- This problemset has *two* questions, and one (substantially more difficult) bonus question. The programming problems may be harder, or require more time, than their point value suggests.
- Please type your solutions to the written component and send a pdf file to **REDACTED**. The pdf filename should be "EDS-A2-<your_user_name>.pdf"
- The deadline is **9.05.2014** anywhere on Earth.

(80) 1. **Computing the Inverse:** Given a permutation π, let $\pi(i)$ denote the position that the i-th element is swapped to according to π: e.g., if $\pi = (2, 3, 4, 1)$ then $\pi(1) = 2$, $\pi(2) = 3$, etc. One way of representing a permutation is by storing an array containing the values of each $\pi(i)$. Since each $\pi(i)$ is a number between 1 and n, we can clearly store this array using $n\lceil\log(n+1)\rceil$ bits (assuming that we know n). This array representation allows us to access each $\pi(i)$ in constant time. However, it is often useful to also be able to compute the *inverse* $\pi^{-1}(i)$, which is the value j such that $\pi(j) = i$. Obviously, we could store these values explicitly in *another* array, but that would take twice the space. Alternatively, we can apply $\pi(i)$ repeatedly, examining the entire cycle to determine $\pi^{-1}(i)$. This uses no extra space (beyond $\Theta(1)$ words of working space), but could take linear time if π contains long cycles. Your task: show how to use the FKS hashing scheme to speed up the computation of $\pi^{-1}(i)$, for any $i \in [1, n]$. You should be able to get a space/time trade off. How much extra space *in words* do you need to compute $\pi^{-1}(i)$ in $\Theta(t)$ time for some parameter $1 \le t \le n$? Don't try to work out the exact constant factors: an asymptotic bound will suffice. What about in bits? Hint: it might be helpful to complete Question 2 before trying to count the number of bits.

(20) 2. **Universal Hashing:** This is a programming question, but you need not submit it on SPOJ; instead you will run a small experiment, plot a graph of the results, and include that in your pdf file. The set up: for each $n \in \{2^4, ..., 2^{10}\}$ generate a set of n distinct integers, which we will call *keys*, uniformly at random from the range $[0, 2^{20} - 1]$. For each $m \in \{n^2/8, n^2/4, n^2/2, n^2, 2n^2, 4n^2, 8n^2\}$ create a hash table of size m. Now, attempt to hash the keys into the hash table using the strategy described in the Carter and Wegman paper (Ref. 12 on the course website; the Wikipedia article on Universal Hashing might be easier to read, in particular the section called "Constructions"). A single *test* will count the number of attempts you make before succeeding to hash without collisions. What to plot: you can create a single plot, where each n will be represented by a different curve. The x-axis will be m/n^2, and should be plotted on a logarithmic scale. The y-axis of the plot will be the average number of attempts you had to make during a test to hash the n keys without collisions into the table. The y-axis should also be plotted on a log scale. Connect the points for a given n with a line in your plot. Experiment with how many sets of keys you generate as well as how many tests you perform per set, in order to reduce noise.

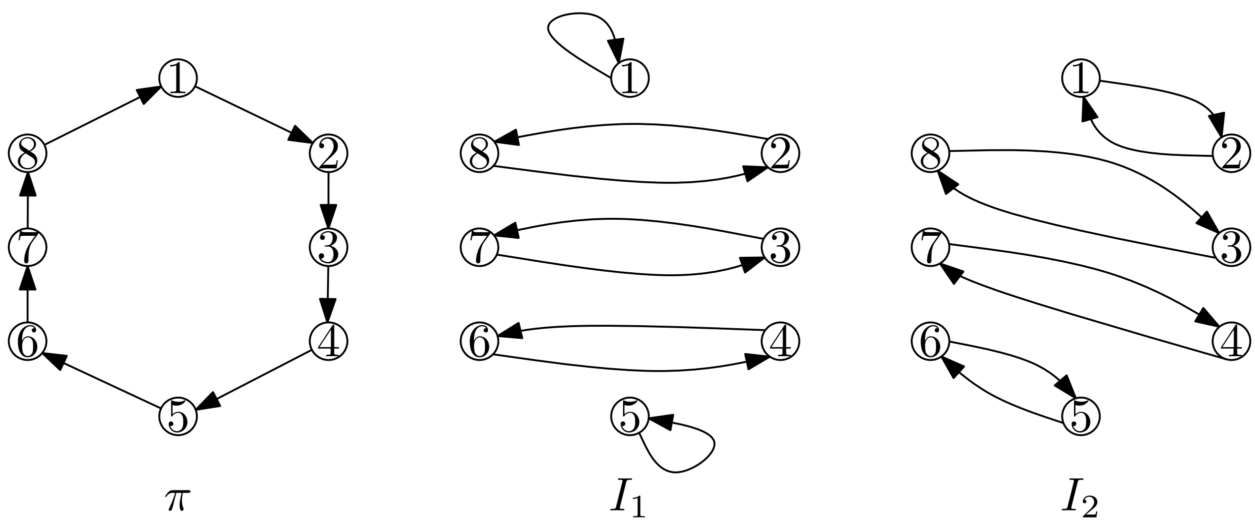

Figure 1: Illustration of the permutations π, and the two involutions I_1 and I_2 into which π can be decomposed.

(SPOJ) 3. **Bonus Programming Question: Involutions** In class we saw how involutions can be used to speed up multikey search in the implicit model. We applied several levels of involutions to the keys in each D_i, but each involution was *not arbitrary*, so length of cycles induced in the final permutation of D_i's keys were also not arbitrary. In this programming question we will show that composing two arbitrary involutions can lead to a permutation with linear length cycles!

Suppose we are given an arbitrary permutation π. The problem we wish to solve is to construct two involutions I_1 and I_2 such that, when composed, are identical to π. For example, consider the permutation $\pi = (2, 3, 4, 5, 6, 7, 8, 1)$, which is just a cycle of length 8. We can construct the following involutions $I_1 = (1, 8, 7, 6, 5, 4, 3, 2)$ and $I_2 = (2, 1, 8, 7, 6, 5, 4, 3)$, as shown in Figure 1. Thus, for each $i \in [1, n]$ we have $\pi(i) = I_2(I_1(i))$: for example $\pi(2) = 3$, $I_1(2) = 8$ and $I_2(8) = 3$. This illustrates one method of decomposing such a permutation, and there are other examples on the problem website. `http://www.spoj.com/DS/problems/INVDECOM/`