Algorithms and Data Structures
Winter Term 2015/16
Exercises for Units 18 & 19

1. Assume a UNION-FIND strategy as described in class is used where each partition is described by a forest of rooted trees and the root of each tree in that forest is the representative element of the corresponding set. Moreover, assume that path compression is used and arbitrary linking.

   Show that if starting with an initial forest of n one-node trees a sequence of ℓ unions is executed followed by a sequence of m finds, then the total time taken is $O(\ell + n + m)$.

   Does the stronger bound of $O(\ell + m)$ also hold?

2. Assume the same setup as in question 1, but the unions do not all happen before all finds, i.e. these operations are arbitrarily mixed. In class we showed an $O((m+n)\log n)$ worst case time bound for this case.

   Show that for any integer $k > 1$ also the following worst case time bound holds:

   $$O(\ (m + (k-1)n)\lceil \log_k n \rceil\ )$$

   Given n and m, which k optimizes this bound?

   Hint: Use a divide-and-conquer approach in the analysis that does not use divide evenly but in dependence on k.

3. Prove the following lemma:

   - Let $\mathcal{F}$ be a rank forest with node set X and maximum rank r. Let s be some integer, let $X_{\leq s}$ be the set of nodes with rank at most s, and let $X_{>s}$ be the set of nodes with rank exceeding s. Then

     a) $(X_{\leq s}, X_{>s})$ is a dissection.

     b) $\mathcal{F}(X_{\leq s})$ is a rank forest with maximum rank at most s.

     c) $\mathcal{F}(X_{>s})$ is a rank forest with maximum rank at most $r - s - 1$.

     d) $|X_{>s}| \leq |X|/2^{s+1}$.

     e) The number of roots in $\mathcal{F}(X_{\leq s})$ is at least $(s+1)|X_{>s}|$.

4. The *off-line minimum problem* asks us to maintain a set T of elements from the domain $\{1, \ldots, n\}$ under the operations *insert* and *deleteMin*, which returns and deletes the minimum element from the set. We are given a sequence S of n `insert` and m `deleteMin` calls, where each key is inserted exactly once. We wish to fill an array `extracted`$[1, \ldots, m]$ of size m, where `extract`$[i]$ yields the value of the i-th call of `deleteMin` call. The problem is "off-line", that is, we can process the entire sequence S before determining any extracted key.

   a) An example sequence is given below: Here, an `insert` is given by a number and a `deleteMin` is represented by the letter D:

   $$4, 8, D, 3, D, 9, 2, 6, D, D, D, 1, 7, D, 5$$

   What would be the correct `extract` array to be returned?

For $i \in \{2, \ldots, m\}$, let $K_i \subseteq \{1, \ldots, n\}$ be the set of elements inserted after the $(i-1)$-st and before the i-th `deleteMin` call. Moreover, let K_1 be the elements inserted before the first, and K_{m+1} the elements inserted after the last `deleteMin` call. In the example above, $K_1 = \{4, 8\}$, $K_2 = \{3\}$, $K_3 = \{2, 6, 9\}$, $K_4 = K_5 = \emptyset$, $K_6 = \{1, 7\}$, $K_7 = \{5\}$. Knowing $K_1, \ldots, K_{m+1}$, we use the following algorithm:

```
1: procedure OFF-LINE-MINIMUM(m, n)
2:     for i from 1 to n do
3:         Determine j with i ∈ K_j.
4:         if j ≠ m + 1 then
5:             extracted[j] ← i
6:             Let ℓ be the smallest value greater than j for which the set K_ℓ exists
7:             Set K_ℓ ← K_ℓ ∪ K_j and remove the set K_j
8:     return extracted
```

b) Argue that the array returned by this method is correct.

c) Describe how to use a union-find data structure to implement the above algorithm efficiently and analyze the worst-case running time of your implementation.