COMP20003 Algorithms and Data Structures

Worksheet 9

[week starting 19 of September]

Second (Spring) Semester 2016

Overview

The workshop for Week 9 will start with a tutorial on Graph representation, and you will implement Adaptive Mergesort.

Tutorial Questions

Question 9.1 In this question, you are asked to construct weighted graphs from records whose format is $vertex, vertex', weight$. For example, in a weighted, directed graph, the record $1, 3, 500$ means that there is an edge from vertex 1 to vertex 3 with weight 500.

The following questions all refer to the input data below

```
1 2 200
1 3 100
1 4 500
2 3 150
2 4 300
4 5 100
```

1. Draw a weighted directed graph that reflects these edges and weights (logical representation).
2. Construct an adjacency matrix for the weighted digraph you have just drawn, including the weights. Be explicit about how you are going to handle matrix cells for which there is no information in the data, and self-loops (edges between a node and itself)
3. If you are told that these inputs refer to a weighted `undirected` graph, how would your adjacency matrix change?
4. Which representation, adjacency matrix or adjacency list, would be most suitable for this graph?

Programming exercises

Programming 9.1 An adaptive sorting algorithm takes advantage of already existing order in a file. For example, insertion sort is an adaptive sort, since it exhibits linear behavior with sorted or nearly sorted files.

`Standard` mergesort (top-down or bottom-up) are not adaptive; they take $nlogn$ time even for sorted files. However, `natural` mergesort, is adaptive.

In natural mergesort, existing "runs" in the data are used as the first units input into a bottom-up mergesort. For example, if the first ten items are already in order, and the next five items are already in order, then the first items to be dequeued and merged would be a "run" of size 10 and a "run" of size 5. No time would be wasted merging the first two elements, or the next two, etc.

since they are already in order.

Write code for natural mergesort. If you completed the code for bottom-up mergesort last week, you might build on that.

September 16, 2016