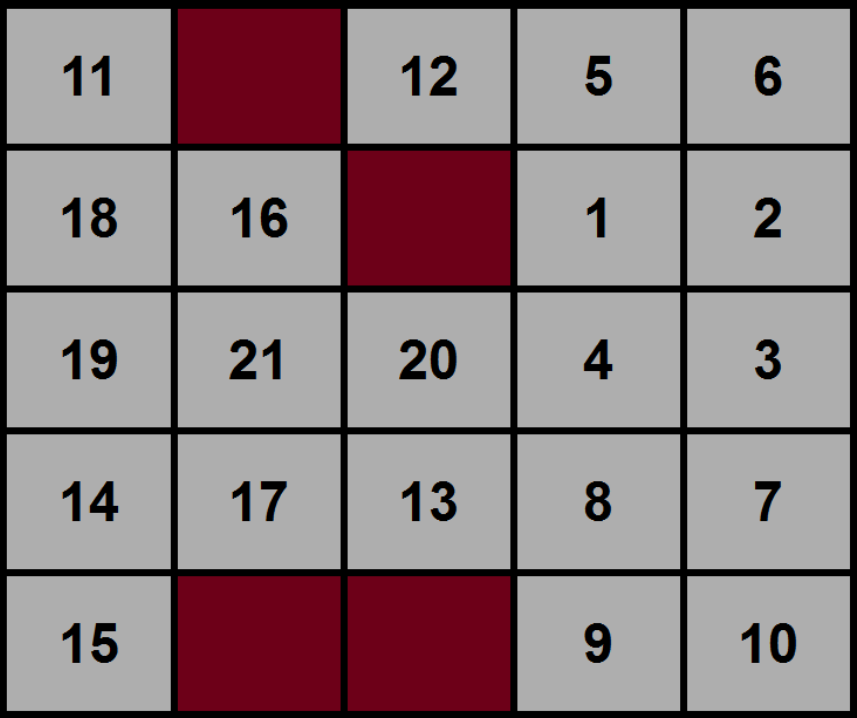
Cycle Data

dc01c



By H. Khalfan.
May31 - 2020

About this:

This is document about Cycle data, data that fill grid with certain number of cycles and with certain logical ways. The document highlights important procedures and rules to consider while dealing with it.

The document is provided as is, things and software used are all provided as they are and/or without any warranty from YPH so use them for your own preference.

Note also that: I established Cycle data, its materials, app etc for my own research, testing and logistic examination. If they are not important to you kindly ignore them.

--- H. Khalfani

May31 – 2020
for details
see this page.

Dsm, Tanzania

YPH - 2019-2020
dc01c

First publication - Oct31-2019

Contents:

Tip: Click content page number to go directly to the page. On any page, click page number to come back here.

01 Introduction

What is cycle data?

It is data of numbers deployed on grid with systematic cycles. So when you collect or refer to all those numbers together are called cycle data. You may still wonder what is this, just read next sections for more information.

Where it comes from!??

For long time since when in Primary levels, I was always interested in playing with numbers. Properties of numbers are so funny to me, the same as how I enjoy logical games like Sokoban. Numbers contain uncountable logical phenomena. Few of things that make me funny from numbers are divisibility, derivation, sequence and interconnection among basic operations.

When in college, I started thinking about how I can apply my less knowledge of numbers property to my field (ICT). Then I discovered that, numbers are very important in programming because programming field itself required more logical skills.

So I started cycle data scheme to produce certain logical pseudo random numbers to fill grid. I did this with my hand to papers then I started creating program to accomplish this task more easily for me. The first is J2ME app and the second app is PC version with more enhancement.

What is importance of cycle data?

There is nothing special about this and you should not care or concentrate too much if you are not interested in observing it. For me I use it to generate certain numbers for my own use but sometimes I use it for funny, testing how I can attentively fill entire grid by following rules. So it is possible to use cycle data scheme as game (puzzle).

Cycle data is among the scheme I created for my own use but shared with others who like to see or learn extra things. Another scheme I shared was **Henum** in 2017 which is about theory of three figures number. I use that for naming or numbering.

Can I use for encryption?

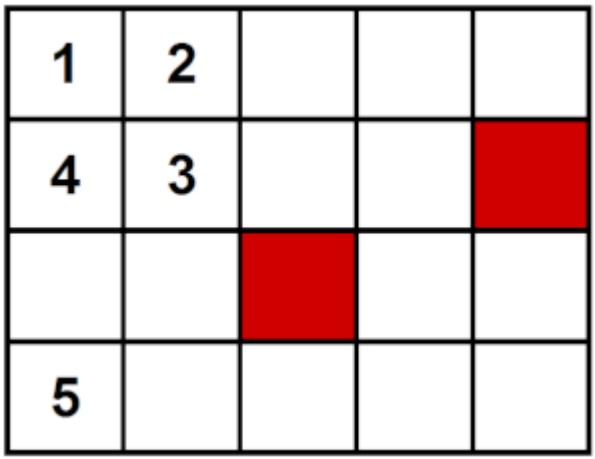
NO or yes. Although output of this scheme looks like random numbers, they are not random at all because output does not change according to time but starting position, and if there is no mistake done while deploying numbers, output remains same all the time.

02 Grid structure

Grid size:
Size of grid can be 1x1, 5x2 even 99999999x999999 but it us up to you to be able to fill it. In general any grid dimension can be used. So for example if your grid is 4x3 the you have total 12 cells to deal with. For clarification on this guide, we will always use small grids as examples. Grid dimension is denoted by **rc** that is rows times columns. Product of columns and rows determines total number of cells of grid.

Disbanding:
You can shade any cell to mark the cell so that it cannot be written during deploying. You mark cells to make them as if they are not part of grid. This process is called disbanding, but calling *shading* or *discarding* is also very appropriate.

On the right side is an example of grid (5x4). Any cell having number 1 is called starting point. Red cells are shaded cell. This grid is incomplete because it has empty cells so it has no complete cycle data. This guide will use grid generated by Cycle data desktop app. See here.



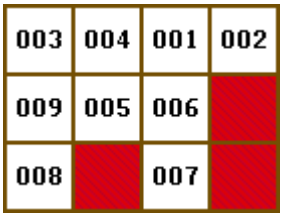
Uses of disbanding and starting cell:

Disbanding or shading is helpful when you want to generate cycle data with more complex structure. Complex shading structure also causes complex cycle data output to be generated and this is useful when you want to get more random data.

Starting cell is where circulating starts. It can be written on any empty cell from grid. Starting cells also causes different cycle data outputs to be generated. Note that, same number cannot be written twice on the same grid.

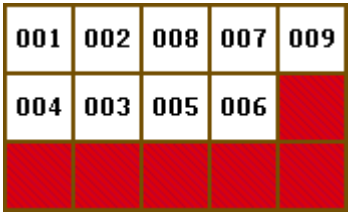
Grid notation:

You can represent grid with text. Notation must have grid size. When possible, it can also have disbanded cells, cycle data or both. Cycle data on notation is written in form of sequence with random numbers which are output of the grid. Actually notation with grid size, starting point and disbanded cells is enough to redesign original grid. Let's examine this grid:



Its notation can be written as : m4x4a3≠8,10,12=3,4,1,2,9,5,6,8,9. The portion **m4x4** is grid size, **a3** indicates starting cell i.e. third cell. Numbers after sign **≠** indicates all disbanded cells. You can also use # or != to indicate disbanding cells because these are normal characters supported on many kinds of document. Numbers after equal sign **=** are cycle data outputs. Notation is usually appropriate for small grid or grid with few disbanded cells. Notation can also be written as question like m4x4a3≠8,10,12=?

Now see also this grid:



For grid like this, its notation can be written as:
m5x3a1≠>9=1,2,8,7,9,4,3,5,6 or
m5x3a1≠>9=?

The portion >9 means that, disbanded cells are all cells from cell number 10, so you don't need to mention all. Grid with no disbanded cell can be written as e.g. m7x5a1≠0=? Or simply m7x5a1=?
If grid is fully with disbanded cells, then notation can be written as: m7x5a0≠m=? or simply as m7x5≠m=? because such grid can never have starting cell. You can also write as m7x5≠>0=?.

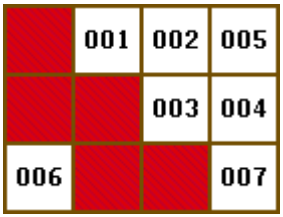
Thing you should note is that, you must shade all required cells first before starting circulation in order to get valid cycle data. Also numbers always start from 1. You cannot use any other number as starting point. And note that, number should be sequential increment by 1, you cannot skip number.

Grid can be of single line like (16x1):

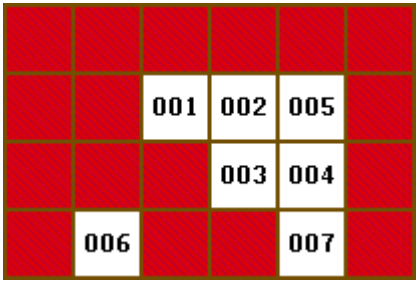
Its notation is m16x1a1≠16.

Cycle data comparison:

Cycle data can be said similar if their outputs are exactly similar. Outputs considered here are collections of all numbers from grid. So grids below are the same because they produce similar collections which are: 1,2,5,3,4,6,7.

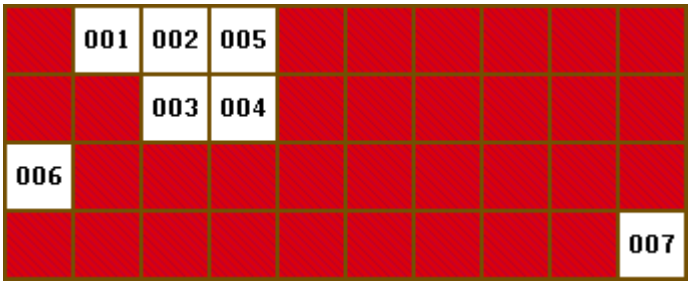


4x3



6x4

Therefore those two grids are equal in outputs but structurally different. You can also say that, left grid is simplified form of right grid.



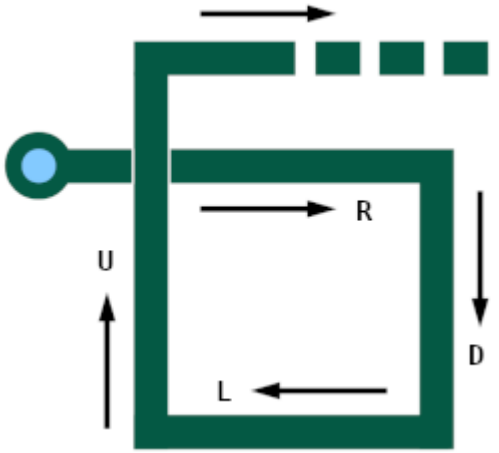
10x4

Even this is similar / equal to all above because its output is like of grids above.

03 Circulation

Directions:
Directions are of two categories:
(a) Direction of deploying numbers on cells.
(b) Direction of selecting lines: columns or rows.

Direction of number deploy:
This is what makes circulation. It is usually shortened as RDLU which means you go to the right, down, left and up. This can also be indicated by number **2301** or by the following illustration:



From diagram we assume the handle or point of intersection as starting point i.e. starting cell. Arrow indicates directions of deploy. If you deploy through all directions then you complete single cycle. Dotted line means it might be continued.

Theoretical number of cycles on grid is equal to total numbers of unshaded cells divided by four.

RDLU direction structure never changes until you fill entire grid. It is also possible to determine next direction with calculation using the last written number **N**. Next direction is equal to the remainder of (**N**+1)/4. For example if last / entered number is 1, then next direction is: (**N**+1)*mod*4 = (1+1)*mod*4 = 2. Now remember RDLU = **2301**. Therefore 2 = Right.

Direction of line selecting:

It is also called direction of line search or grid line serach (GLS). This is of two types, for searching rows (DOSR) or columns (DOSC).

(i) Direction of searching row (DOSR):

This happens only when you are in rightward or leftward deploy, that is when expected cell is not empty on right side or left side of last numbered cell. For example if you want to write on right cell but the cell is not empty you have to proceed on next column from *left to right* until you find empty cell on the same line (row).

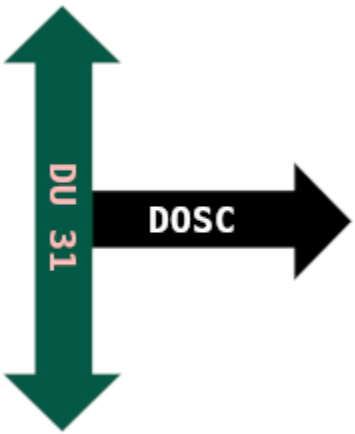
If entire row is empty then you select next row *downward* of the last examined row. If you reach the last bottom line, you restart from top row downward until you find empty cell. So DOSR is always Up-Down.

Now suppose you want to write left cell adjacent to the last numbered cell. If that adjacent cell is not empty then proceed next column i.e. next cell on same row *right to left* until you find empty cell on that row, and again if it happens that entire row has no empty cell, select next row *downward* as you did previously until you find row with empty cell.

(ii) Direction of searching column (DOSC):
This is applicable when in downward or upward deploy. If you want to enter number below the last numbered cell and if such cell is not empty, now instead selecting next column on the same row, just select next row on the same column from *top to bottom* until you find empty cell. If entire column has no empty cell, now DOSC will take place… Select next column *rightward* until you find column with empty cell. If you reach the rightmost column and no empty cell, restart from the leftmost until you get column with empty cell.

Now if you want to enter number up the last numbered cell and cell is not empty, find next row on the same column from *bottom to the top* until you find empty cell. When no success, start DOSC test the same as explained in previous text.

Diagrams below show when to search rows or columns. Green arrows are for deploy directions while black arrows indicate directions when you find row or column with any empty cell.



If it still confuses you can start practising and comparing outputs generated by app. You will therefore be able to fill entire grid yourself. Just enable *Free mode* on those apps (PC and mobile version).

Make sure you practise with small grid like 3x3, 4x4, 5x3 and 5x4 because large grid usually causes confusion.

Also you can contact me using contact details I have provided in this guide document.

04 Using apps

Introduction:
Before developing Cycle data apps, I always practised it myself on grid paper. Finally I started developing cross-platform apps to do task more quickly, as of this time, there are two Java apps released for free download. The first app is J2ME mobile app and another is JavaSE-based (to be used on PC). These apps also support Free mode that can allow you to deploy numbers yourself if you want to examine your skills. As of v2.00+ game feature included so you can play as game or you can enable game autoplay. Just click on Option knob from app interface.

J2ME mobile app:
It is very small app that works on any feature phone with J2ME support, that has display of at least 240x320. The grid size for this app is limited up to 8x8. So you cannot use for example 9x8. For Android users this app works well. Just install J2ME Loader and set display size of 240x320. You can also tick for filtering for smooth look. J2ME Loader is very powerful emulator for J2ME app to run on Android devices and you can get it from PlayStore.

It is supported on both touch phones and phones with keypad therefore if you run it with J2ME Loader you have to disable virtual keyboard because touch input is enough to use app.

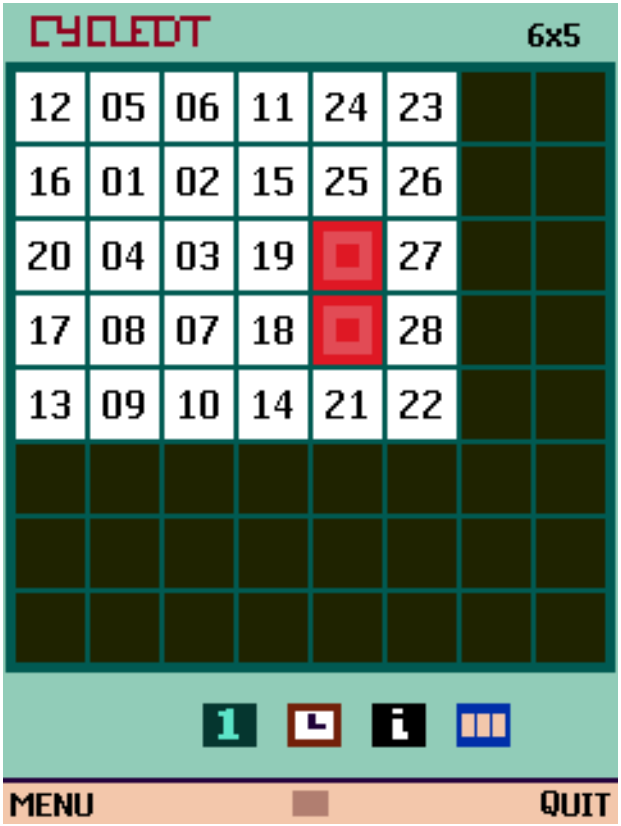
This is image of J2ME app for Cycle data:

Image of J2ME app for Cycle data. Red cells are disbanded cells in 6x5 grid.

Options for specifying grid, circulating etc can be found from application menu. You can toggle modes using mode knob on interface or by pressing key 1. Clock knob is for displaying device time.

You can also press 3 to achieve this.

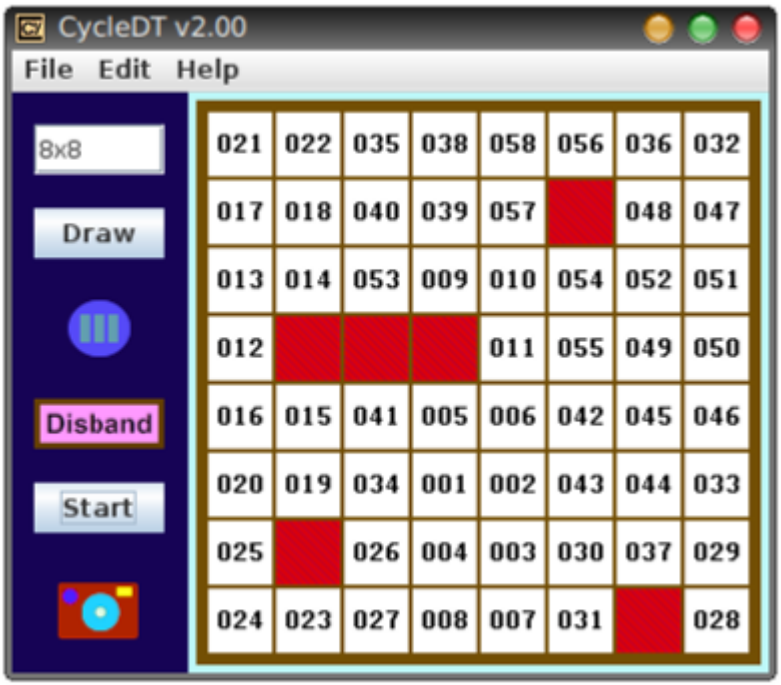
Platform required is JavaME MIDP 2.0



J2ME version also support gaming feature and it is the first app however to have grid game before PC version. But still game it has some limitation, it does not have game autoplay, but you can solve it from Main pop-up menu (Circulate).

Desktop version:

This is also written in Java programming language, so it can run on any platform with JavaSE support e.g. Linux, Windows and Mac OS. If your PC has no Java (JRE), download it from java.com. JRE is runtime environment to enable any program written in Java to be executed directly on any OS by using Java Virtual Machine (JVM) included on JRE. Next image is how PC version looks like:



PC version as you can see contains more options. Camera icon enables you to take screenshot of grid into PNG image. Image can be saved on hard disk or memory card for reference.

Desktop version is more enhanced as it can deploy numbers up to 961 that is equal to the maximum grid size of 31x31. App is also flexible according to the grid size. Common knobs for functionalities can be found on app interface but through menus you can find more. For example, in desktop app you can locate certain number on grid from *Edit / Find* menu. From *Edit / Details* you can get information about current grid. Version 2.00+ includes Option knob for more features. There you can choose grid game options. If you choose *Play grid* then the currently drawn grid will be used as game so that you can play it. But you can also choose random games based on current grid size.

Also you can set game so that it can play automatically for you until it fills entire grid.
If you are sure game still run smoothly, you can turn app sound on so that action sound feedback will be heard as you play game or you fill it with Free mode.

Actually apps themselves contain enough information about how to use them, so this document is not for tutorial about how to use those apps rather it is highlight about Cycle data.

05 Conclusion

What you should know …

You have to know that grid size for Cycle data has no maximum but too large grid can be impractical to human being that is why apps I have developed to deal with Cycle data have maximum grid size. Also too large grid can cause memory issues and display could not handle such size.

Terminologies:

These are few words used in this Cycle data scheme, we should read them to be clear about how they are used in this context.

Circulation – Also called deploy, the process of filling grid with numbers through cycles to form Cycle data.
Cycle – Single cycle is formed by all deploy directions: right, down, left and up.
Cycle data – Output formed by cycles of numbers on grid with all appropriate rules.
Deploy (*see Circulation*).
Disbanding – The verb in this context is used transitively for process of marking cell as inapplicable. The verb *shading* or *discarding* can also be used.
DOSC – Direction of searching columns, when you find any column with empty cell.

DOSR – Direction of searching row, when you find any row with empty cell.
GLS – Grid line search, when searching either row (DOSR) or column (DOSC). It is used as generic term for both DOSR and DOSC.
RDLU – Or *2301*, is a single cycle formed by directions right, down, left and up during deploying.
Starting cell – Also called starting position or starting point is a cell with number 1 where the first cycle starts.

Contacts:

You may contact me with given contact details below:
Email: ye2hh@mail.com
Home page: Click here.

© YPH – (2019-2020) [Yeye!]
All rights reserved.

2020