Weekplan: Warm Up

Philip Bille   Inge Li Gørtz   Eva Rotenberg

References and Reading

[1] http://courses.compute.dtu.dk/02289/2020/

We recommend reading [1] in detail.

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 notification of announcements on DTU Learn.

2 Blocked Binary Search Let A be an array of n elements in sorted order and suppose that A is partitioned into $\lceil N/B \rceil$ *blocks* each of B consecutive elements (except possibly the last block which may contain fewer elements). Solve the following exercises.

2.1 Consider the classic binary search algorithm on A. Analyse the worst-case number of blocks that the algorithm accesses.

2.2 Suggest a new data structure for A organized into blocks that supports searching with a minimal number of blocks accesses.

2.3 [$*$] As in the previous exercise. However, now your data structure must work efficiently with *any* block size. Try to achieve the same asymptotic number of I/Os as in the previous exercise.

3 Professor Brick and The Frequent Colored Legos Professor Brick wants to find frequent colored Lego bricks in his huge box of n Lego bricks. He runs the following algorithm.

1. Initialize an empty stack S of lego bricks and empty discard pile D of pairs of brick, i.e., clear the desk to make room for the stack and the discard pile.
2. Pickup a new brick b from the box. Proceed according to one of the following cases:
   (a) If S is empty. Put b on S.
   (b) If the color of b matches the color of the bricks in S, push b onto the top of S.
   (c) If the color of b does not match the color of the bricks in S, pop the top brick t of S, click b and t together, and put them in the discard pile.
3. Repeat until the box is empty.

We say that a color c is a *majority color* if more than $n/2$ bricks in the box have color c. Solve the following exercises.

3.1 Run the algorithm on small examples. Try cases with and without a majority color. If you have some legos use those. Otherwise color some pieces of paper.

3.2 Show that if there is a majority color c then the stack will contain bricks of color c at the end of the algorithm. *Hint:* can you say something interesting about the discard pile?

3.3 Let A be an array of integers of length n. A *majority element* in A is an integer that appears more than $n/2$ times in A. Give an algorithm to find a majority element if it exists. Your algorithm should use constant space (in addition to the input A), linear time, and only perform a single pass over A.

3.4 [$*$] Generalize professor Brick's algorithm to handle colored bricks appearing more than n/k times for some integer $k > 2$.

4 Tree Labels Let T be a rooted tree with n nodes. We are interested in assigning small bit strings, called *labels*, to the nodes of T, that will allow us to answer various queries using only the information stored in labels.

4.1 We now want to assign to a label to each node in T such that given (only) the labels of two nodes u and v you can determine if u and v are adjacent. Make the worst-case length of your labels as small as you can. Precisely analyze the length of labels in *bits*, including any constant factor in front of the leading term.

4.2 As above, but now assign labels such that we can determine if u is an *ancestor* of v. *Hint:* how would you solve it if T was a path?

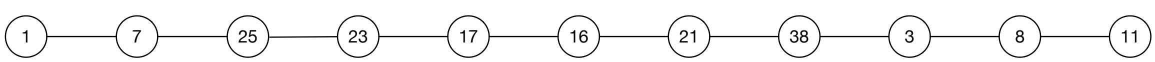
5 Distributed Path Coloring Let P be a path with n nodes, where each node has a unique number/identifier. A *3-coloring* of a path is a coloring where each node gets a color from the set $\{1,2,3\}$ such that all adjacent nodes have different colors.

In this exercise we are interested in a distributed algorithm that can compute a 3-coloring of the path. All nodes run the same algorithm in synchronized rounds: In a round each node first send messages to its neighbors, then recieves messages from its neighbors, and finally perform some computations. Consider the algorithm P3C.

Algorithm 1: P3C

Set c to the unique identifier of the node.
repeat until no colors are updated
    Send message c to all neighbors
    Receive messages from all neighbors. Let M be the set of messages received.
    if $c \notin \{1,2,3\}$ *and* $c > \max(M)$ **then**
        Let $c \leftarrow \min(\{1,2,3\} \setminus M)$
    end
end

5.1 Run the P3C algorithm on the following example:



5.2 Give an instance where the algorithm PC3 runs in n rounds before a 3-coloring is obtained.

5.3 Argue that the P3C algorithm correctly computes a 3-coloring.