Assignment

PhD Course on I/O-Efficient Graph Algorithms

3rd Quarter (Spring 2010)

Note: *The solutions to all these questions can be found somewhere in the literature. You are, however, strongly encouraged to develop the solutions yourself. In order to achieve this goal, be careful which papers by certain authors ;) you read.*

Question 1: Undirected Shortest Paths

In class, we discussed Kumar/Schwabe's single-source shortest path algorithm for undirected graphs with non-negative edge weights. The algorithm achieves I/O complexity $O\left(n + \frac{m}{B}\log_2 \frac{n}{B}\right)$ but works correctly only if no two adjacent vertices have the same distance from the source s. Develop an algorithm that achieves the same complexity but is correct even without this assumption. Argue briefly that the algorithm achieves the desired I/O complexity and prove its correctness.

Hint: *Take Kumar/Schwabe's algorithm and find a way to ensure that edges between equidistant vertices are never relaxed.*

Question 2: Directed Shortest Paths

Develop a single-source shortest path algorithm for *directed* graphs with non-negative edge weights. The complexity should be $O\left(\left(n + \frac{m}{B}\right)\log_2 n\right)$. Argue briefly that the algorithm achieves the desired I/O complexity and prove its correctness.

Hint: *Combine the techniques from Kumar/Schwabe's algorithm for undirected graphs with the techniques for directed BFS and DFS.*

Question 3: Planar Topological Ordering

Develop an algorithm that computes a topological ordering of a planar directed acyclic graph in $O(\text{sort}(n))$ I/Os. Argue briefly that the algorithm achieves the desired I/O complexity and prove its correctness.

Hint: *This can be done using the same techniques as used for planar shortest paths.*

Question 4: Outerplanar Directed Depth-First Search

A graph is *outerplanar* if it can be drawn in the plane without edge intersection and so that all vertices are on the boundary of the outer face. We call such a drawing an *outerplanar embedding*. Assume you are given an n-vertex outerplanar graph, along with an outerplanar embedding represented by arranging the edges in each vertex's adjacency list in clockwise order around the vertex. Assume further that there exists a vertex s in the graph that can reach every vertex in G. Develop an algorithm that computes a directed DFS tree of G with root s in $O(\text{sort}(n))$ I/Os. Argue briefly that the algorithm achieves the desired I/O complexity and prove its correctness.

Hint: *The structure of every outerplanar graph can be represented by a tree T. The desired DFS tree can be computed using an appropriate traversal of T.*

Question 5: List Ranking Lower Bound

In class, we have ignored lower bounds altogether. In this question, your task is to prove that any "reasonable" list ranking algorithm has to take $\Omega(\text{perm}(n))$ I/Os, where $\text{perm}(n) = \min(n, \text{sort}(n))$ is the cost of permuting n data items. Since BFS, DFS, and shortest paths can be used to solve list ranking, the same bound holds for every "reasonable" algorithm for these problems. We call a list ranking algorithm *reasonable* if it has the property that, for every list node u with successor $\text{succ}(u)$, both u and $\text{succ}(u)$ are in memory together at some point during the course of the algorithm. Every list ranking algorithm we know either has this property or is easily transformed into an equivalent algorithm that has this property.

Hint: *Show how to construct, in $O(n/B)$ I/Os, a $2n$-node list from any permutation problem on n elements such that the I/O sequence performed by a reasonable list ranking algorithm can also be used to arrange the n input elements in the desired order. Then argue that this implies the claimed lower bound.*