Algorithms - Project 2

This project is to write a program that implements the union-find algorithm and use it to estimate the value of "percolation threshold" for a square lattice.

The algorithm can be readily used to generate square mazes:

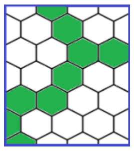
Problem: If sites are independently set to be opened with probability p, what is the probability the maze percolates (we can cross the maze from one side to another)? What is the threshold p* such that when p>p*, we can almost certainly to do so and when p<p*, the chance of success is small?

Approach: 1) performing a computer simulation to estimate the threshold. You should run your simulation for *n* runs for every p value you choose. For each run, based on the value of p, randomly pick sites to be open; 2) using Union-find to examine whether there is a path from one side to another for each run; 3) Estimated p*.

To facilities our grading of your project, your program should allow the user to specify the number of runs for the simulation and size of the game board (assume width = length) through common line arguments. We'll use the following to grade your program:

```
yourProgram.exe p n s // p - percolation probability; n - # of runs; s – board size;
                              // your program should output percolation rate
yourProgram.exe fname.txt // the game board setting is saved in the file named fname.txt
                    //click here for a sample input. 1 stands for open site; 0 for closed site.
                              // your program should output the number of clusters on the board
```

Bonus (5% each):



1) Color-code the clusters on the board and save the board as an image. Use PPM format with ascii encoding.
   (ppm files can be viewed using IrfanView http://www.irfanview.com/)
2) Make your implementation working for Honeycomb lattice.

Bonus (1pt for undergraduate, required for graduate students):
A report on the analysis of the Find operation with/without path compression.

More description on how union-find is used for percolation can be found at:
https://www.cs.princeton.edu/courses/archive/fall10/cos226/assignments/percolation.html

What to submit.

1. Submit your source code. Make sure to test your code on more than one set of data. DO NOT submit programs that are not reasonably correct! To be considered reasonably correct, a program must be completely documented and work correctly for sample data provided with the assignment.
2. A text file named readme.txt file to give the reader a high-level explanation of your implementation and anything we need to know to run your program..
3. (**Graduate student**) A report of the analysis of the Union-Find algorithm you implemented in this project. You should present your asymptotic analysis on the space requirement and the running time of the algorithms used in this program.

Source code submission instructions. When you are ready to submit, obtain a printed copy of your program. Remember to sign and attach the required Academic Integrity Pledge cover sheet. The printed program and cover sheet must be turned in to your instructor by the start of class on the date the program is due. You must submit an electronic copy of the program using your Springboard dropbox. Follow these steps:

1. Create a folder named jsmith_1 (but use your first initial and last name).
2. Place just the source files inside the folder.
3. Right-click on the folder and choose Send To... Compressed Folder (or use some other Zip utility to archive the entire folder). The goal is to create a zip archive named jsmith_1.zip (your initial and name) that contains the folder which contains the source files for your project.
4. Drop this single zipped file to the drop box.

Please pay attention to the naming conventions for the submission files. These must be followed exactly in order to receive credit. Invalid submissions will need to be resubmitted with a penalty assessed.

Be sure to electronically submit your working solution before the due date! Do not submit non-working programs. The electronic submission time will be used to assess late penalties (if applicable).

Grading. Your code will be graded on correctness, efficiency, clarity and elegance. What you claim in your writeup (readme.txt, report) will be checked by running your code. For **graduate students**, your report has to be written clearly and professionally. Use figures and tables wherever you can to justify your conclusions. Your report should follow the format and suggestions given the document at: http://www.cs.york.ac.uk/projects/howtowrt.html .