Introduktion til algoritmik og datastrukturer

IT-højskolen i København

9. januar 2002

Dette eksamenssæt består af 3 opgaver med i alt 13 delopgaver. De 13 delopgaver vægtes ens i bedømmelsen. Du har i alt 4 timer til din rådighed. Husk at angive sidetal, navn og cpr.-nummer på alle sider i din besvarelse. Eksamenssættet består af 4 nummererede sider.

CLR refererer til "Introduction to Algorithms" af Cormen, Leiserson og Rivest, 18. tryk, 1997. CLRS refererer til "Introduction to Algorithms" Second Editon af Cormen, Leiserson, Rivest og Stein, 2001.

I besvarelser, hvor der skal angives effektive algoritmer, lægges der i bedømmelsen vægt på den beskrevne løsnings asymptotiske tidskompleksitet. I spørgsmål, hvor der skal angives tidskompleksitet, skal denne udtrykkes i O-notation med mindst mulig vækstrate.

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 ← 1 to n
2     do A[i] ← i
3        B[i] ← 1
4 Merge-Sort(A, 1, n)
5 Counting-Sort(A, B, n)
6 for i ← 1 to n
7     do Counting-Sort(A, B, n)
```

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

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

Betragt proceduren Add1(A, m)

```
1 if m ≥ 2
2     then for i ← 1 to m
3               do A[i] ← A[i] + 1
4          Add1(A, ⌊m/2⌋)
```

Betragt koden B:

```
1 for i ← 1 to n
2     do A[i] ← 0
3 Add1 (A, n)
```

c) Angiv den samlede tid for udførelsen af Add1 i linie 3 for koden B.

d) Angiv i O-notation den største værdi i tabel A efter udførelsen af Add1 i linie 3 for koden B.

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.

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.

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

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

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.

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 ← 1 to 5
2     do MAKE-SET(i)
3 UNION(1, 3)
4 UNION(2, 3)
5 UNION(5, 3)
```

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

Betragt følgende kode D:

```
for i ← 1 to n
    do MAKE-SET(i)
for i ← 2 to n
    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?

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.

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 datastrukuren 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.