}\,, & i \geq 1, j = 0 \\ Y_{ij}\,, & i = 0, j \geq 1 \\ \texttt{Sand}\,, & i,j = 0, \end{cases}$$

hvor X_{ij} og Y_{ij} er udsagnene

$$\begin{aligned} X_{ij} &= (z_{i+j} = x_i \ \wedge \ F[i-1,j]), \\ Y_{ij} &= (z_{i+j} = y_j \ \wedge \ F[i,j-1]). \end{aligned}$$

Spørgsmål a: Opskriv tabellen for F, når x er `uro`, y er `gled` og z er `gulerod`. □

Spørgsmål b: Beskriv i form af pseudo-kode en algoritme baseret på dynamisk programmering, som i tid $O(nm)$ afgør, hvorvidt z er et flet af x og y. Der kræves et argument for, at algoritmen har den angivne kompleksitet. □

Spørgsmål c: Gør rede for, hvordan algoritmen kan udvides til i tilfælde af et positivt svar også at returnere indekserne for en delsekvens af z, som er lig x. □

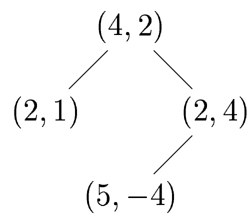
Opgave 4 (25%)

I denne opgave betragtes søgetræer, der indeholder heltal. Repræsentationen af tallene er imidlertid usædvanlig, idet hver knude indeholder et par,

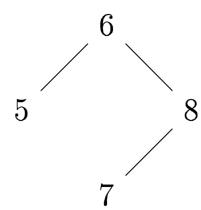
$$(basis, addend),$$

hvor *addend* skal opfattes som et tal, der skal lægges til samtlige baser i det undertræ, hvori knuden er rod.

Følgende træ

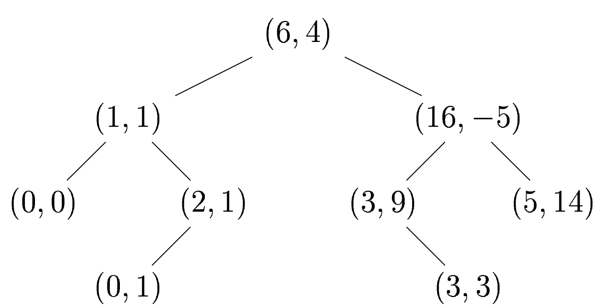


repræsenterer således det normale søgetræ



idet en knudes resulterende værdi opnås som summen af dens basis samt alle addender på vejen fra roden til knuden.

Spørgsmål a: Angiv det normale søgetræ, der repræsenteres af træet



□

Spørgsmål b: Angiv, hvordan følgende metoder kan realiseres:

$$\begin{array}{ll} \text{FIND}(k) & \text{i tid } O(h) \\ \text{INSERT}(k) & \text{i tid } O(h) \\ \text{PLUS}(d) & \text{i tid } O(1) \\ \text{SHIFT}(k,d) & \text{i tid } O(h) \end{array}$$

hvor h er træets højde. Her er PLUS(d) en operation, der lægger tallet d til alle værdier i træet, og SHIFT(k, d) er en operation, der lægger tallet $d \geq 0$ til alle værdier i træet, der er større end eller lig k. $\square$

Spørgsmål c: Forklar, hvordan man kan sikre at h er $O(\log n)$, ved at angive hvordan de repræsenterede værdier kan bevares under rebalanceringsoperationer. Her er n antallet af elementer i træet. $\square$

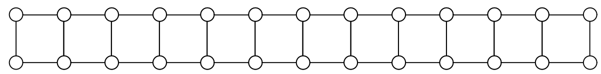
Skriftlig Eksamen
Algoritmer og Datastrukturer (dADS)

Datalogisk Institut
Aarhus Universitet

Fredag den 3. august 2001, kl. 09–13

Opgave 1 (30%)

En *jernbaneskinne* af længde k er en uorienteret graf bestående af k sammenhængende firkanter. For $k = 12$ ser grafen sådan ud:



Spørgsmål a: Angiv sammenhængen mellem antallet m af kanter og antallet n af knuder i en jernbaneskinne. □

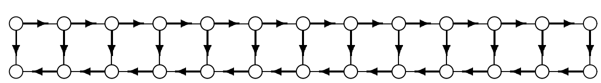
Vi lader nu kanterne i jernbaneskinnen være vægtede med ikke-negative tal. Vi ønsker at finde et minimum udspændende træ for grafen.

Spørgsmål b: Beskriv en $O(n)$ algoritme, som finder et minimum udspændende træ for en vægtet jernbaneskinne. Du skal argumentere for algoritmens kompleksitet, og for, at den finder det ønskede. Antag at grafen som sædvanligt er givet ved en *adjacency list* representation. □

Man kan uden bevis bruge, at der gælder følgende sætning om minimum udspændende træer:

> *Lad $G = (V, E)$ være en vægtet uorienteret graf, og lad $K \subseteq E$ være en kantmængde, som indeholder et minimum udspændende træ for G. Hvis K indeholder en cykel C og hvis e er en kant i C af størst vægt, da vil $K - \{e\}$ også indeholde et minimum udspændende træ for G.*

Vi ser nu på vægtede jernbaneskinner, hvor kanterne er *orienterede* som angivet på eksemplet herunder. Vi orienterer med andre ord alle vandrette kanter i øverste række mod højre, orienterer alle vandrette kanter i nederste række mod venstre og orienterer alle lodrette kanter nedad.



Spørgsmål c: Beskriv en $O(n)$ algoritme, som givet en reference til knuden i øverste venstre hjørne i en orienteret jernbaneskinne finder længden af de korteste veje til alle andre knuder. Du skal argumentere for algoritmens kompleksitet, og for, at den finder det ønskede. □

Spørgsmål d: Argumentér for, at Dijkstras algoritme for korteste veje bruger $\Omega(n \log n)$ tid på en orienteret jernbaneskinne. $\square$

Opgave 2 (25%)

I en række algoritmiske sammenhænge er der brug for at kunne beregne den såkaldte *prefix-sum* af et array A.

Prefix-summen er et andet array P (af samme længde som A), hvor elementet $P[i]$ indeholder summen af elementerne $A[0], A[1], \ldots, A[i]$. Formelt:

$$P[i] = \sum_{j=0}^{i} A[j]$$

for $0 \leq i \leq n$.

Spørgsmål a: Angiv prefix-summen af array'et

$$A = 5, 2, 6, 1, 4, 3.$$

$\square$

Betragt følgende algoritme:

Algoritme: PrefixSum(A, n)

Input: A: Array af længde $n+1$, hvor $n \geq 0$
Output: P: Prefix-summen af A
Metode: S^{Init}
$\{I\}$
while $k < n$ **do**
    S^{Body}

hvor I er invarianten

$$I: \; \left(P[i] = \sum_{j=0}^{i} A[j] \text{ for } 0 \leq i \leq k\right) \;\wedge\; (k \leq n).$$

Spørgsmål b: Angiv, udtrykt ved S^{Init}, S^{Body} og I, hvilke bevisbyrder, der skal eftervises i et gyldighedsbevis for algoritmen. $\square$

Spørgsmål c: Angiv konkret kode for S^{Init} og S^{Body}, således at bevisbyrderne kan eftervises. Gennemfør beviserne. $\square$

Spørgsmål d: Argumentér for, at den således konstruerede algoritme er korrekt. $\square$

Opgave 3 (25%)

I denne opgave betragtes en model for omkostningen ved at transformere to strenge til at matche hinanden. For simpelheds skyld tillades kun én operation, nemlig indsættelse af tegnet '`?`', som matcher ethvert andet tegn.

Givet to strenge $X = x_1 x_2 \ldots x_n$ og $Y = y_1 y_2 \ldots y_m$ skal man indsætte '`?`' i X og Y, således at de to resulterende strenge matcher hinanden. At to strenge matcher hinanden betyder, at de er lige lange og at der i hver position står et par af tegn der matcher hinanden. To tegn siges at matche hinanden, hvis de enten er ens, eller mindst ét af dem er '`?`'.

Omkostningen ved transformationen er det samlede antal af indsatte '`?`' i X og Y. Vi ønsker finde den mindste omkostning ved at transformere X og Y til to matchende strenge.

Betragt som eksempel strengene $X =$ `hund` og $Y =$ `høne`. Ved indsættelse af i alt fire '`?`' kan de bringes til at matche hinanden på følgende måde:

```
h?un?d
hø?ne?
```

Spørgsmål a: Argumenter for, at fire er den minimale omkostning i dette eksempel. $\square$

Spørgsmål b: Argumentér for, at der generelt gælder, at den minimale omkostning ved at transformere to strenge X og Y til at matche hinanden højst kan blive $|X| + |Y|$. Her angiver $|Z|$ antallet af tegn i strengen Z. $\square$

Man kan ved hjælp af dynamisk programmering finde den mindste omkostning ved at transformere X og Y til to matchende strenge.

Spørgsmål c: Opskriv en rekursionsformel for det minimale antal '`?`', der skal indsættes i X og Y for at de matcher hinanden, og konstruér på basis heraf en algoritme, som ved hjælp af dynamisk programmering finder dette antal. Du skal argumentere for såvel korrekthed som udførelsestid for algoritmen. □

Opgave 4 (20%)

For et talsæt $\{x_1, x_2, \ldots, x_k\}$ defineres *middelværdien* M som

$$M = \frac{1}{k} \sum_{i=1}^{k} x_i.$$

I denne opgave betragtes den sædvanlige ADT *dictionary* (jf. Goodrich og Tamassia, side 248), som ønskes udvidet med følgende operation:

> $\textsc{mean}(a, b)$: Angiv middelværdien for talsættet bestående af de nøgler x, der opfylder $a \leq x \leq b$.

Spørgsmål a: Angiv hvad operationen $\textsc{mean}(7, 27)$ returnerer, hvis nøglerne i dictionary'en er

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37$$

□

Spørgsmål b: Vis hvorledes en sådan udvidet dictionary kan implementeres, så operationerne FINDELEMENT, INSERTITEM, REMOVE og MEAN alle får logaritmiske udførelsestider. □