CSC263H1: Problem Set 7

Due Tuesday November 26 before 10pm

General instructions

Please read the following instructions carefully before starting the problem set. They contain important information about general problem set expectations, problem set submission instructions, and reminders of course policies.

- Your problem sets are graded on both correctness and clarity of communication. Solutions that are technically correct but poorly written will not receive full marks. Please read over your solutions carefully before submitting them.
- Each problem set may be completed in groups of up to three. If you are working in a group for this problem set, please consult the articles on collaboration and plagiarism on posted on quercus.
- Solutions must be typeset electronically, and submitted as a PDF with the correct filename. **Hand-written submissions will receive a grade of ZERO.**

  The required filename for this problem set is **problem_set7.pdf**.
- Problem sets must be submitted online through CrowdMark. If you haven't used CrowdMark before, give yourself plenty of time to figure it out, and ask for help if you need it! If you are working with a partner, you must form a group on CrowdMark, and make one submission per group. "I didn't know how to use CrowdMark" is not a valid excuse for submitting late work.
- Your submitted file(s) should not be larger than 9MB. You might exceed this limit if you use a word processor like Microsoft Word to create a PDF; if it does, you should look into PDF compression tools to make your PDF smaller, although please make sure that your PDF is still legible before submitting!
- Submissions must be made *before* the due date on CrowdMark.
- The work you submit must be that of your group; you may not use or copy from the work of other groups, or external sources like websites or textbooks.

Additional instructions

- If you are working in a group, remember to form your group on CrowdMark before you submit your solutions.
- Ensure that your last submission before the deadline is the submission you want graded - CrowdMark does not allow late resubmissions.

1. **[10 marks] Disjoint Set Trees.**
Let T_1, T_2 and T_3 be three different *disjoint-set forest* implementations.

- T_1 uses union-by-weight, without path compression;
- T_2 uses union-by-weight, with path compression;
- T_3 uses union-by-rank, with path compression.

Union-by-rank and path compression are described in the textbook discussion of disjoint-set forests. Union-by-weight, which described in the textbook as a heuristic for linked-list implementations, can be used as a heuristic for disjoint-set forests: the representative of the set containing more items becomes the representative of the union.

In all three implementations, when the rank or weight heuristic encounters a tie, the representative of the set that contains x becomes the new representative for UNION(x, y).

Definition: two disjoint-set forests are equivalent iff they store the same items and can be drawn the same.

Let P be a sequence of n disjoint set operations. Each operation may be a MAKE-SET, FIND-SET, or UNION operation. Suppose that we perform P on each of T_1, T_2 and T_3.

(a) What is the smallest n such that T_1 and T_2 are *not* equivalent? Fully justify your answer.

(b) What is the smallest number n such that T_2 and T_3 are not equivalent? Fully justify your answer.

Page allowance: Please neatly present your solution using 1–2 pages. Any additional pages will not be marked.

2. **[10 marks] Graphs: breadth-first search.**

   (a) Consider a *connected undirected* graph $G = (V, E)$ represented by an **adjacency matrix** $G.A$. Design an algorithm that finds a vertex whose removal (deleting the vertex and all its incident edges) does NOT disconnect the graph. Your algorithm *must* be based on breadth first search (BFS).

   i. Describe your algorithm in clear and precise English.

   ii. Explain why your algorithm works correctly.

   iii. Analyse the worst-case running time of your algorithm and express the result in asymptotic notation.

   HINT: Think about the BFS-tree.

   Page allowance: Please neatly present your solution using a single page. Any additional pages will not be marked.

3. **[0 marks] Row Data Structure. This extra question is for your own edification. If you attempted it, we encourage you to upload your solution, for our data, but it will not be marked.**

   (a) Consider the following abstract data type ROW that describes the pixels on one row of a screen of width M. An object S of ROW is a subset of $\{1, \ldots, M\}$, which denotes those pixels that are illuminated. A *line* is a maximal[1] set of consecutive integers in S. For example, If $S = \{3, 4, 5, 11, 12, 13, 14, 17\}$, then the row contains three lines: the line $\{3, 4, 5\}$ whose endpoints are 3 and 5, the line $\{11, 12, 13, 14\}$ whose endpoints are 11 and 14, and the line $\{17\}$ whose endpoints are both 17.

   ROW supports two operations:

   - ON(x) illuminates pixel $x \in \{1, \ldots, M\}$, i.e. it adds x to S, if it is not already in S.
   - ENDPOINTS(x) returns the endpoints of the line containing x. If $x \notin S$, it returns (0,0).

   Give a data structure that implements ROW such that the worst case complexity of any sequence of m ON and ENDPOINTS operations is $O(m \log^* n)$, where n is the size of S, and $\log^*$ is the iterated logarithm. Justify why your data structure is correct and has the required complexity.

   (b) Give a data structure that implements ROW such that

   - its space complexity is $O(\ell)$ and
   - the time complexity of ON and ENDPOINTS is $O(\log \ell)$,

   where ℓ is the number of lines in S. Briefly justify why your data structure is correct and has the required complexity.

[1] If $Q \subset S$ is a set of consecutive integers, and there is no other subset of S that contains Q, then Q is maximal.