Weekplan: Warm Up

Philip Bille Inge Li Gørtz

References and Reading

[1] http://courses.compute.dtu.dk/02282/2021/

We recommend reading [1] in detail.

Exercises

1 Gradescope and DTU Learn

1.1 Sign up for the course on Gradescope. Please do so carefully according to the instructions on the course homepage.

1.2 Activate instant notifications of announcements on DTU Learn (if you have not already done so).

2 Range Minimum Queries Let A be an array of n integers. We want to preprocess A into a compact data structure that supports the following query:

- RMQ(i, j): return the index of a smallest integer in the subarray $A[i..j]$.

Solve the following exercises.

2.1 [w] Give a simple solution that uses $O(n)$ space and has $O(\ell)$ query time, where $\ell = j - i + 1$ is the length of the subarray.

2.2 [w] Give a simple solution that uses $O(n^2)$ space and has $O(1)$ query time.

2.3 Give a solution that uses $o(n^2)$ space and has fast query time. *Hint:* store the minima of some subarrays.

3 Dynamic Bitvectors Let B be a bitvector of n bits. We want to maintain B while supporting the following operations:

- set(i): set $B[i]$ to 1.
- unset(i): set $B[i]$ to 0.
- left(i): return the largest index i^- such that $B[i^-]$ is 1 and $i^- \leq i$

Solve the following exercises.

3.1 [w] Give a simple solution that supports set and unset in $O(n)$ time and left in $O(1)$ time.

3.2 [w] Give a simple solution that supports set and unset in $O(1)$ time and left in $O(n)$ time.

3.3 Give a solution that supports set and unset in $O(1)$ time and left in $O(\sqrt{n})$ time. *Hint:* Come up with a data structure that applies the idea of exercise **3.2** twice.

3.4 [$*$] Give a solution that supports set and unset in $O(1)$ time and left in $O(n^\varepsilon)$ time, for any constant $\varepsilon > 0$. *Hint:* recursively apply exercise **3.3**.

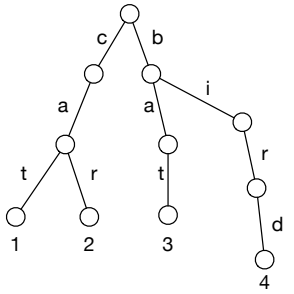

Figure 1: The trie of the collection $S = \{S_1, S_2, S_3, S_4\} = \{\texttt{cat}, \texttt{car}, \texttt{bat}, \texttt{bird}\}$.

4 Tries and String Dictionaries Let $S = \{S_1, \ldots, S_k\}$ be a collection of k strings of total length n. The strings are over an alphabet of size σ and we assume that no string is a prefix of another. The *trie* T_S of S stores the strings in a rooted tree with labeled edges such that

1. Each root-to-leaf path represents a string in S, namely, the string obtained by concatenating the labels of the edges along the path.
2. Common prefixes in S share the same path maximally.

See Figure 1 for an example. Solve the following exercises.

4.1 Give good bounds on the depth, the number of leaves, the number of nodes, the number of branching nodes, and the maximum degree of T_S.

We are interested in preprocessing S into a compact data structure that supports the following queries.

- prefix(P): determine if the string P is a prefix of any of the strings in S.
- report(P): report all the strings in S that P is a prefix of.

For instance, for the collection of strings in Figure 1 we have that prefix(`ca`) is true, and report(`ca`) $= \{1, 2\}$. Solve the following exercises.

4.2 Give a compact data structure that supports fast prefix queries. The query time should be fast in terms of the length m of the pattern string P.

4.3 Give a compact data structure that supports fast report queries. The query time should be fast in terms of the length m of the pattern string P and the size of the output occ.

5 Contagious networks You work at a large company where your boss wants to avoid spreading of a contagious disease by giving masks to the employees. The company wants to save money by only giving as few masks as necessary. It is enough if for all pairs of employees that meet during the day at least one of them wear a mask (due to the pandemic there are never more than 2 persons in a meeting). We call a meeting *unprotected* if none of the meeting participants wear a mask. You have a list of everyone that meets during the day and your task is to find the minimum subset of employees that should be given masks to ensure that there are no unprotected meetings.

Example Suppose the company has four employees: Alice, Bob, Charlie, and Debora and they have the meetings: Alice–Bob, Alice–Charlie, Debora–Bob, Debora–Charlie, Bob–Charlie. In this case, an optimal solution is to give masks to Bob and Charlie.

5.1 [w] Show how to model this as a graph problem.

5.2 Your boss suggest a simple greedy algorithm: Until there are no unprotected meetings left do the following. Give a mask to the employee participating in most unprotected meetings.

Give an example that shows that this algorithm does not always find the optimal solution.

5.3 An *disjoint set of meetings* is a set of meetings M that shares no participants. I.e., an employee can be in at most one meeting in M. Fx. is $M = \{\text{Alice–Bob}, \text{Debora–Charlie}\}$ a disjoint set of meetings.

Show that for any disjoint set of meetings M you need to hand out at least $|M|$ masks.

5.4 Consider the following algorithm: Until there are no unprotected meetings left do the following. Pick an unprotected meeting m and give masks to both persons participating in m.

Show that this algorithm obtains a valid solution and always hands out at most twice as many masks as an optimal solution.