TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in data structures
Fall Semester, 2006/2007

Homework 2, November 18, 2006

Due on Monday December 4. Don’t forget to keep a copy of the homework.

1. We denote the $i$th character of string $s$ by $s[i]$. Given a string $s$ we define a partition of $s$ (sometimes called Lempel-Ziv parsing) into blocks $s_1, \ldots, s_k$ as follows. The first block $s_1$ is the first character of $s$. Say we defined $s_1, \ldots, s_i$ and the last character of $s_i$ is $s[\ell]$. The $i$th block $s_{i+1}$ is the shortest substring of $s$ starting with $s[\ell + 1]$ which does not equal to one of the blocks $s_1, \ldots, s_i$. If there is no such substring starting with $s[\ell + 1]$ then $s_{i+1}$ is the suffix of $s$ from $s[\ell + 1]$ until the last character of $s$. For example if $s = AABABBBABAABABBB$ then its partition into blocks is $|A|AB|ABB|B|ABA|ABAB|BB$.

a) Describe an efficient algorithm to find the partition of a string $s$ of length $n$. Analyze your algorithm.

Notice that each block $s_j$ consists of a previous block plus one additional character. For example the block $s_5$ in the partition above is equal to $s_2$ followed by “B”. We can encode $s$ using a sequence of pairs, one for each block. The pair encoding $s_j$ consists of an index of a previous block $s_i$, $i < j$, and a character $\sigma$ such that $s_j = s_i \sigma$. (The index is 0 in case $s_i$ does not exist.)

b) What is the length (asymptotically) as a function of $n$ of the encoding of a periodic string $s$ consisting of $n$ repetitions of AB ($s = (AB)^n$)?

c) What is the Burrows-Wheeler Transform of the string $s = (AB)^n$? We denote this string by $BWT(s)$.

d) Use the Move-To-Front (MTF) encoding to convert $BWT(s)$ to a string of integers, what is the sequence of integers that you obtain?

e) If you use Huffman code to encode the integer string $MTF(BWT(s))$ into a bit sequence, what would be the length of the sequence?

f) Can you suggest an improvement to the compression algorithm that uses the BWT followed by MTF, followed by Huffman encoding, so it performs better on periodic strings as $s$ above.

2. We maintain items in an array subject to the following operations.

1. insert(i): insert item $i$ in the next free slot of the array.

2. delete: delete an item from the last occupied slot of the array.

Describe an implementation for these operations such that each operation takes $O(1)$ amortized time and the length of the array at any time is proportional to the number of occupied slots. (I.e.
there should be constants $c_1$ and $c_2$ such that $c_1n \leq L \leq c_2n$ at any time where $L$ is the length of the array and $n$ is the number of occupied slots.) Prove that your implementation indeed achieves these time bounds.

3. Consider an implementation of Fibonacci heaps without cascading cuts (all other details are as shown in class, the only difference is that delete and decrease-key just cut the subtree and do not continue with cascading cuts). For any large enough $m$ show a sequence of $m$ operations on heaps of size at most $n$ such that the average cost of an operation is as high as possible. (By $m$ large enough we mean larger even than some function of $n$.)

4. For any positive integer $n$, give a sequence of Fibonacci heaps operations that creates a Fibonacci heap consisting of just one tree that is a linear chain of $n$ nodes (make your sequence as short as you can).

5. Brouvka’s algorithm for finding a minimum spanning tree works in iterations each of which performs the followings.
   1) Each vertex picks the edge of minimum cost incident to it and colors it blue.
   2) Add all blue edges to the minimum spanning tree and contract them.

The algorithm terminates when only one vertex remains.

   1. Describe a straightforward implementation of Brouvka’s algorithm that runs in $O(m \log n)$ time. (As usual $m$ is the number of edges and $n$ is the number of vertices.)

   2. Using the idea of Gabow, Galil, Spencer, and Tarjan to group edges into packets, describe an implementation of Brouvka’s algorithm that runs in $O(m \log \log n)$ time.