DATALOGISK INSTITUT, AARHUS UNIVERSITET

Det Naturvidenskabelige Fakultet
EKSAMEN
Grundkurser i Datalogi
Algoritmer og Datastrukturer (gammel ordning)
Antal sider i opgavesættet (incl. forsiden): 6 (seks)
Eksamensdag: Onsdag den 11. august 2004, kl. 9.00-13.00
Eksamenslokale: Trøjborg-komplekset, Niels Juelsgade 84, lok. 139, 8200 Århus N
Tilladte medbragte hjælpemidler: Alle sædvanlige hjælpemidler (lærebøger og notater)
Materiale der udleveres til eksaminanden:

OPGAVETEKSTEN
BEGYNDER
PÅ NÆSTE SIDE

—oOo—

Opgave 1 (25%)

I denne opgave betragtes nedenstående algoritme, der givet et input array indeholdende 0-1 værdier, beregner længden af det længste delarray hvor alle indgange indeholder værdien 1.

Algoritme Ones(A)
Inputbetingelse : Array A af længde $n \geq 0$ indeholdende 0-1 værdier
Outputkrav : $m = \text{MaxOnes}(A)$
Metode : $i \leftarrow 0$;
$m \leftarrow 0$;
$\ell \leftarrow 0$;
$\{I\}$ **while** $i < n$ **do**
  if $A[i] = 1$ **then**
    $\ell \leftarrow \ell + 1$;
    $m \leftarrow \max\{\ell, m\}$
  else
    $\ell \leftarrow 0$
  $i \leftarrow i + 1$

hvor I er invarianten

$$\ell = \text{MaxSuffixOnes}(A[0..i]) \ \wedge \ m = \text{MaxOnes}(A[0..i]) \ \wedge \ 0 \leq i \leq n$$

Her betegner $A[0..i] = A[0]A[1]\cdots A[i-1]$, MaxOnes($X$) længden af det længste delarray af X kun indeholdende værdien 1, og MaxSuffixOnes(X) længden af det længste suffix af X kun indeholdende værdien 1.

For nedenstående eksempel er MaxOnes(A) = MaxOnes($A[8..14]$) = 6 og f.eks. er MaxSuffixOnes($A[0..17]$) = MaxOnes($A[15..17]$) = 2.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	1	0	1	1	1	0	0	0	1	1	1	1	1	1	0	1	1	1	1	0	1

Spørgsmål a: Hvilke værdier får ℓ tildelt i linien "$\ell \leftarrow \ell + 1$;" i løbet af algoritmen, og hvor mange gange tildeles ℓ de forskellige værdier på ovenstående eksempel? □

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

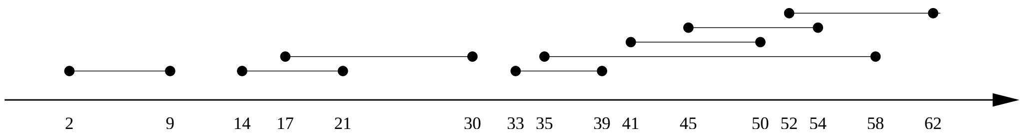
Spørgsmål c: Eftervis bevisbyrderne fra spørgsmål **b**, og argumenter for at algoritmen er korrekt. □

Opgave 2 (25%)

I denne opgave betragtes en datastruktur til at opbevare en mængde af intervaller, og som understøtter forespørgsler om et punkt q er indeholdt i mindst et interval.

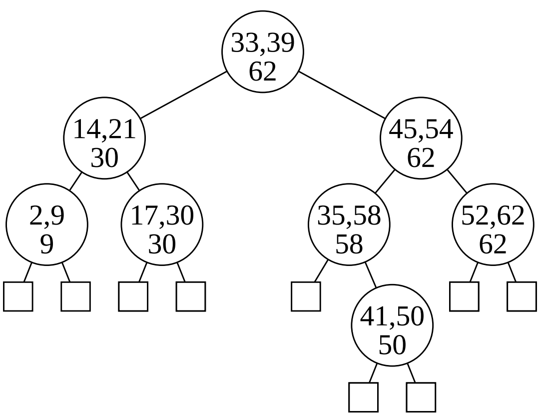
For intervallerne

$$[2,9]\,,\ [14,21]\,,\ [17,30]\,,\ [33,39]\,,\ [35,58]\,,\ [41,50]\,,\ [45,54]\,,\ [52,62]$$



gælder at f.eks. $q = 10$ ikke er indeholdt i et interval, hvorimod $q = 53$ er indeholdt i mindst et interval (faktisk tre intervaller, men dette er uvæsentligt for denne opgave).

Datastrukturen for at opbevare intervallerne er et (balanceret) søgetræ, hvor hver indre knude opbevarer et interval og hvor intervallerne er gemt sorteret efter intervallernes *venstre endepunkt*. Hver knude opbevarer desuden en værdi $r_{\max}$, som er det maksimale højre endepunkt der er opbevaret i knudens undertræ. Nedenstående viser datastrukturen for ovenstående intervaller, hvor der for hver knude øverst er angivet knudens interval og nederst er angivet knudens $r_{\max}$ værdi.



Spørgsmål a: Tegn en tilsvarende datastruktur for intervallerne

$$[21,23]\,,\ [1,6]\,,\ [18,20]\,,\ [11,13]\,,\ [17,19]\,,\ [3,8]\,,\ [5,10]\,,\ [22,24]$$ $\square$

Spørgsmål b: Beskriv hvordan et nyt interval $[l, r]$ kan indsættes i ovenstående datastruktur, hvor det antages at træet ikke kræves balanceret efter indsættelsen. $\square$

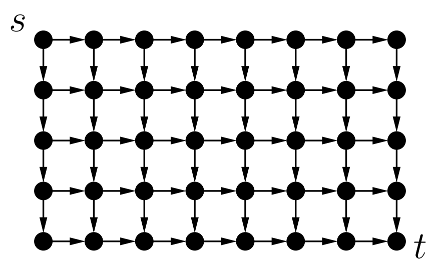
Spørgsmål c: Beskriv hvordan man under indsættelser af nye intervaller i tid $O(\log n)$ kan holde ovenstående datastruktur balanceret, d.v.s at træets højde er $O(\log n)$. $\square$

Spørgsmål d: Beskriv en rekursiv procedure $\textsc{Covered}(q)$, der afgør om et punkt q er indeholdt i mindst et interval. Procedurens udførelsestid skal være $O(h)$, hvor h er træets højde. $\square$

(Opgavesættet fortsætter)

Opgave 3 (25%)

I denne opgave betragtes (r, k) gitter-grafer, som er orienterede grafer hvor knuderne er arrangeret i et gitter med r rækker og k søjler. Hver knude v har en kant til knuden umiddelbart nedenunder v i v's søjle (såfremt v ikke er i den nederste række) og en kant til knuden umiddelbart til højre for v (såfremt v ikke er i søjlen længst til højre). Den øverste venstre knude betegnes s og knuden nederst til højre betegnes t. Nedenstående er en $(5, 8)$ gitter-graf.



I det følgende antages alle kanter at have en ikke negativ vægt.

Spørgsmål a: Angiv antal kanter m og antal knuder n i en (r, k) gitter-graf som funktion af r og k. Angiv udførelsestiden af Dijkstra's algoritme til at finde de korteste afstande fra s til t. $\square$

Spørgsmål b: Angiv en algoritme med udførelsestid $O(n)$, der beregner den korteste afstand fra s til t i en (r, k) gitter-graf. $\square$

Spørgsmål c: Angiv en algoritme med udførelsestid $O(n)$, der beregner længden af den længste sti fra s til t i en (r, k) gitter-graf. $\square$

Antag nu at kanterne også kan have en negativ vægt.

Spørgsmål d: Angiv en algoritme der beregner længden af den længste sti i en (r, k) gitter-graf. Angiv algoritmes udførelsestid. Bemærk at stien ikke behøver at gå fra s til t. $\square$

Opgave 4 (25%)

I denne opgave betragtes to strenge

$$S = s_1 s_2 \cdots s_n$$

$$T = t_1 t_2 \cdots t_m$$

af længde henholdsvis n og m. Vi ønsker at finde den fælles delsekvens for S og T der opnår den *maksimale blok-score*, som er defineret nedenfor.

Antag at en fælles delsekvens for S og T kan deles op i p blokke $B_1, \ldots, B_p$, hvor B_i er en sammenhængende delsekvens i både S og T. Vi definerer blok-scoren af opdelingen som

$$|B_1| + |B_2| + \cdots + |B_p| - p$$

Den *maksimale blok-score* er blok-scoren af den fælles delsekvens for S og T der har den maksimale blok-score.

For eksempel har nedenstående to strenge en maksimal blokscore på $2+4+4-3=7$. Kasserne angiver opdelingen i blokke af den fælles delsekvens for S og T.

$$S = \boxed{ab}\,cc\,\boxed{baca}\,ba\,\boxed{abad}$$

$$T = \boxed{ab}\,aba\,\boxed{baca}\boxed{abad}\,ad$$

Vi definerer nu:

$$B(i,j) = \text{den maksimale blok-score af } s_1 s_2 \cdots s_i \text{ og } t_1 t_2 \cdots t_j.$$

Spørgsmål a: Lad $S = bbcba$ og $T = bccbba$. Udfyld følgende tabel for $B(i,j)$ for $0 \le i \le 5$ og $0 \le j \le 6$.

$i \backslash j$	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							

□

Det påstås at $B(i,j)$ opfylder følgende rekursionsformel:

$$B(i,j) = \begin{cases} 0 & \text{hvis} \quad i = 0 \ \vee \ j = 0 \\ \max\{B(i,j-1), B(i-1,j)\} & \text{hvis} \quad s_i \neq t_j \\ \max\{B(i,j-1), B(i-1,j), B(i-k,j-k)+k-1\} & \text{hvis} \quad s_i = t_j \ \wedge \\ \qquad k \geq 1 \text{ er maksimal så: } s_i = t_j \ \wedge \ s_{i-1} = t_{j-1} \ \wedge \cdots \wedge \ s_{i-k+1} = t_{j-k+1} & \end{cases}$$

Spørgsmål b: Beskriv en algoritme baseret på dynamisk programmering, der givet to strenge S og T af længde henholdsvis n og m, beregner $B(n,m)$, d.v.s. den maksimale blok-score for strengene S og T. Angiv algoritmens udførelsestid. □

Spørgsmål c: Beskriv en udvidelse af algoritmen fra spørgsmål **b** til også at rapportere en fælles delsekvens af S og T der har maksimal blok-score. Angiv algoritmens udførelsestid. □