335 — Algorithm Engineering — Cella Ant #x15

Project #1 – Cella Ant #x15

Introduction

This project is to write a program to display the generational progress of Turk & Propp's (TP) Ant #x15, a cellular automaton (Cella) variant of Langton's Ant. (Read about this at the Wikipedia page for Langton's Ant, in the section on Extension to Multiple Colors). The program will be written in P5.js+Javascript with an HTML web page for display (as described in lecture).

The cella will be shown in a 2D grid of black, red, yellow, blue, and green cells; initially all are black.

TP Ants

Turk & Propp's Ants (TP Ants) crawl on a 2D grid where each cell has a color (initially all are black). As an Ant crawls, it first notices the color of the cell it is on and then changes its direction/heading based on this color, then "increments" the color of the cell under it to the next color in sequence (wrapping around if needed), and finally moves one cell in its new direction (wrapping around on the grid if needed). Each color is associated with a change of direction: to the Left or to the Right of the Ant's current heading, based on the Ant's number and the index order of that color in the color sequence.

Initially, the grid is all black and the Ant is in the center cell, heading North (toward the top).

TP Ant #x15

For Ant #x15, the Ant's number is 15 hexadecimal = 10101 binary (= 21 decimal) ==> cell turning states of LRLRL, where a binary 1 bit means turn Left, and a 0 means turn Right. Each cell has 5 states, one for each binary digit position in the 5 bits, 11100. Each state is represented on the grid by a color (for easy visuals), with the color/state sequence (from low bit to high bit) being Black=0 → Red=1 → Yellow=2 → Blue=3 → Green=4 → Black=0 (wrapped), et cetera. Hence, Ant #x15 turns Right on Red or Blue, and else Left. Also, treat the Ant as a (reasonably visible) white triangle which sits on top of the cell color and points toward the cell edge it is heading toward.

Setup

Your program should initialize a 41 by 41 square grid to have all cells black (state 0). The lone Ant should be in the center cell. (The cells should be of reasonably visible size: eg, 10x10 or 20x20 pixels.)

Running

After setup, your program should run at least for 1,000 moves, showing the grid changes after each move.

Complexity Order

You should prepare a 1-page paper describing your analysis of the Big-O running time of your algorithm. Address the usual issues such as main operations, input size, etc.

Team

Your team may contain up to four members. We would prefer 3-4 sized teams. Pick a short-ish (<= 8 letter) name for your team (e.g., "Groggy").

Project Development Reporting

Standup Status Report, twice weekly. The Standup Status Report is due **Monday's** and **Friday's** by noon-ish. One report per team, **CC'ing the other team members**. It should contain, team name and the member names. This documents should be delivered **as a PDF file**; and the **filename** should be in the following format: include your course and section number, project number, your team name, the document type (Standup), and the date as YYMMDD:. E.g., "`335-02-p1-Groggy-Standup-200831.pdf`".

Standup Status Report contents should be, for each team member, a list of the 3 **Standup question short answers**: Q1: what have you **completed** (name sub-tasks) by the time of this Standup report; Q2: what do you **plan to complete** (name sub-tasks) by the time of the next Standup report; and Q3: what obstacles if any (1-line description) are **currently blocking you** (for which you've reasonably tried to find the answers by yourself, including asking your team about them – known as "due diligence"). Note, that you can email the professor, or ask questions during office hours to get answers.

Readme File

When your project is complete, you should provide a README.txt text file. Be clear in your instruction on how to build and use the project by providing instructions a novice programmer would understand. If there are any external dependencies for building, the README must also list them and how to find and incorporate them. Usage should include an example invocation. A README would cover the following:

- Class number
- Project number and name
- Team name and members
- Intro (including the algorithm used)
- Contents: Files in the .zip submission
- External Requirements (None?)
- Setup and Installation (if any)
- Sample invocation
- Features (both included and missing)
- Bugs (if any)

Academic Rules

Correctly and properly attribute all third party material and references, lest points be taken off.

Submission

All Necessary Files: Your submission must, at a minimum, include a plain ASCII text file called **`README.txt`**, all project documentation files (except those already delivered), all necessary source files to allow the submission to be built and run independently by the instructor. [For this project, no unusual files are expected.] Note, the instructor not use use your IDE or O.S.

Headers: All source code files must include a comment header identifying the author, author's contact info (please, no phone numbers), and a brief description of the file.

No Binaries: Do not include any IDE-specific files, object files, binary **executables**, or other superfluous files.

Project Folder: Place your submission files in a **folder named** like your Standup report files: `335-02-p1-Groggy`.

Project Zip File: Then zip up this folder. Name the .zip file the **same as the folder name**, like `335-02-p1-Groggy.zip`. Turn in by 11pm on the due date (as specified in the bulletin-board post) by **submitted via emailed zip file(s) (preferred)**, or via accessible cloud (eg, Github, Gdrive, Dropbox) with emailing the accessible cloud link/URL. See the Syllabus for the correct email address. The email subject title should include **the folder name**, like `335-02-p1-Groggy`. Note that some email-programs block .ZIP files (but maybe allow you to change .ZIP to .ZAP) and block sending zipped files with .JS files inside (but maybe allow you to change .JS to .JS.TXT).

Email Body: Please include your team members' names and campus IDs at the end of the email.

Project Problems: If there is a problem with your project, don't put it in the email body – put it in the README.txt file, under an "ISSUES" section.

Grading

- 75% for compiling and executing with no errors or warnings
- 10% for clean and well-documented code (Rule #5(Clean))
- 10% for a clean and reasonable documentation files
- 5% for successfully following Submission rules