McGill University COMP251 Fall 2009 **Instructor: Phuong Nguyen**

Practice problems

Note: Some problems from the textbook that are used in assignments will not be repeated in this list.

The numbers such as 8.3-1 refer to exercises in the textbook: Cormen, Leiserson, Rivest, Stein Introduction to Algorithm (2nd edition) McGraw-Hill (2001), ISBN: 0-07-013151-1. It is available online at http://library.books24x7.com/ (with McGill ID). The book ID is 3444.

1 Linear sorts: radix sort and bucket sort

8.3-1, 8.3-2, 8.3-3, 8.4-1, 8.4-2, 8-2, 8-3, 8-4, 8-5.

2 Graph searchs

Note: In the textbook, for DFS $d[v]$ is the "discovery time" of a vertex v. This is the same as $s[v]$ used in lecture (which stands for "starting time"). Also, consult your notes and the handouts for the proofs of correctness for topological sort and strongly connected components.

2.1 Graph representations

22.1-2.

Problem 2.1 (CLRS 22.1-1)**.** *Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees?*

Problem 2.2 (CLRS 22.1-3)**.** *Describe efficient algorithms for computing the transpose G^T of a directed graph G from G, for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.*

Problem 2.3 (CLRS 22.1-5)**.** *The* square *of a directed graph $G = (V, E)$ is the graph $G^2 = (V, E^2)$ such that $(u, w) \in E^2$ if and only if for some $v \in V$, both $(u, v) \in E$ and $(v, w) \in E$. That is, G^2 contains an edge between u and w whenever G contains a path with exactly two edges between u and w. (Note that G^2 may contain self-loops even though G does not.) Describe efficient algorithms for computing G^2 from G for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.*

Problem 2.4 (CLRS 22.1-6)**.** *When an adjacency-matrix representation is used, most graph algorithms require time $\Omega(|V|^2)$ (because the adjacency matrix has size $|V|^2$), but there are some exceptions. Show that determining whether a directed graph G contains a* universal sink*–a vertex with in-degree $(|V| - 1)$ and out-degree 0–can be determined in time $\mathcal{O}(|V|)$, given an adjacency matrix for G.*

Problem 2.5 (CLRS 22.1-7)**.** *The* incidence matrix *of a directed graph $G = (V, E)$ is a $|V| \times |E|$ matrix $B = (b_{ij})$ such that*

$$b_{ij} = \begin{cases} -1 & \text{if edge } j \text{ leaves vertex } i \\ 1 & \text{if edge } j \text{ enters vertex } i \\ 0 & \text{otherwise} \end{cases}$$

Describe what the entries of the matrix BB^T represent, where B^T is the transpose of B.

2.2 Breadth-first search

22.2-1, 22.2-2, 22-1.

Problem 2.6 (CLRS 22.2-3)**.** *What is the running time of BFS if its input graph is represented by an adjacency matrix and the algorithm is modified to handle this form of input?*

Problem 2.7 (CLRS 22.2-4)**.** *Argue that in a breadth-first search, the value $d[u]$ assigned to a vertex u is independent of the order in which the vertices in each adjacency list are given. Using Figure 22.3 as an example, show that the BFS tree computed by the algorithm can depend on the ordering within adjacency lists.*

Problem 2.8 (CLRS 22.2-5)**.** *Given an example of a directed graph $G = (V, E)$, a source $s \in V$, and a set of tree edges $E_\pi \subseteq E$ such that for each vertex $v \in V$, the unique path in the graph (V, E_π) from s to v is a shortest path in G, yet the set of edges E_π cannot be produced by running BFS on G, no matter how the vertices are ordered in each adjacency list.*

Problem 2.9 (CLRS 22.2-7)**.** *The* diameter *of an undirected graph $G = (V, E)$ is the the maximum distance between two vertices in the graph:*

$$max\{\delta(u, v) : u, v \in V\}$$

(recall that $\delta(u, v)$ is the distance between u and v).

This problem is about computing the diameter of a tree*. Give an $\mathcal{O}(|V|)$-time algorithm for determining the diameter of a tree $T = (V, E)$.*

Problem 2.10 (CLRS 22.2-8)**.** *Let $G = (V, E)$ be a connected, undirected graph. Give an $\mathcal{O}(|V| + |E|)$-time algorithm to compute a path in G that traverses each edge in E exactly once in each direction. (Note that this path has length $2|E|$.)*

2.3 Depth-first search

22.3-1, 22.3-2, 22.3-3, 22.3-4, 22.3-5, 22.3-8, 22.3-9, 22.3-11.

2.4 Topological sort

22.4-1, 22.4-2, 22.4-3, 22.4-5.

2.5 Strongly connected components

22.5-1, 22.5-2, 22.5-3, 22.5-4, 22.5-6, 22.5-7

Problem 2.11 (CLRS 22.3-12)**.** *(***This is a starred question in CLRS. It can be solved with more ease with knowledge of strongly connected components.***)* *A directed graph* $G = (V, E)$ *is* singly connected *if for every two distinct vertices* u *and* v *in* V*, there is at most one path from* u *to* v*. Give an* $\mathcal{O}(|V| + |E|)$*-time algorithm that determines whether or not a directed graph is singly directed, when the input graph is given using adjacency list representation.*

3 Stack, queue, and linked list

10.1-1, 10.1-2, 10.1-3, 10.1-5, 10.1-6, 10.1-7.
10.2-1, 10.2-2, 10.2-3, 10.2-6.
10.4-2, 10.4-3, 10.4-4, 10.4-5.
10-1.

4 Binary search tree

12.1-1, 12.1-2, 12.1-4, 12.1-5, 12.2-1, 12.2-3, 12.2-4, 12.2-5, 12.2-6, 12.2-7, 12.2-8, 12.2-9.
12.3-2, 12.3-3, 12.3-5.
12-1, 12-2.

5 Red-black tree

Note: The textbook uses $T.nil$ as a sentinel, which is always colored black. In the lecture we do not use this, so the definition of red-black tree is a bit simpler. You may find Sections 13.3 and 13.4 in the textbook hard to follow (at least I do), so refer to your notes and the handouts.
13.1-5, 13.1-6, 13.1-7.
13.2-1, 13.2-3.
13.3-2, 13.4-4, 13.4-7.
13-2.