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 $S[i,j]$. $\square$

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. $\square$

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.

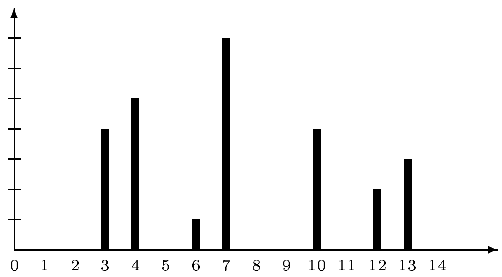

DIAGRAM skal have følgende metoder:

`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. $\square$

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. □