

Vejledende besvarelser for IADS eksamenssæt E2001

IT-højskolen i København

9. januar 2002

Opgave 1

Denne opgave handler om O -bestemmelse og sortering fra kapitlerne 2 og 8 i CLRS, eller tilsvarende kapitlerne 1 og 9 i CLR.

Betragt koden A :

```
1 for  $i \leftarrow 1$  to  $n$ 
2   do  $A[i] \leftarrow i$ 
3      $B[i] \leftarrow 1$ 
4 MERGE-SORT( $A, 1, n$ )
5 COUNTING-SORT( $A, B, n$ )
6 for  $i \leftarrow 1$  to  $n$ 
7   do COUNTING-SORT( $A, B, n$ )
```

a) Angiv den samlede tid for udførelse af linierne 1-5 for koden A .

Svar a) $O(n \log n)$, tiden majoriseres af kaldet til MERGE-SORT.

b) Angiv den samlede tid for udførelsen af linierne 6-7 for koden A .

Svar b) $O(n^2)$, n iterationer, der hver tager tid $O(n)$.

Betragt proceduren $\text{ADD1}(A, m)$

```
1 if  $m \geq 2$ 
2   then for  $i \leftarrow 1$  to  $m$ 
3     do  $A[i] \leftarrow A[i] + 1$ 
4      $\text{ADD1}(A, \lfloor m/2 \rfloor)$ 
```

Betragt koden B :

```
1 for  $i \leftarrow 1$  to  $n$ 
2   do  $A[i] \leftarrow 0$ 
3  $\text{ADD1}(A, n)$ 
```

c) Angiv den samlede tid for udførelsen af ADD1 i linie 3 for koden B .

Svar c) $1 + 2 + 4 + \dots + n/2 + n = O(n)$

d) Angiv i O -notation den største værdi i tabel A efter udførelsen af ADD1 i linie 3 for koden B .

Svar d) $O(\log n)$, indgang 1 i A inkrementeres med 1 for hvert rekursivt kald, der højst er $\log n$ af, da m halveres pr.kald.

Opgave 2

Denne opgave handler om grafer og mindst udspændende træ (“minimum spanning tree”) med terminologi som defineret i kapitel 22 og 23 i CLRS og tilsvarende kapitel 23 og 24 i CLR. Betragt følgende uorienterede graf

$$H = (\{1, 2, 3, 4, 5\}, \{(1, 2), (2, 5), (4, 5), (2, 3), (1, 5), (3, 5), (3, 4)\}).$$

a) Tegn grafen H .

Svar a) En tegning af en graf med fem knuder, der korresponderer med H .

b) Angiv adjacency-list repræsentation for H i stil med figur 22.1 side 528 i CLRS eller tilsvarende figur 23.1 side 466 i CLR.

Svar b)

1		→	2	→	5				
2		→	1	→	3	→	5		
3		→	2	→	4	→	5		
4		→	3	→	5				
5		→	1	→	2	→	3	→	4

c) Angiv kanterne, der krydser snittet (“cut”) $(\{1, 2\}, \{3, 4, 5\})$ i H .

Svar c) $(1, 5), (2, 3), (2, 5)$

Lad $G = (V, E)$ være en sammenhængende, uorienteret, vægtet graf med n knuder og m kanter. Lad knudemængden V være identificeret med heltallene $\{1, 2, \dots, n\}$ som sædvanligt.

d) Beskriv en effektiv algoritme, der givet G i adjacency-list repræsentation udfylder en tabel A med n indgange, hvor hver indgang i indeholder et j , således at kanten (i, j) er med i et mindst udspændende træ for G . D.v.s. at der for alle knuder $i \in V$ gælder at $(i, A[i])$ er en kant i et mindst udspændende træ for G . Angiv tidskompleksiteten af din løsning.

Svar d) $A[i]$ beregnes ved at løbe adjacency-listen igennem for knude i og vælge nabo j , hvor $w(i, j)$ er mindst mulig. Tiden for dette er $O(|Adj[i]|)$ for indgang i , dvs. totalt for alle knuder V , $O(|V| + \sum_i |Adj[i]|) = O(|V| + |E|)$.

e) Beskriv en effektiv algoritme, der givet G i adjacency-list repræsentation, samt en kant $e \in E$, afgør om e er med i et mindst udspændende træ for G . Angiv tidskompleksiteten af din løsning.

Svar e) Kanten e er med i et mindst udspændende træ, hvis og kun hvis forbinder to forskellige sammenhængskomponenter i grafen $G' = (V, E')$, hvor E' defineres til at være kanter fra E , hvis kantvægt er mindre end kantvægten for e . Beviset for dette er som følger. Antag e forbinder to forskellige sammenhængskomponenter i G' . Vi skal nu vise at e kan være med i et mindst udspændende træ for G . Lad $(S, V - S)$ være et snit, der respekterer sammenhængskomponenterne i G' (ingen kan i E' krydser snittet), men samtidigt separerer de to komponenter, der forbindes af e . Der gælder da, at e er en *safe* kant i G , da den er lettest i nævnte snit idet kun kanter i $E - E'$ krydser snittet. D.v.s. jvf. Thm. 23.1 i CLRS er e med i et mindst udspændende træ for G .

Omvendt, hvis $e = (u, v)$ ikke forbinder to sammenhængskomponenter, så er u og v allerede forbundet med kanter, der alle har vægt mindre en vægten for e , d.v.s. e har større vægt en de øvrige kanter på en cykel i G . Dvs. ifølge opg. 23.1-5 CLRS medfører det, at der er et mindst udspændende træ T uden kanten e . Det sidste medfører at e ikke kan være med i et MST, da ethvert udspændende træ T' , der medtager kanten $e = (u, v)$ har kantvægtsum skarpt større end træet $T'' = T(u) \cup \{e'\} \cup T(v)$, hvor $T(u)$ og $T(v)$ er subtræerne, der opstår ved at slette kant (u, v) , og $e' \neq e$ er en kant på ovennævnte cykel, der forbinder træerne $T(u)$ og $T(v)$ igen. Altså har T'' kantvægtsum skarpt mindre end T' , der derfor ikke kan være mindst udspændende træ. Altså er e ikke med i noget MST. QED

For at undersøge om e forbinder to sammenhængskomponenter i G' , beregnes først adjacency-list repræsentation for G' , ved at der i et enkelt gennemløb af repræsentation af G fjernes kanter med vægt $\geq w(e)$. Herefter undersøges for $e = (u, v)$ ved hjælp af f.eks. et dybde-først gennemløb i G' , om u er forbundet med v . Totalt tager dette lineær tid i inputtets størrelse, d.v.s $O(|V| + |E|)$.

Opgave 3

Denne opgave handler om datastrukturen for disjunkte mængder som defineret i kapitel 21 i CLRS eller kapitel 22 i CLR, samt i binære søgetræer.

Betragt følgende kode C :

```
1 for  $i \leftarrow 1$  to 5
2   do MAKE-SET( $i$ )
3 UNION(1, 3)
4 UNION(2, 3)
5 UNION(5, 3)
```

a) Gælder der at $\text{FIND-SET}(2) = \text{FIND-SET}(5)$ efter udførelse af linie 1-5 i koden C ?

Svar a) JA, da 2 og 5 er i samme mængde som 3.

Betragt følgende kode D :

```
1 for  $i \leftarrow 1$  to  $n$ 
2   do MAKE-SET( $i$ )
3 for  $i \leftarrow 2$  to  $n$ 
4   do UNION(1,  $i$ )
```

b) Antag at ovennævnte union operationer er implementeret som *disjoint-set forests* (CLRS kap. 21.3 og tilsvarende CLR kap. 22.3) og med *union by rank* heuristikken. Hvad er *worst-case* tiden for en UNION operation i sekvensen af operationer udført i ovennævnte kode D ?

Svar b) Der linkes hele tiden et blad til træet, dvs. højende af træet med element 1 forbliver konstant, dvs. tiden er $O(1)$.

c) Lad A være en tabel med n heltal, sorteret i stigende orden. Beskriv en effektiv algoritme, der opbygger et balanceret binært søgetræ for tallene i A . D.v.s. højden skal være $O(\log n)$. Angiv tidskompleksiteten af din løsning.

Svar c) Det midterste element, $A[\lfloor n/2 \rfloor]$ i A gøres til rod, og træerne bestående af elementerne til venstre $A[1.. \lfloor n/2 \rfloor - 1]$ og højre $A[\lfloor n/2 \rfloor + 1..n]$ for dette element beregnes rekursivt og gøres til henholdsvis venstre og højre undertræ. Hvis A kun består af 0 elementer returneres *NIL*. Tiden er udtrykt ved $T(n) = 2T(n/2) + O(1)$, dvs. $O(n)$. Højden af træet er $O(\log n)$, da rekursionsdybden kun er $\log n$ (for hver rekursivt kald halveres n).

d) Lad A være en tabel med n heltal, der ligger mellem 1 og $3n$. Beskriv en effektiv datastruktur, der gemmer heltallene i A i lineær plads, så datastrukturen efterfølgende kan understøtte operationen:

SEARCH(k): Returnerer **true** hvis heltallet k er i A og ellers **false**.

Angiv tidskompleksiteten for opbygning af datastrukturen og **SEARCH**.

Svar d) Initialiserer en table S med $3n$ indgange til 0 i alle indgange. Løb A igennem, og sæt $S[A[i]] = 1$ for alle $1 \leq i \leq n$. **SEARCH(k)** returnerer nu blot $S[k] = 1$. Tiden for initialisering og udfyldning af S er klart $O(n)$ og **SEARCH** $O(1)$.