Homework

Due: May 16th, 2020

- The assignment should be delivered digitally by email. Your message should be sent to **pribeiro@dcc.fc.up.pt** with subject *"[TAA HWK] - FirstName Last Name StudentNumber"*
- Your delivery should be a **zip file**, containing a **PDF report with the answers** and all additional files that were used for producing those answers
- You may work in a (small) group, but you should do your own **individual writeup**. This means you can collaborate by talking about the exercises, but **you should not copy answers or code**.
- Please **acknowledge any help you got** and state any references you consulted (including internet pages) and any students with whom you talked about the exercises.
- Answers should be **submitted until 23:59 of the due date**. Up to 24h of delay will get you a 25% penalty. 24h to 48h of delay will get you a 50% penalty. After 48h your work will not be counted.

Multiple versions of the same question

Some questions (indicated by *(*)* have multiple versions ($v1, v2, \ldots$). Refer to the **attribution sheet** (sent by email) to know which version you should answer on each question)

Balanced Binary Search Trees

Red-Black Trees

1. *(*)* Consider a complete **red-black** binary search tree of height 3 with the following 15 keys:
   - ($v1$) $\{2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30\}$
   - ($v2$) $\{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29\}$

   (a) What are the possible **black heights** of this tree?

   (b) **Draw** $\mathcal{T}$, the corresponding red-black tree with the **minimum** possible black-height.

   (c) Give a **set of operations** that would result on the tree $\mathcal{T}$ when starting in an empty tree.

   (d) *(*)* **Draw** (and **explain**) the red-black tree that would result from starting with $\mathcal{T}$ and **applying the following operation** (after the red-black tree properties are restored).
   - ($v1$) insert(15)
   - ($v2$) insert(16)

   (e) Imagine that you need to implement the operation *enumerate(a,b)* on a red-black tree, that enumerates all keys x such that $a \leq x \leq b$. **Describe an algorithm** that could do this operation in $\mathcal{O}(m + \log n)$, where m is the number of keys that appear in the output and n is the number of nodes in the tree (you can augment the tree with new attributes, but you then need to explain how to keep those attributes valid when inserting or removing keys).

Self-Adjusting Data Structures

Splay Trees

2. Consider you are using **splay trees** to store a set of keys.
   (a) What does it mean to say that each basic operation (ex: *find, insert, remove*) has an **amortized complexity** of $\mathcal{O}(\log n)$? Can any of these operations have a linear cost? Why?
   (b) Explain in what situations would **splay trees perform better than AVL or red-black trees**, justifying your answer.
   (c) *(*)* Suppose that the splay tree is *augmented* and each any node v also stores $size(v)$, the number of nodes in the subtree rooted at v. Describe an algorithm $splaySelect(k)$ that returns the k-th smallest/largest *(depends on the version)* item in $(O)(\log n)$ time
      - $(v1)$ k-th smallest
      - $(v2)$ k-th largest

Probabilistic Data Structures

Treaps

3. **Prove** that a treap is exactly the binary search tree that results from inserting the nodes one at a time into an initially empty tree, in order of increasing priority, using the standard insertion algorithm.

Skip Lists

4. **Describe and analyze** the expected running time of the best **algorithm** you can come up with to **merge** two skip lists. Merging is taking two lists and combining them into a single list (which should store all the elements in sorted order). You may assume all the elements in the skip lists are distinct.

Spatial Data Structures

Quadtrees

5. *(*)* Suppose you are using **point-region quadtrees** to store a set of n integer 2D points in the range $(0,0)$ to (a,b) (a and b *depends on the version*). What is the **maximum height** of the quadtree? **Justify** you answer and show a set of points that would give origin to that height.
   - $(v1)$ $a = b = 10^6$
   - $(v2)$ $a = b = 10^7$

kd-trees

6. *(*)* Suppose you insert the following 20 3D points on a kd-tree (depends on the version). What is the **smallest height** you could get? **Show an order** in which to insert the points to obtain that height and **draw** the corresponding kd-tree *(it's ok to use auxiliary programs implemented by you to help).*
   - $(v1)$ `https://www.dcc.fc.up.pt/~pribeiro/aulas/taa1920/20points_v1.txt`
   - $(v2)$ `https://www.dcc.fc.up.pt/~pribeiro/aulas/taa1920/20points_v2.txt`
7. Suppose you have a set of points stored on a **kd-tree** and that you want to find what are the k **closest points** from a given query point Q. Briefly describe how you could traverse the kd-tree with that goal in mind, indicating how you could make the search as efficient as possible.

LCA and RMQ

Sparse Tables

8. **Implement** a program in **any programming language** that reads as an input a tree T with n nodes and Q queries (x_i, y_i) and **outputs the LCA** of nodes x and y for each query. The program should use a **reduction to RMQ and sparse tables** to solve the problem in $\mathcal{O}(n \log n)$ preprocessing and $\mathcal{O}(1)$ for each query (also output the euler tour of the tree).

   Attach your code to the homework delivery email and provide brief comments on the code, including an example of how to run it (how can I give it input, and what is the format of the output).

1D Problems

Segment Trees

9. Imagine you are given a sequence $a[1], a[2], \ldots, a[n]$, and several queries like the following:
   $query(x, y) = max\{a[i] + a[i+1] + ... + a[j]; x \leq i \leq j \leq y\}$
   In informal terms, we are looking for the **highest sum consecutive subarray in a given range**.

   Describe an algorithm for solving this problem using segment trees, answering each query in $\mathcal{O}(\log n)$.

   You should also **exemplify its functioning** for computing the following sequence and queries (don't forget to show its corresponding segment tree).

   - $(v1)$ $a = \{-2, 3, 4, -15, 5, 3, -3, -1, 2, 1\}$, $query_1(1, 4)$, $query_2(0, 9)$
   - $(v2)$ $a = \{4, 2, -3, -4, 3, 6, -20, 5, 1, -2\}$, $query_1(6, 9)$, $query_2(0, 9)$

String Matching

Knuth-Morris-Pratt (KMP) Algorithm

10. In this question you will be asked to modify the standard KMP algorithm.

    (a) Describe a **modification** of the Knuth-Morris-Pratt (KMP) algorithm in which you only want to report the **first occurrence** and the pattern can contain any number of **wildcard** symbols '*', each of which matches an arbitrary substring. For instance, the pattern "al*thm*s" matches "ilovealgorithmsarelife": the first wildcard matches *"gori"* and the second wildcard matches an empty substring. The algorithm should run in $\mathcal{O}(n + m)$, where n is the size of the text and m the size of the pattern.

    (b) **Implement** a program in **any programming language** that corresponds to the algorithm you described. It should have as an input a pattern P and Q query texts and computes all first occurrences of P in each of the queries. It should print the π function (or its equivalent).

    Attach your code to the homework delivery email and provide brief comments, including an example of how to run it (how can I give it input, and what is the format of the output).

Suffix Trees

11. A **palindrome** is a word which reads the same backward as forward, such as "madam". Given a string S of size n, **describe an algorithm using suffix trees** that can compute $P(S)$, the size of the **largest palindrome which is a substring** of the word, and indicate its runtime and memory complexity (assume you can build a suffix tree or array in $\mathcal{O}(n)$).

    For instance, $P(\text{"}banana\text{"}) = 5$, because the largest substring which is also a palindrome is "anana".

    Besides describing the algorithm, you should exemplify its functioning for computing $P(\text{"}adambcmada\text{"})$.

Suffix Arrays

12. *(*)* Suppose you already have available a suffix array SA of a string s with length n, and its corresponding LCP array (containing the longest common prefix of consecutive suffixes). **Describe an algorithm (based on suffix arrays)** that computes $L(s, k)$ the **length of the largest substring that appears in** s **at least** k **times**. Indicate its runtime complexity.

    For instance, $L(\text{"}banana\text{"}, 2) = 3$ because "ana" appears 2 times and has length 3 (no other larger substring appears at least 2 times).

    Besides describing the algorithm, you should exemplify its functioning for computing the following string (don't forget to show its corresponding suffix and LCP arrays).

    - ($v1$) $L(\text{"}senselessness\text{"}, 2)$
    - ($v2$) $L(\text{"}mississippis\text{"}, 2)$