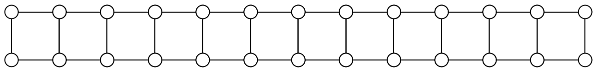
Skriftlig Eksamen
Algoritmer og Datastrukturer (dADS)

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. $\square$

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. $\square$

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.*

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 lodrette 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. $\square$

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

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

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. □

Spørgsmål d: Argumentér for, at den således konstruerede algoritme er korrekt. □

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. □

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. □

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:

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

Spørgsmål a: Angiv hvad operationen 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. □