University of Waterloo
CS240 Fall 2017
Assignment 4

Due Date: Wednesday, November 15, at 5:00pm

Please read `http://www.student.cs.uwaterloo.ca/~cs240/f17/guidelines.pdf` for guidelines on submission. This assignment contains both written problems and a programming problem. Submit your written solutions electronically as a PDF with file name a04wp.pdf using MarkUs. We will also accept individual question files named a04q1w.pdf, a04q2w.pdf, ... , a04q5w.pdf if you wish to submit questions as you complete them.

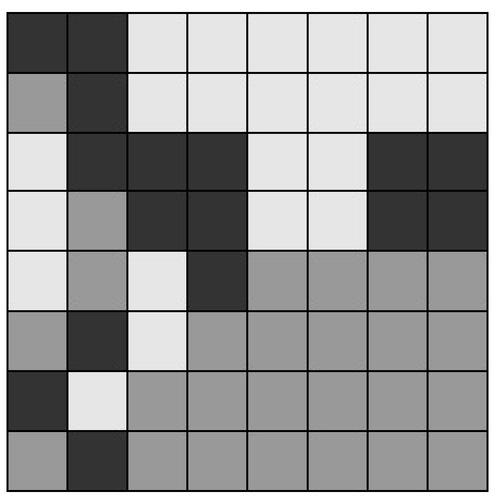
Problem 5 contains a programming question; submit your solution electronically as a file named `kdpartition.cpp`.

Problem 1 Quadtrees [6+6+8=20 marks]

For all parts of this question, use the convention that each internal node of a quadtree has exactly four children, corresponding to regions NW, NE, SE and SW, in that order.

a) Give three 2-dimensional points such that the corresponding quadtree has height exactly 7. Give the (x, y) coordinates of the three points and show the quadtree. (Do not give the plane partitions.)

b) One application of quadtrees is image compression. An image (picture) is recursively divided into quadrants until the entire quadrant is only one colour. Using this rule, draw the quadtree of the following image. There are only three colours (shades of grey). For the leaves of the quad tree, use 1 to denote the lightest shade, 2 for the middle shade and 3 for the darkest shade of grey.



c) Another application is to compare two images. Given two black and white images (i.e. each pixel of the image is either 0 or 1) each of size $2^k \times 2^k$ store as quadtrees, give algorithms for the following operations:

  i) Union. If corresponding pixels are both 0, then their union is 0; otherwise 1.

  ii) Intersection. If corresponding pixels are both 1, then their intersection is 1; otherwise 0.

Problem 2 kd-Trees [6+6=12 marks]

Consider the following set of points in two dimensions:

$$S = \{(3,5),(7,8),(6,2),(8,0),(0,3),(4,6),(2,9),(9,1),(10,4),(1,7),(15,10)\}.$$

a) Draw the analogous plane partition diagram and kd-tree.

b) Show how a search for the points in the query rectangle $R = [2, 6.5] \times [0.5, 7]$ would proceed. More specifically, list the nodes of the kd-tree in the order that they are examined in the search.

Problem 3 Range Trees [3+6+6=15 marks]

a) Draw a 2-dimensional range tree for the following set of points:

$$\{(7,88),(12,19),(22,33),(27,29),(28,9),(31,99),(42,66)\}$$

b) Assume that we have a set of n numbers (not necessarily integers) and we are interested only in the number of points that lie in a range rather than in reporting all of them. Describe how a 1-dimensional range tree (i.e., a balanced BST) can be modified such that a range counting query can be performed in $O(\log n)$ time (independent of k). Briefly justify that your algorithm is within the expected runtime.

c) Next, consider the 2-dimensional case where we have a set of n 2-dimensional points. Given a query rectangle R, we only want to find the number of points inside R, not the points themselves. Explain how to modify the Range Tree data structure discussed in class such that you can answer any of these counting queries in $O((\log n)^2))$ time. Briefly justify that your algorithm is within the expected runtime.

Problem 4 Pattern Matching KMP [8+6+4=18 marks]

a) For each of the following pattern strings, determine the Knuth-Morris-Pratt failure array:

  i) `P=SHE SELLS SEASHELLS`

ii) `P=ABRACADABRACADABRA`

iii) `P=ABRACADABRACAPABRA`

iv) `P=ABRACADABRACABABRA`

b) For the next part of this question, we will extend the pattern-matching capabilities of the Knuth-Morris-Pratt algorithm by adding support for the special wildcard character `?`, which matches any single character. For example, the pattern `B?T` matches `BAT, BOT, BST` or any other three-character string that starts with `B` and ends with `T`. For simplicity, you may assume that the target string contains no `?` characters.

i) Give pseudocode for the KMP search function and the function that computes the KMP failure array, taking into account the special behaviour of the character `?`.

ii) Give the failure array for the following pattern strings:

```
P=S?E SE?LS SEASHELLS
P=ABRACADABRAC???BRA
```

Problem 5 Programming [12+6=18 marks]

a) Implement an $O(n \log n)$ algorithm to construct a kd-tree for 2-dimensions. Your algorithm should read $2n+1$ integers from standard input, separated by whitespace. The first integer is the number of points. The remaining $2n$ integers are the points themselves, given as x and y coordinates. Use the algorithm on slide 13 of module 8. You may assume that no two points lie on the same horizontal or vertical line.

The idea of the algorithm is to first do some preprocessing: sort the points by both their x and y coordinates. For this preprocessing step you may use a standard library function and assume it runs in $O(n \log n)$ time. You may also need additional preprocessing. Next, call a recursive function (which you write) to produce the tree shown on slide 14 of module 8. Actually, your program does not need to construct the tree, but rather only needs to print to standard output the n points stored at the nodes of the tree, in the order they are visited during a pre-order traversal.

Here is an example input:

```
3
3 4
2 2
1 1
```

The points are p_1, p_2, p_3 = (3,4),(2,2),(1,1). These three points correspond to the following *kd*-tree:

```
                                        |
                                        |     * p1
                                        |
                                        |
                                        |
         [p2]                           |
        /    \                          |
     [p3]    [p1]                       |
                                        * p2
                                        |
                                        |
                                   *    |
                                   p3   |
```

Thus, for this example, your algorithm should print out:

```
2 2
1 1
3 4
```

Your implementation must be in C++ and run in $O(n \log n)$ time.
Submit the code for your `main` function, along with any helper functions, in a file called `kdpartition.cpp`.

b) Justify the $O(n \log n)$ runtime of your algorithm. To do this, you should show that for a subproblem on k points, your recursive algorithm has running time satisfying the following recurrence: $T(k) \leq 2T(\lfloor k/2 \rfloor) + O(k)$. In other words, if you are splitting the region with a vertical line according to the x-coordinate, explain how you produce the two sets of points in each region sorted according to their y-coordinate in time $O(k)$. Submit this solution with the written part of your assignment.