School of Computer Science

Carleton University

Name: **A.M.** Problem Set: **3**

Course: **COMP 3804** Week: **1-5**

Problems

Consult [1, 2, 3]. The problem set consists of problems that have been asked in Test 1 and problems on DFS traversal of undirected and directed graphs.

Problem 1. *Show that the recurrence* $T(n) = T\left(\frac{\alpha}{\alpha+1}n\right) + O(n)$ *evaluates to* $O(n)$*, where* $\alpha > 1$ *is a fixed integer. You may assume that for small values of* n*,* $T(n) = O(1)$*.*

Problem 2. *Suppose that the time complexity* $T(n)$ *of an algorithm is given by* $T(n) = 2022n^{\alpha} - 2020n^{\lceil \frac{\alpha}{2} \rceil} + 2018$ *for a problem whose input size is* n*, where* $\alpha > 1$ *is a fixed integer. Show that* $T(n) \in \Theta(n^{\alpha})$*.*

Problem 3. *Let* A *be an array consisting of* n *real numbers, where a number may occur multiple times in* A*. We can assume that* n *is large and* $n = 2^k$*. We say a number* x *is* nice *if* x *occurs at least* $n/4$ *times in* A*. Answer the following:*

1. *Show that if* A *is sorted, all nice numbers in* A *can be reported in* $O(n)$ *time.*
2. *Devise an algorithm, running in* $O(n)$ *time, to report all nice numbers in* A *even when* A *is not given in the sorted order.*

Problem 4. *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.*

Problem 5. *The input consists of an array* A *of* n *distinct real numbers and a real number* t*. Devise an algorithm, running in* $O(n)$ *time, that outputs* $\sqrt{n}$ *numbers of* A *that are closest to the target* t*. (If there are ties, you can break them arbitrarily.) We can assume that* $n = 2^k$ *for some positive integer* k*. For example, for the array* $A = [21.5, 70.3, 3.4, 1, 6, 7.2, 11, 2.6, 9.4, 8, 17, 4, 14, 25, 24, 27]$ *consisting of* 16 *elements and* $t = 10$*, the* $\sqrt{16} = 4$ *closest numbers to* t *are* $8, 7.2, 9.4$ *and* 11*.*

Problem 6. *Let* A *be an unsorted array consisting of* n *integers (some of them may be negative integers). Define* $\delta(i, j) = \sum_{k=i}^{j} A[k]$ *for any pair of indices* $1 \leq i \leq j \leq n$*. Answer the following questions:*

1. *Design an* $O(n^2)$ *time algorithm to determine if there is a pair of indices* i *and* j *in* A*,* $1 \leq i \leq j \leq n$*, such that* $\delta(i, j) = 100$*.*
2. *Design a divide-and-conquer algorithm, running in* $O(n \log^2 n)$ *time, to solve Problem 5(b). I.e., determine if there is a pair of indices* i *and* j *in* A*,* $1 \leq i \leq j \leq n$*, such that* $\delta(i, j) = 100$ *in* $O(n \log^2 n)$ *time.*

Problem 7. *This problem is related to the representation of graphs. Assume that the number of edges in the graph $G = (V, E)$ is small, i.e., it is a sparse graph. In the adjacency matrix representation of G, the normal tendency is to first initialize the matrix, requiring $O(|V|^2)$ time. Is there any way we can initialize the adjacency matrix in time proportional to $O(|E|)$ and still have $O(1)$ adjacency test?*

Problem 8. *Show that in a depth-first search, if we output a left parenthesis '(' when a node is accessed for the first time and output a right parenthesis ')' when a node is accessed for the last time, then resulting parenthesization (or bracketing sequence) is proper. Each left '(' is properly matched with a right ')'.*

Problem 9. *Consider the following modified pseudo-code.*

Modified DFS Algorithm
Input: A graph $G = (V, E)$, represented by adjacency list $L[v]$ for each vertex $v \in V$.
Output: A pair of integers $(pre[v], post[v])$ assigned to each vertex v.

1. *$Clock := 1$;*
2. *for all $v \in V$ do mark v as* unvisited*;*
3. *While there exists an unvisited vertex v do SEARCH(v)*

procedure SEARCH(v)

1. mark v as *visited*;
2. $pre[v] := Clock$;
3. $Clock := Clock + 1$;
4. for each vertex w on $L[v]$ do
   if w is unvisited then SEARCH(w);
5. $post[v] := Clock$;
6. $Clock := Clock + 1$;

Answer the following questions:

1. *Execute the Modified DFS Algorithm on a couple of undirected graphs.*
2. *Suppose $G = (V, E)$ is undirected graph. Show that for any pair of nodes u and v in G, the two intervals $[pre[u], post[u]]$ and $[pre[v], post[v]]$ are either disjoint or one interval contains the other.*
3. *Execute the modified dfs algorithm on a couple of directed graphs.*
4. *Suppose $G = (V, E)$ is directed graph. Show that for any pair of nodes u and v in G, the two intervals $[pre[u], post[u]]$ and $[pre[v], post[v]]$ are either disjoint or one interval contains the other. Moreover, show that if for a directed edge $(u, v) \in E$, $pre[v] < pre[u] < post[u] < post[v]$, then there is a directed cycle in G.*
5. *Call an edge $e = (u, v)$ of a directed graph a* back edge *if $pre[v] < pre[u] < post[u] < post[v]$. Show that a directed graph has a directed cycle if and only if the modified dfs algorithm reveals a back edge.*

6. *Design an algorithm that determines whether a directed graph $G = (V, E)$ is an acyclic graph (i.e., it doesn't contain a directed cycle). Your algorithm must run in $O(|V| + |E|)$ time.*

7. *Let $G = (V, E)$ be a directed acyclic graph. Show that for any directed edge $e = (u, v) \in E$, $post[u] > post[v]$.*

8. *All vertices with no incoming edges in a directed acyclic graphs are called the* source *vertices, and all the vertices that have no outgoing edges are called the* sink *vertices. In any directed acyclic graph, can you say what property the vertex with the largest post number satisfies? the vertex with the smallest post number? Does the ordering of the vertices with respect to decreasing post number results in a linear order?*

Problem 10. *Let $G = (V, E)$ be a directed graph given in the adjacent matrix representation. Define the square of G to be the graph $G' = (V', E')$ where $V' = V$ and $(u, v) \in E'$ if and only if there is a directed path consisting at most two edges between u and v in G. Given G show how you can compute G' efficiently.*

References

[1] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms.* 3rd. ed., MIT Press, 2009.

[2] S. DasGupta, C. Papadimitriou, V. Vazirani. *Introduction to Algorithms.* McGraw Hill.

[3] A. Maheshwari. *Notes on Algorithm Design*, Chapter 1, `https://people.scs.carleton.ca/~maheshwa/Notes/DAA/notes.pdf`