Up to **four** *people can work on an exercise together. But each of you should be able to explain the solutions to the TA (Bremser). Write your names* **and** *the name of your group (time, TA) on the sheets. Staple them together*

Assignment 6

Deadline: December 8, 2003

Solve at least two of the following four exercises *but* one of them must be Exercise 1 or Exercise 2.

Exercise 1

The multikey quicksort algorithm for string sorting runs in time $O(D + n \log n)$ where D is the total length of the distinguishing prefixes and n is the number of strings. We ask you to show that this makes it a good suffix sorting algorithm for a string $S[0 : n)$ in the best case and *on the average*, i.e., characters are chosen uniformly and independently at random from an alphabet $\{1, \ldots, \sigma\}$.

1. A naive use of the multikey quicksort algorithm would make copies of all suffixes and hence needs $\Omega(n^2)$ space and time. Outline how this can be avoided.
2. Show that $D = O(n)$ in the *best case* if $\sigma \geq n$, i.e., if you can choose a particular input for given n.
3. Consider two nonoverlapping substrings $s_i = S[i : i + k)$ and $s_j = S[j : j + k)$. Show that the probability that $s_i = s_j$ is σ^{-k}.
4. Consider two substrings $s_i = S[i : i + k)$ and $s_j = S[j : j + k)$ with $i \neq j$. Show that the probability that $s_i = s_j$ is σ^{-k} even if s_i and s_j overlap. Hint: $P[A \cap B] = P[A] \cdot P[B|A]$
5. Show that the probability that there are two identical substrings of length k is bounded by $n^2\sigma^{-k}$. Hint: $P[A \cup B] \leq P[A] + P[B]$
6. Show that there is at most a $1/n$ probability that there are any two suffixes with a common prefix of length $3 \log_\sigma n$ or longer.
7. Now show that the average case running time of the multikey quicksort algorithm for suffix sorting is bounded by $O(n \log n)$.[1] Hint: Assume the runtime of an algorithm is bounded by $f(n)$ with probability $p(n)$ and otherwise it is bounded by $g(n)$, then its expected runtime is bounded by $(1 - p(n))g(n) + p(n)f(n)$.

Exercise 2

The *Burrows-Wheeler* transform of a string T of length n is defined as follows: Assume that T is terminated by a special sentinel character \$ that is considered smaller than any other character. Now consider the $n \times n$ matrix of characters where each row contains a different cyclic rotation of T. Sort the rows lexicographically. $BW(T)$ is the rightmost column of the sorted matrix.[2] For example, $BW($`mississippi$` $=$ `ipssm$pissii`$)$.

[1] Note that this is optimal for ordered alphabets.

[2] The most important application of the BW tranform is text compression. It turns out that T can be reconstructed from $BW(T)$. Moreover, $BW(T)$ contains a lot of repeated characters and is easy to compress using simple compression methods. For details refer to P. Ferragina and G. Manzini, An experimental study of a compressed index, Information Sciences, 135 (2001) 13–28.

What is $BW(\texttt{abracadabra})$? Assume you are given a suffix array of T. Explain how to derive $BW(T)$ in linear time using the suffix array.

Exercise 3
Construct a finite automaton that enters an accepting state at every occurrence of the strings `mama` or `papa`.

Exercise 4
Give the Morris-Pratt failure function for the pattern `ananas`.