Winter Term 2001/02
November 5, 2001

Prof. Dr. Susanne Albers
Institute of Computer Science

Algorithms Theory

Problem Set 2

To be submitted on November 19, 2001 (in the lecture)

Problem 1: (10 points)

A *d-ary heap*, for some fixed integer $d \geq 2$, is a d-ary tree, whose nodes contain keys of some key type, such that for all nodes $v \neq$ root, $key(v) \geq key(parent(v))$. Implement a priority queue as a d-ary heap, using an array $H[1 \ldots n]$ of key type. Formulate algorithms for all the priority queue operations and analyze their running time.

Problem 2: (10 points)

a) In the lecture we studied a binary counter with the operations *Init* and *Increment*. Suppose that there is a third operation *Decrement*, which decrements the counter by 1. Show that a sequence of n operations, starting with *Init*, could cost as much as $\Theta(n \log n)$.

b) Let Φ be a potential with $\Phi(D^{(i)}) \geq \Phi(D^{(0)})$, for all i, and $\Phi(D^{(0)}) \neq 0$. Show that there is a potential function Φ' such that (a) $\Phi'(D^{(0)}) = 0$ and $\Phi'(D^{(i)}) \geq 0$ and (b) the amortized costs using Φ' are the same as the amortized costs using Φ.

c) Consider a sequence of n operations on a given data structure. The cost of the i-th operation is i if i is an exact power of 2 and 1 otherwise. Analyse the amortized costs using a potential function.

Problem 3: (10 points)

We are given a stack implementation with the following operations:

- *CreateStack():* Creates an empty stack.
- *Push(S, i):* Inserts element i at the top of the stack S.
- *Pop(S):* Returns and deletes the element at the top of the stack S.
- *Top(S):* Returns the element at the top of the stack S without deleting it.
- *Empty(S):* Checks whether or not S is empty.

Each of these operations takes $O(1)$ time. Show how to implement a queue with two stacks such that the amortized time of each of the following operations is $O(1)$:

- *CreateQueue():* Creates an empty queue.

- *Enqueue(Q, i):* Inserts element i at the end of the queue Q.
- *Dequeue(Q):* Returns and deletes the element at the front of the queue Q.
- *Empty(Q):* Checks whether or not Q is empty.

Problem 4: (10 points)

a) Prove the following properties of the Fibonacci numbers:
(i) $F_{i+2} = 2 + \sum_{j=2}^{i} F_j$ and (ii) $F_{i+2} \geq ((1+\sqrt{5})/2)^i$.

b) Construct a sequence of operations such that the height of a tree in a Fibonacci heap is linear in the number of elements in the heap.