- This problemset has *four* questions.
- To get the credit for questions marked as SPOJ, you must get them accepted on `http://www.spoj.com/AOS`, but you **don't** have to send any explanation!
- For other questions, either send the solutions to `gawry1+aos@gmail.com`, or leave them in the envelope attached to the doors of my office (room 321).

1. Let $d(i,j)$ be the edit distance between $s[1..i]$ and $t[1..j]$.

   (a) Prove that $d(i+1,j+1) \geq d(i,j)$.

   Solution: If $d(i+1,j+1) = d(i,j)+1$ or $d(i+1,j+1) = d(i,j)$, i.e., the last edge on the path corresponding to $d(i+1,j+1)$ is diagonal, we are done. Because of the symmetry we can then assume that the last edge was horizontal, so $d(i+1,j+1) = d(i+1,j)$. Say that the last $k \geq 1$ edges are horizontal, so $d(i+1,j+1) = d(i+1,j+1-k)+k$. Then the previous edge is:

   1. diagonal, so $d(i+1,j+1-k) = d(i,j-k)$ or $d(i+1,j+1-k) = d(i,j-k)+1$. Then $d(i,j-k) \leq d(i+1,j+1)-k$. But then we can go from $(i,j-k)$ to (i,j) following k horizontal edges, so $d(i,j) \leq d(i+1,j+1)-k+k = d(i+1,j+1)$.
   2. vertical, so $d(i+1,j+1-k) = d(i,j+1-k)+1$. Then we can go from $(i,j+1-k)$ to (i,j) following $k-1$ horizontal edges, so $d(i,j) \leq d(i+1,j+1)-k-1+(k-1) < d(i+1,j+1)$.

   (b) Prove that $d(i+1,j+1) \leq d(i,j)+1$.

   Solution: $d(i+1,j+1)$ is the minimum of $d(i,j+1)+1$, $d(i+1,j)+1$, and $d(i,j)$ or $d(i,j)+1$. Hence $d(i+1,j+1)$ is at most $d(i,j)+1$.

2. Describe how to combine the Hirschberg's and Myer's algorithms to output the path corresponding to the edit distance using $\mathcal{O}(nD)$ time and $\mathcal{O}(n)$ space, where $D = d(s,t)$ and $n = |s|+|t|$.

   Solution: The Myer's algorithm can be used to compute the the part of any row containing values not exceeding D in $\mathcal{O}(n)$ space and $\mathcal{O}(nD)$ time. As in the original Hirscheberg's algorithm, we choose $x = \frac{n}{2}$, and compute two values for each node y in the x-th row: the cheapest path from $(0,0)$ to (x,y), and from (x,y) to (n,n). All those values can be computed by running the Myer's method twice, once going top-bottom, and once going bottom-top. Then we choose y minimizing the sum of the two values, and recurse on two resulting subproblems. Note that in each recursive call we need to know the value

of D. We can determine the initial value of D using the doubling, and then notice than whenever we select y, we actually have the edit distance of each pair of strings that we are going to recurse on available (they are exactly the computed values). Furthermore, those two edit distances sum up to D. Hence the running time can be described as $T(n, D) = T(\frac{n}{2}, D') + T(\frac{n}{2}, D - D') + \mathcal{O}(nD)$, which solves to $T(n, D) = \mathcal{O}(nD)$ by the same argument as the the one used during the lecture.

(SPOJ) 3. Let $d \in \{1, 2, ..., |s|\}$ be a period of a word s iff $s[i] = s[i + d]$ whenever both $s[i]$ and $s[i + d]$ are defined, i.e., $i = 1, 2, \ldots, |s| - d$. You are given a word s. Print all periods of this word in decreasing order.

Solution: Compute the π function of s. Then output $\pi[|s|]$, $\pi[\pi[|s|]]$, $\pi[\pi[\pi[|s|]]]$, ...

(SPOJ) 4. Extra credit: we say that a sequence of numbers $x_1, x_2, \ldots x_k$ is zigzag if no three of its consecutive elements create a nonincreasing or nondecreasing sequence. More precisely, for all $i = 1, 2, \ldots, k - 2$ either $x_{i+1} < x_i, x_{i+2}$ or $x_{i+1} > x_i, x_{i+1}$. You are given two sequences of numbers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$. The problem is to compute the length of their longest common zigzag subsequence.

Solution: Assume that the sequence we are looking for begins with an increase, and then run your solution twice, once for the original input, and once for its negation. For each i and j such that $a_i = b_j$ compute two zigzag subsequences of $a_1, a_2, \ldots, a_i$ and $b_1, b_2, \ldots, b_j$: the longest even and the longest odd one. It turns out that computing two such subsequences is enough. Computing them efficiently requires looking more closesly at what happens when we are calculating them in a row-by-row, column-by-column manner.