EECS 340 Algorithms

2018 Spring Semester

Homework 3

Due on February 28, 2018, 12:45pm

This assignment covers sorting, selection, union-find, and graph algorithms.

1. **Sorting Leftover Elements** Suppose that you are given an array consisting of n sorted elements followed by $f(n)$ elements in an arbitrary order, where $f(n) \in O(n^{1-\epsilon})$ for some $\epsilon \in (0, 1)$. Describe a method to sort the array in $O(n)$ time. ($\star\star$)

2. **Theory: Sorting Algorithm Runtimes**

   (a) Suppose that we assign a code to each permutation of n elements, where each code is a natural number, and the set of all such codes is of the form $[0, k)$ for some k dependent on n. Let $f(n)$ be the number of bits in the binary encoding of the k corresponding to a given n. Give a tight asymptotic bound (a $g(n)$ such that $f(n) \in \Theta(g(n))$) on $f(n)$. Note that if you do this, the asymptotic runtime of sorting n elements will be bounded below by $\Omega(g(n))$, for reasons described in a footnote. [1]

   (b) Read the footnote from the previous part. Why doesn't the derived lower bound ($\Omega(g(n))$, for the particular $g(n)$ you found) from the previous part hold for non-comparison-based sorting algorithms like radix sort (even when that is performed in base-2)? ($\star\star$)

3. **Post Office Placement** Solve A-9.3 in the text. ($\star\star\star$)

4. **Sorting Sequences** Solve C-9.4 in the text. ($\star\star$)

5. **Median From Two Lists** Solve C-9.10 in the text. Note that merging or concatenating the arrays would take $O(n)$ time. ($\star\star\star$)

[1] Intuitively, we can view sorting as a search for the permutation which rearranges the unsorted list into a sorted one. There is only one such permutation, and every comparison operation gives us only one bit of information, assuming all elements are distinct. If the runtime of comparison-based sorting was asymptotically faster than $g(n)$, our worst adversary could apply some collection permutations to the original list, and we would be unable to distinguish between these permutations.

6. **Warm-up: Graphs**

   (a) Solve R-7.1 in the text. ($\star$)

   (b) Solve R-13.2 in the text. ($\star$)

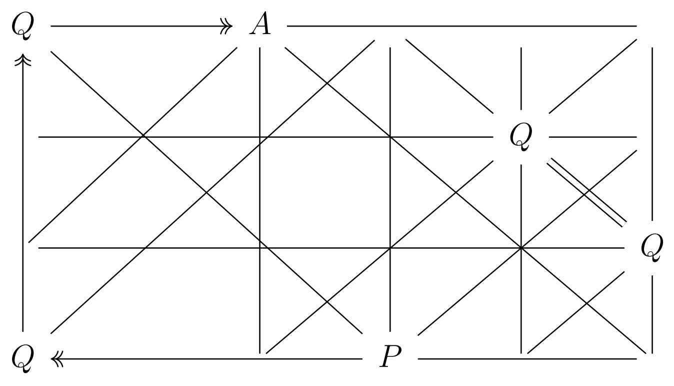
   (c) Given the graph

$$\begin{tikzcd}
A \arrow[d] & B \arrow[l] \arrow[rd] & \\
C \arrow[r] \arrow[ru] & D \arrow[d] \arrow[ld] & E \arrow[ld] \\
E & G \arrow[luu] &
\end{tikzcd}$$

      i. Draw the graph where all nodes are annotated with the order (expressed by natural numbers) that they are visited in a **depth-first** search from A. ($\star$)

      ii. Draw a representation of the tree generated by the DFS annotated over the previous part. ($\star$)

7. **Application: Chess** A *standoff* of queens on a generalized $n \times n$ chessboard is a collection of queens, where every queen in the standoff can attack some other queen in the standoff, and for every selection of a ″protagonist″ and an ″antagonist″ queen from the standoff, there is a sequence of attacks [2] on other queens which eventually leads the protagonist queen to defeat the antagonist. As an example, if we have queens in the following configuration, where ″P″ is a protagonist queen, ″A″ is an antagonist queen, and each ″Q″ is another queen, ″P″ could carry out the marked (with directional arrows) sequence of moves to attack ″A″.



   Here, the rightmost two queens do not belong to the standoff defined by the other queens, but they are in a standoff of their own.

[2] No positioning moves other than direct attacks are permitted. We also assume that the protagonist queen is somewhat like an anime protagonist in that the other queens are assumed to be stationary during the entire sequence of moves.

Suppose that the number of queens on the board is $O(n^{2-\epsilon})$ for some $\epsilon \in (0,1)$. Design an algorithm which finds the largest queen standoff in $O(n^2)$ time. You may assume that the input is an $n \times n$ boolean two-dimensional array where $True$ indicates the presence of a queen, and the desired output is a list of ordered pairs (i,j) representing the positions of queens in the largest standoff. $(\star \star \star)$

8. **Application: A World of Voxels** [3] DigShaft is the latest and greatest open-world block-based 3d building game for the 12-year-old gamer market [4] You are busy developing a mod for this game to add electricity and (super) conducting metals, and quickly realize the importance of keeping track of the collections of all connected conducting metal blocks in order to propagate electrical current. Assume that the world is (partially) represented by an $M \times N \times P$-dimensional array of boolean values, where $True$ indicates the presence of a metal block. Let $n = MNP$.

   (a) Describe a method to preprocess the array in $O(n\log(n))$ time to create some data structure fitting within $O(n)$ space which allows queries about whether a metal block at a given position is connected [5] to a metal block in another position in $O(\log(n))$ time. $(\star \star \star)$

   (b) Working from your answer in the previous part, what if we have already preprocessed the voxels in the world, but we want to efficiently support additions of metal blocks to the world, while still maintaining information about which blocks are connected? [6] Describe how to do this. What is the asymptotic runtime of an addition? $(\star)$

   (c) Keeping with everything you wrote above, describe a way to support deletions of blocks. Also describe any practical problems with the runtime of your method if such problems exist. $(\star\star)$

9. **Practice : Topological Sorting**

   (a) Find four distinct topological orderings of the nodes in the following directed

[3] A *voxel* is to a 3d grid as a pixel is to a 2d grid. See `https://en.wikipedia.org/wiki/Voxel` for a nice illustration of this, which may also help in reasoning about the problem.

[4] This is in no way meant as an attack on some other trademarked open-world building game, nor its players. The author has played said other game extensively.

[5] Two blocks are connected if they share a face, or if there is some sequence of blocks connected in this manner between them.

[6] One-by-one, where the addition is given by the index triple representing the position that the metal block is added to.

acyclic graph, presented here as a 3×3 grid. $(\star)$

$$\begin{tikzcd}
(0,0) & (0,1) \arrow[l] & (0,2) \arrow[l] \\
(1,0) \arrow[u] & (1,1) \arrow[l] \arrow[u] & (1,2) \arrow[l] \arrow[u] \\
(2,0) \arrow[u] & (2,1) \arrow[l] \arrow[u] & (2,2) \arrow[l] \arrow[u]
\end{tikzcd}$$

(b) Describe how to generalize your orderings to the case of a graph with the same connectivity pattern on an $n \times n$ grid. $(\star)$

10. **The following is for EECS 454 students only**. Read Section 17.2 in the textbook and then solve the following problems:
    (a) C-17.4
    (b) C-17.5

Submission

Hand in your paper in-class by the beginning of class on the due date. Always check Canvas for updates and corrections.