Umeå University
Institution för Datavetenskap

Datavetenskapens byggstenar 7.5 p
DV160HT15

OU4 Data Representation

Submitted 2016-03-03
Author: Lorenz Gerber (`dv15lgr@cs.umu.se`)
Instructor: Lena Kallin Westin / Johan Eliasson / Emil Marklund / Lina Ögren

Contents

1 Introduction

In this assignment the aim was to specify three different possible data representations for a spreadsheet application. The representations were to be described such that they could be implemented from the descriptions. Several criteria to judge the suitability of the chosen representations were discussed in the mandatory seminar *OU2*. Three of those criteria were then applied to judge the chosen data representations.

It was chosen to structure the report into five parts: First an *introduction* where the aim of the work is presented. Second, the *interpretation of problem description* including also a short reference to the chosen data representations and criterias. Third comes an *construction of the data types for each chosen data representation*. Then the *evaluation according the chosen criteria*. And finally *summarizing discussing* where also the decision for the most suitable data representation is revealed.

2 Interpretation of the Problem Description

The problem description from the course homepage defines a spreadsheet as a 'table'. Hence, a spreadsheet can be seen seen as a potentially infinite table. A requirement given in the description is that the implementation shall be more economic than an plain rectangular structure covering all non-empty table elements. From those descriptions it becomes obvious that a more concise description of problem is the *efficient implementation of a sparse matrix*.

2.1 Typical Usecases of a Spreadsheet

During a user session, text, numbers, formulas and links to other elements are stored in the table elements. In many usecases the number of filled elements will be very low compared to the virtual rectanglular set of elements that surrounds the outermost non-empty table elements, hence the table is said to have a low *fill ratio*. This is the main reason why the potential data structure should only store non-empty elements.

Another important property of a spreadsheet program is that the data structure is represented in the graphical user interface. Due to the size, there will usually just one part of the data be visible, hence blockwise value lookup for scrolling over the data table is a very common operation.

Spreadsheets are often used to prepare sorted lists. Hence rearranging the order of whole rows or columns is another operation of high importance.

When spreadsheets are used for calculation purposes, extensive linking between memeber elements of the spreadsheet is common. Links are either between two elements, much in the style of a pointer, but then can also be many elements to one cell, in the case of functions, such as calculating the sum of multiple cells.

2.2 Chosen Criteria

Time Complexity

The speed of specific operations is an important criteria. In some cases, it could decide whether a ceratin construction is feasible at or not. Whith 'speed' I would define the processing time needed at a realistic use case size of the instance in question. Hence, sometimes a bad complexity can be accepted as the typical realworld instance size happens to be in a very narrow bandwith. If possible, also the time complexity of individual operations will be asessed.

The chosen operations of interest are given below:

- Block Lookup - Get values for a block of cells. This operation will usally be applied when the user scrolls or jumps to another place in the spreadsheet. If row/column wise scrolling is assumed, typicall lookup sizes would be around 10 to 50.
- Link following - Lookup of links as used in functions.
- Deletion - Deleting elements of the spreadsheet.
- Traversal - Traversing the data structure to search for a value

Ease of Implementation

During OU2 discussion, it was agreed that a lower complexity of the implementation is generally desirable as it decreases the susceptibility for bugs and code maintenance. Below are indicated some features/operations for which the ease of implementation will be judged and compared across the different table constructions.

- Basic Structure - The data container as such
- Sort - Sorting larger blocks of non-empty elements

Memory efficency

It is understood from the problem definition that the used construction shall just store non-empty spreadsheet elements. Hence the judgement of memory efficency focuses on the amount of memory used for ‘administratiion’ of the non-empty elements, such as hash tables and non value bearing data structure.

2.3 Chosen Datatypes

The abstract datatypes that were chosen to be evaluated are:

- Array
- Binary Tree
- Directed Acyclic Graph

3 Construction of Datatypes

3.1 Array

The abstract datatype *array* ressembles the description of how a spreadsheet is defined. However, when looking at the problem description, it became obvious that the physical datatype array was not suited as representation as an *efficient representation of a sparse matrix*. Janlert [2, pp 101-103] suggest in such a case to construct the array from a vector of tables. In C, the array is a physical datatype. Typical for the datatype array, it can not be re-sized during runtime. Hence this construction interpretes the requirement of a infinite large spreadsheet by chosing a large enough length for the row dimension. This seems however to be a rather accepted implementation: ‘Microsoft Excel’, one of the arguably most popular spreadsheet applications has for example a row/column limitation of 1’048’576/16’384 [1].

3.2 Binary trees

Binary search trees offer a good time complexity. A table can be implemented as one tree for the rows and multiple trees for the columns in each row.

3.3 Directed Acyclic Graph and Hash Table

A directed graph is an n-linked structure. The nodes can be constructed as cells. The cells are created dynamically on demand. Edges represent links in the spreadsheet. They are also created dynamically. Direct access to individual nodes can be achieved by a hash table.

4 Evaluation of Data Representations

5 Discussion

And our results looked like this...

References

[1] Microsoft Excel number of rows and columns. `https://en.wikipedia.org/wiki/Microsoft_Excel#Number_of_rows_and_columns`, 2016. Accessed: 2016-02-27.

[2] L.E. Janlert and T. Wiberg. *Datatyper och algoritmer*. Studentlitteratur, 2000.