# A Directed Acyclic Graph Map in Java

# CITS2200 (Data Structures and Algorithms)

## By

## Caleb Fetzer (21384976) Reece Notargiacomo (21108155)

**A report describing:**

Implementation Choice (20%)

* Details of implementation
* Reasoning behind choice

The Complexity of Methods (40%)

(Reece Sidenote: I'm assuming we should place specific importance on the more complex algorithms)

* Describe methods
* Analysis of performance (Big-O)
* Justification for choice
* What could be done to improve choices

Test results section (20%)

* Instructions for running the tests
* The rationale behind the tests
* Evidence that the practical complexity of the methods agrees with the theoretical growth rate

# Implementation choice

We decided to implement an Adjacency List for the DAGMap, as opposed to an Adjacency Matrix. The Adjacency Lists also have lists for predecessor and successor nodes. To store the key value pairs, a set of keys is created. A rootNode value is also created, which keeps track of the first parent node (a node with children).

There were points identified, to justify the decision, such as the pro’s and con’s of each data type. The time complexity for the standard operations these data types implemented, were also addressed and compared.

­Although the Adjacency List on the surface appears to be a less appealable data type, just via inspection of the general type of methods; it uses less space depending on the size of the graph. An adjacency matrix creates space for a graph with vertices that may not exist in the graph, thus occupying more space. On very sparse graphs, an adjacency matrix uses a lot of space that is not needed.

The learning curve, for us, was trying to implement a recursive data type – to gain a better understanding of recursion and it’s role in data structures and algorithms.

|  |  |  |
| --- | --- | --- |
| **Operation** | **Adj Matrix Time Complexity** | **Adj List Time Complexity** |
| Adding edge | O(1) | O(log (n)) |
| Deleting edge | O(1) | O(log (n)) |
| Does edge exist between vi & vj? | O(1) | O(log (n)) |
| Successors of vertex | O(n) | O(k) |
| Finding path between two vertices | O(n2) | O(n + m) , m <= n |
| Examine neighbouring vertices |  |  |

(2014, Algorithm of the Week). ‘*k’ refers to the length of lists containing the successors of a vertex.*

The complexity of methods and the details of their implementation.

To note, methods that run in constant time, such as isEmpty, won’t have their time complexity detailed.

public void put (Key newKey, Value newValue):

The method calls the methods containsKey(from DAGMap), add(from java.util.Set).

This method accepts two parameters, newKey and newValue. It checks to make sure newKey does not already exist in the set of keys in the DAGMap, and if not adds it to the set.

Bibliography

A list of references

Popov, Stoimen, Accessed on May 20th 2014, Algorithm of the Week: Graphs and their Representation, <http://java.dzone.com/articles/algorithm-week-graphs-and?utm_source=twitterfeed&utm_medium=twitter&utm_campaign=Feed%3A+javalobby%2Ffrontpage+%28Javalobby+%2F+Java+Zone%29>