02282: Reexam 2018

Inge Li Gørtz Philip Bille

1 Formalities

The reexam consists of two parts exactly as the ordinary exam: hand-in of mandatory exercises and an oral exam. As in the ordinary exam the final grade is an overall evaluation of your mandatory exercise and the oral exam combined.

- The oral exam takes place August 16th 2018.
- Deadline for handing in the exercises is August 9th 2018 at midnight.
- The collaboration policy for the mandatory exercises is the same as in the course (see the webpage).
- The hand-in should be a single PDF-file.
- Hand in on CampusNet in the group for the reexam (the group is not created yet).
- Label the exercises by the same names as we have.
- Write us an email if you are planning to participate in the reexam (and also remember to sign up officially).

2 Exercises

You should hand in the following exercises.

1 The Subsequence Problem A string P is a subsequence of string T if we can obtain P from T by removing 0 or more characters in T. For instance, `aba` is a subsequence of `bcadfbbba`. Let T be a string of length n with characters from an alphabet of size σ. We are interested in efficient data structures for T that supports the following query:

- subsequence(P): return true if P is a subsequence of T and false otherwise.

Solve the following exercises.

1.1 Give a data structure that answers queries in $O(|P|)$ time and uses little space. **Hint:** a good solution depends on both the size of the alphabet and the length of T.

1.2 Give a data structure that uses $O(n)$ space and supports fast queries. The query time should depend on P.

2 The Highest α-Free Ancestor Problem Let T be a rooted tree with n nodes. Each leaf in T is assigned a label from a set of colors C. Given a node $v \in T$, the subtree rooted at v, denoted $T(v)$, is the tree consisting of v and all descendants of v. A subtree $T(v)$ is *α-free* if it does not contain a leaf with label α. We are interested in efficient data structures for T that support the following query. Let ℓ be a leaf in T and α a color in C.

- HFA(ℓ, α): return the highest ancestor a of ℓ such that $T(a)$ is α-free.

Give a linear-space data structure for T that supports fast HFA queries. Ignore the preprocessing time.

3 Placement of bars You are given a metric with n nodes. Your goal is to place bars at nodes, such that no node is to far from a bar. That is you want to minimize the maximum distance anyone has to a bar. Some places are more expensive to place a bar on than others. For each node j you have a price p_j that indicates how much it costs to place a bar at node j. You have a budget for building bars of B. Give a 3-approximation algorithm for the problem.
Note: Let the radius of a solution be the maximum distance anyone has to a bar in that solution. You may assume that you know the optimal radius r.

4 Supersuffixes Let S and P be strings of lengths n and m, respectively. Both strings are from an alphabet Σ. A *supersuffix* of P is a suffix of P that occurs as a substring of S at least a many times as any other suffix of P. A longest supersuffix is a supersuffix of maximal length. For instance, if $S =$ `cocoa` and $P =$ `oco`, then both `co` and `o` are supersuffixes and `co` is a longest supersuffix. Give an efficient algorithm to compute a longest supersuffix.

5 Longest common extension in LZ78 In this exercise you are given an LZ78 encoding of a string T. The string T has length N and the encoding has n phrases. You can assume that you have the LZ78 trie over the phrases.

5.1 Longest common prefix phrase. Give an efficient $O(n)$ space data structures that can answer the query.

LCPP(P_1, P_2): Return the id of the longest phrase (of the LZ78 encoding) that is a prefix of both phrase P_1 and P_2.

5.2 Longest common extension phrase Give an efficient $O(n)$ space data structure that can answer the query.

LCEP(x): Return the id of the longest phrase (of the LZ78 encoding) that is a prefix of the suffix of T starting at position x.

You can ignore the preprocessing time.

6 2D Range Numbering Let $P \subseteq \mathcal{R}^2$ be a set of n points. The *range numbering problem* is to compactly represent P to support the following query.

- number($R = ((x_1, y_1), (x_2, y_2))$): return $|R \cap P|$.

That is given a rectangle R return the number of points from P contained in the rectangle. Give an efficient data structure for this problem.

7 String Reversal Let S be a string of length N stored in $O(N/B)$ blocks. We want to compute the *reverse* string S^R of S. Solve the following exercises.

7.1 Give an efficient algorithm to reverse S in the I/O model.

7.2 Give an efficient algorithm to reverse S in the cache oblivious model.