Tutorial 10

3-sided range search, the good-suffix heuristic, and problems on range-search and on string matching.

CS 240E W23
University of Waterloo
Monday, March 27

1. The material on 3-sided range search is in section 8.5 (line 5426) [Biedl].

2. The material on the good-suffix heuristic is in section 9.4.3 (line 6092) [Biedl].

Range search

3. **Range tree space.** Prove or disprove: for any set of n points in general position, the range tree uses $\Omega(n \log n)$ space.

4. **Priority Search Tree [optional].** Show how to build a priority search tree in $O(n \log n)$ worst-case time. Note: in fact, $O(n)$ worst-case (using just CS240E material) is possible.

5. **kd-tree.** Create a set of n points and a range-query such that doing the range-query on the kd-tree of the points requires $\Omega(\sqrt{n})$ boundary nodes.

6. **Quad-tree.**

   (a) For an arbitrary n, construct a set of points such that the quad-tree has at least n nodes, and give a range-search query such that all nodes are visited, and not a single point gets returned.

   (b) Assume that T is a quad-tree with at least two points such that during some range-search, there is at least one outside node and at least one inside-node (the example from Module 8, slide 11 satisfies this). What is the minimum possible height of T?

   The example has height 3, so the question is whether height 3 is always required, or whether this could also happen with height 2 or even height 1?

String matching

7. **Cyclic shift.** Given two strings w and x of length n, determine if w can be obtained by cyclically shifting the characters of x. For example, the algorithm should return `true` if the input is `alloy` and `loyal`, and `false` if the inputs are `tarot` and `otter`. Your algorithm should take $O(n)$ time for two strings of length n.

8. **Boyer-Moore.** Apply the Boyer-Moore algorithm to the following pattern and text. Show

   (a) with only the bad-character heuristic,

   (b) [optional] with the good-suffix heuristic.

T :	d	a	y	s	a	y	m	a	y	a	a	a	y	b	a	y	l	a	y	k	a	y	r	a	y	j	a	y
P :	d	a	y	d	a	y	h	a	y	a	y	a	y															
(a)																												
(b)																												

9. **Most common substring.** Let s be a string of length n and let $\mathcal{T}_s$ denote the corresponding suffix tree. For an integer parameter $1 \leq l \leq n$, give a $O(n)$ time algorithm that finds a most commonly occurring substring of length l in s.