TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in data structures
Spring Semester, 2011/2012

Homework 1, March 17, 2012

Due on March 28. Please keep a copy of the homework !

1. In class we applied the SMAWK algorithm to matrices which have at least as many columns as rows ($m \geq n$). Lets consider the situation when $m < n$:
a) Describe a **really simple** algorithm for this case that runs in $O(m + n)$ time. (This answer should be 2 lines long.)
b) Describe an algorithm for this case that runs in $O(m(1 + \log(\frac{n}{m}))$ time.

2. Given a sequence of n real numbers $a_1, a_2, a_3, \ldots, a_n$, describe an algorithm to determine indices i and j ($1 \leq i \leq j \leq n$) such that the sum $s(i, j) = a_i + a_{i+1} + \cdots + a_j$ is a maximum. (Full credit would be given to a linear time algorithm that uses the SMAWK algorithm, please prove that the requirements for applying the SMAWK algorithm hold. A direct linear time algorithm or a less efficient algorithm will be given partial credit depending upon their quality.)

3. Given a totally monotone $n \times m$ matrix M (n rows, m columns) describe a simple $O(mn+m^2)$ time algorithm that sorts all the rows of M. Please describe the algorithm precisely, prove that it is correct, and prove that it runs within the desired time bound.
(Hint: one way that works is insertion sort.)

4. Here is an alternative to Wilber's algorithm for the least weight subsequence problem. Recall that Wilber's algorithm as described in class works on a matrix G with rows $0, \ldots, n-1$ and columns $1, \ldots, n$.

In each iteration instead of using a second call to SMAWK on the submatrix $G[c+1 : p-1, c+2 : p]$ with our current values of $F[j]$ for $j \in [c + 1, p]$ we act as follows.

We maintain an array $N[j], j = 1, \ldots, n$, initialized to ∞. In each iteration after the first call to SMAWK we perform the following

```
for j = c+1 to p do F[j] = min(N[j],F[j])
end for
for j = c+2 to p do
  if G[j-1,j] < F[j] then
    F[j] := G[j-1,j]
    break
  else
    if G[j-1,p] < F(p) then
      N[j+1:p] := F[j+1:p]
```

```
        break
      end if
    end if
  end for
  if j <= p then
    c := j
    r := j-1
  else /* here j=p+1, we completed the previous for loop without breaking out */
    c := p
    r := max(r, the row where F(p) is obtained)
  end if
```

Notice that this variation does not use an entry $C[i, j]$ before the minimum of column i is determined.

a) Establish the correctness of this algorithm by stating a precise set of invariants that hold after each iteration (and possibly within an iteration) and proving that they indeed hold by induction on the steps of the algorithm.
b) What is the running time of this algorithm ?

5. Show how to use segment trees or range trees (or any other data structure) to solve the following problems. Argue why your solution is correct. Analyze its construction time, space, and query time.
a) You are given a set of n points in R^3. Build a data structure that can find the point with largest z coordinate over a rectangle in the xy-plane (this rectangle is the query).
b) Given a set of points in R^2, build a data structure such that given $y_1 \leq y_2$ and α, can report all points (x, y) such that $y_1 \leq y \leq y_2$ and $x \leq \alpha$. (Can you find a solution with $O(\log n + k)$ query time where k is the number of reported points ?)