Introduction to algorithms and data structures

The IT University of Copenhagen

January 17. 2003

This examination assignment consists of 3 exercises with a total of 13 subexercises. Subexercises are given equal weight in the grading. You have 4 hours to answer all 13 subexercises. Remember to write the page number, your name and your CPR.-number on each page of your written answer. The complete assignment consists of 5 numbered pages.

CLRS refers to "Introduction to Algorithms" by Cormen, Leiserson, Rivest and Stein, Second Edition, 2001.

For exercises that asks for effecient algorithms, you should also give the asymptotic time complexity of your algorithm, and the asymptotic complexity of the specified solution will be taken into account when grading. Exercises asking for time complexity must be answered using O-notation with least possible asymptotic growth.

Exercise 1

a) Assume $f(n) \leq n^2$, for $n > 10^9$, and $f(n) = n^4$, for $1 \leq n \leq 10^9$. Is it true that $f(n) = O(n^3 \log n)$. Justify your answer.

Consider the following procedures:

```
F1(n)
i ← 1
while i < n
      do i ← 2i
```

```
F2(n)
i ← 1
while i < n
      do for j ← 1 to i
            do k ← 1
         i ← 2i
```

```
F3(n)
for i ← 1 to n
     do j ← n
        while j > i
              do j ← j − 1
```

b) Assume that the procedures above are only called with an integer $n > 0$. What is the time complexity expressed in n for each of the procedures above?

Let A be an array representing a **max-heap** with $n > 0$ elements as defined in CLRS chapter 6. Let LEFT, RIGHT, etc. also be defined as in chapter 6. Consider the following procedures:

INC1(A, i)

$n \leftarrow$ *heap-size*$[A]$

$A[i] \leftarrow A[i] + 1$

if LEFT(i) $\leq n$

  then INC1(A,LEFT(i))

if RIGHT(i) $\leq n$

  then INC1(A,RIGHT(i))

INC-MORE(A, i)

$n \leftarrow$ *heap-size*$[A]$

INC1(A, i)

if LEFT(i) $\leq n$

  then INC-MORE(A,LEFT(i))

if RIGHT(i) $\leq n$

  then INC-MORE(A,RIGHT(i))

Recall that for a **max-heap** A we have that $A[\text{PARENT}(i)] \geq A[i]$, for all $i > 1$.

c) Assume $A[\text{PARENT}(i)] > A[i]$, for all $i > 1$, before a call to INC-MORE($A, 1$). Notice that ">" is used in the assumption. Is A a **max-heap** after the call to INC-MORE($A, 1$)? Justify your answer.

d) What is the time complexity of a call to INC-MORE($A, 1$)? Justify your answer.

Exercise 2

Consider the following directed graph :
$H = (\{1,2,3,4,5\}, \{(1,2),(1,3),(2,3),(3,4),(4,1),(4,5)\})$.

a) Give the adjacency-list representation for H in the same way as Figure 22.2 page 528 in CLRS.

Let $G = (V, E)$ be a directed graph. Let $G_1 = (V, E - \{e\})$, $e \in E$. If G_1 is acyclic, then e is denoted a *bad* edge.

b) List the bad edges from H.

In the following we assume all graphs are represented as adjacency-lists.

Let $T = (V, E_T)$ be a minimum spanning tree for the undirected weighted connected graph $G = (V, E)$. Let $E' \subseteq \{(v, w) | v, w \in V\}$ be a set of weighted edges. Let $G_1 = (V, E_T \cup E')$ and $G_2 = (V, E \cup E')$.

c) Prove that the weight of a minimum spanning tree for G_1 equals the weight of a minimum spanning tree for G_2.

Let $G = (V, E)$ be a directed weighted graph, with positive edge weight. Let $V_1 \subseteq V$.

d) Describe an efficient algorithm which given G, V_1, and $v \in V$, finds a node in V_1 which has a minimum distance to v among the nodes in V_1.

Exercise 3

Consider the following directed graph :
$J = (\{1, 2, 3, 4, 5\}, \{(1, 2), (3, 1), (3, 4), (4, 2)\})$.

a) Give a topological sorting of the graph J.

b) Let T be a binary rooted tree, where the nodes have associated keys. Give an efficient algorithm which checks if the keys in the tree satisfies the **binary-search-tree property**, see CLRS chapter 12, page 254

Let A be an array of $n > 0$ integers, each integer belonging to $\{1, 2, ..., n\}$. Consider the following procedure:

```
JUMP(A, i)
n ← length[A]
for j ← 1 to A[i]
      do k ← 1
if A[i] + i ≤ n
   then JUMP(A,A[i] + i)
```

c) What is the time complexity for a call to JUMP(A, i)? Justify your answer.

d) Let G be a directed graph. Assume the edge weight on each edge is either 1/2 or 2/3. Give an efficient algorithm, which given two nodes $s, t \in V[G]$, computes the length of a shortest path from s to t in G.

Let $G = (V, E)$ be a directed graph. Each node v has initially a unique positive integer value, $U(v)$, associated.

Consider the following program:

```
FUN(G)
while there is an edge (v, w) where U(v) > U(w)
      do U(w) ← U(v)
output for each node v the value U(v)
```

e) Assume the graph G is acyclic. Describe an efficient algorithm which given G outputs the same as FUN(G).