Assignment 2

COMP 3804, Fall 2021

Upload in Brighspace by 11:59PM on October 8, 2021

1 Guidelines

General guidelines are as follows:

1. Since we are only accepting assignments via Brightspace, no late submissions will be entertained after the cut-off time & date.
2. Please write clearly and answer questions precisely. It is your responsibility to ensure that what is uploaded is clearly readable. If we can't read, we can't mark!
3. Please cite all the references (including web-sites, names of friends, etc.) which you used/consulted as the source of information for each of the questions.
4. All questions/problems carry equal marks.
5. When a question asks you to design an algorithm - it **requires** you to
   - (a) Clearly spell out the **steps** of your algorithm in pseudo code.
   - (b) **Prove** that your algorithm is correct
   - (c) **Analyze** the running time.
6. You can assume that a graph $G = (V, E)$ uses adjacency list representation.
7. n is a positive integer, and it typically represents the size of the input to a problem.

2 Problems

1. Let S be a set of n distinct real numbers, and we are interested in finding the median element of this set. In the class (refer to the class notes), we saw an algorithm that forms groups of five elements each, finds the median element of each group, finds the median of these medians and uses that element to partition the set. The overall running time of that algorithm for finding median was $O(n)$, as the underlying recurrence $T(n) = T(n/5) + T(7n/10) + O(n)$ (and $T(c) = O(1)$ for any constant $c \leq 200$) evaluates to linear. Determine what will be the corresponding **recurrence equation** if we form $n/3$ groups, each group consisting of three elements. We will compute the median of each group, then the median of $n/3$ medians, and use that median to partition the set. Does the new recurrence equation which you obtained also evaluates to $O(n)$? Justify your answer.

2. Let S be a set of n distinct real numbers. Devise an algorithm, running in $O(n+k \log k)$ time, to report the k smallest elements of S in sorted order, where $k \in \{1, \ldots, n\}$.

3. Let S be a set of n-distinct real numbers and let $k \leq n$ be a positive integer (k may not be a constant). Design an algorithm, running in $O(n)$ time, that determines the k numbers in S that are closest to the median of S. For example, for the set $\{21, 70, 3, 1, 6, 7, 11, 2, 9, 8, 17, 13, 25\}$, median is 9, and the $k = 4$ closest numbers to 9 are $8, 7, 11$, and 6.

4. Let A and B be two sorted arrays, and each array consists of n real numbers in ascending order. Give an algorithm, running in $O(\log n)$ time, to compute the median of the elements formed by the union of the elements in both the arrays. (You may assume that the union consists of $2n$ distinct real numbers.) Hint: Assume that the median element is from A, and assume it is at index i. Then for $x = A[i]$ to be the median element, you can say something about how many elements in B need to be smaller than x (which can be checked in $O(1)$ time). The problem to solve here is how fast you can search for x in A?

5. Let A be an array consisting of n distinct real numbers. You need to find a real number x (x may not be an element of A) such that $\sum_{i=1}^{n} |x - A[i]|$ is minimized. Your algorithm should run in $O(n)$ time.

6. Consider that there is an array P containing n-photographs. It is possible that all photos are not distinct. We need to find out if there is a photograph that occurs at least $\lfloor \frac{n}{2} \rfloor + 1$ times in P (if such an element exists, then call it the *majority element*). Note that we can compare two photo's and decide in $O(1)$ time whether they are identical (i.e., the test $P[i] \overset{?}{=} P[j]$), but we don't have any mechanism to decide whether $P[i] < P[j]$ (i.e., the elements of P cannot be sorted).
a) First devise an $O(n \log n)$ time algorithm for this problem using divide-and-conquer

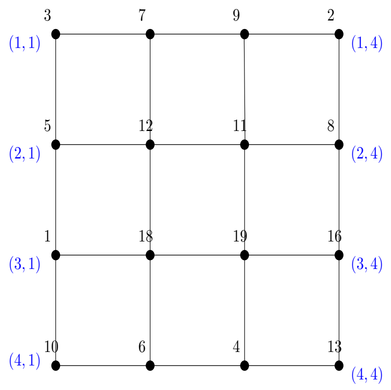

Figure 1: 4×4 grid graph G. Vertex coordinates are in blue color. Neighbors of vertex $(1,1)$ are $(1,2)$ and $(2,1)$. Vertex at coordinate $(3,3)$ with value 19 is a dominant vertex of G.

(e.g. split the array into two equal parts, and will knowing the majority element of both the halves will help in finding the majority element of the whole array?).
b) Devise an $O(n)$ algorithm for this problem. Think of forming $n/2$-pairs, each pair consisting of 2 elements, and then decide which elements to keep.

7. Let G be a $n \times n$ integer grid graph, where n is a positive integer. Each vertex of G is specified by its coordinate (i, j), where $1 \le i \le n$ and $1 \le j \le n$. The neighbors of a vertex (i, j) are the vertices $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$ (if they exist). Each vertex holds a unique number. We say a vertex (i, j) is dominant, if the number stored at that vertex is larger than the numbers stored at all of its neighbors. Show that G has a dominant vertex. Design an algorithm running in $O(n)$ time to report a dominant vertex in G. Note that G has n^2 vertices. For an illustration, see Figure 1.

8. Let $G = (V, E)$ be a simple undirected graph. Provide an algorithm running in $O(|V| + |E|)$ time that outputs whether G contains a cycle or not. If it contains a cycle - then it needs to output at least one cycle. What graph representation you have used for your algorithm. Justify why you used that, and remember to link this justification with your complexity analysis.

9. Let $T = (V, E)$ be a binary tree on n vertices. If we remove an edge e from T, we obtain two subtrees T_1 and T_2, and the number of vertices in each subtree is strictly less than n. Show that an edge $e \in E$ always exists such that each of the two subtrees T_1 and T_2 obtained after deleting e from T have at most $\lceil 2n/3 \rceil$ vertices. Design an algorithm that finds such an edge e in $O(n)$ time. See Figure 2 for an illustration.

10. For the graph in Figure 3, perform a depth-first search starting at the vertex marked A, and whenever there is a choice of vertices, pick the one the lexicographic smallest. Classify each edge as a tree edge or a back edge. Show your work!

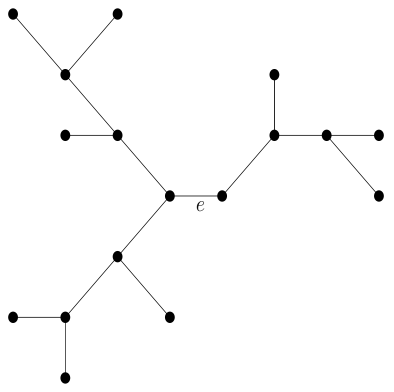

Figure 2: Removal of e partitions the tree into two subtrees consisting of 11 and 6 vertices, respectively.

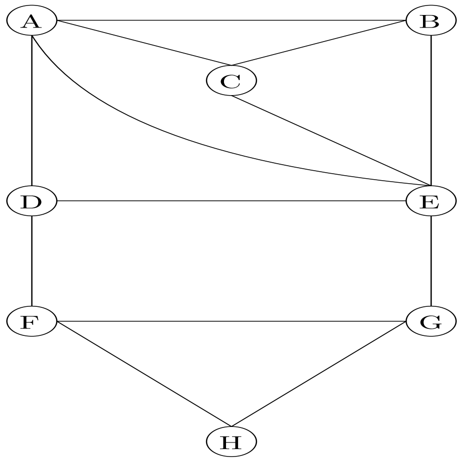

Figure 3: Graph for DFS traversal