COSC201 Assignment 1: Counting the seas
Sample report

April 28, 2022

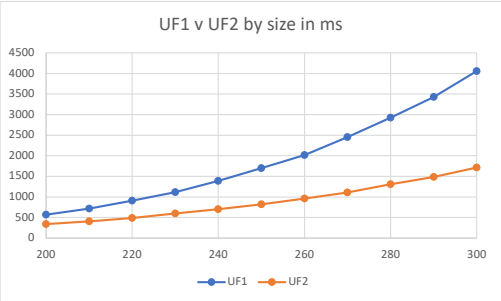
Introduction

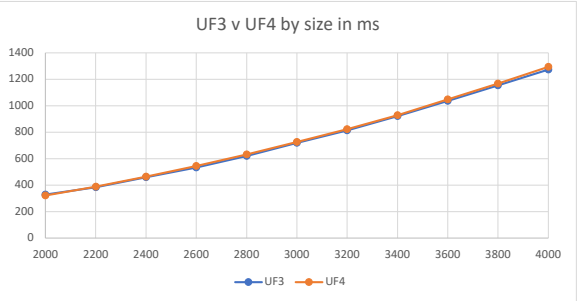
This report concerns various experiments carried out in imaginary worlds consisting of grids of cells each of which is water or land. This framework was mainly used to investigate the efficiency of various union-find implementations in counting the number of seas in such a world.

Question 1

The efficiency of the various union-find instances affects how large a square map can be analysed. A reasonable length of time for a single computation might be something on the order of a second. . . . Comment briefly on whether the outcomes match our theoretical discussions of the four different algorithms and how changing the water probability might affect their performance.

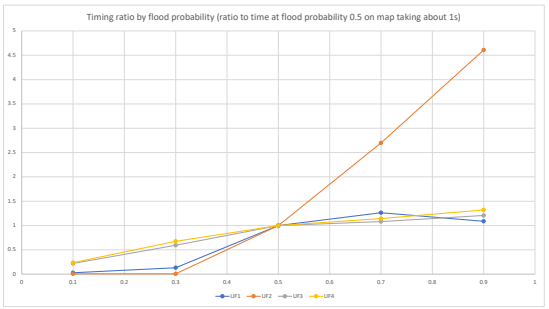
Initial investigations on single trials suggested that `UF1` and `UF2` would only be able to analyse maps up to about 300×300 in about a second, while `UF3` and `UF4` could analyse much larger maps. So, repeated trials were carried out with the two groups of union-find algorithms at various map sizes. The data were found to be quite consistent so 10 trials at each size were carried out. The mean times over a range of sizes can be seen in the graphs below.





These show that (on my hardware) UF1 could handle maps up to 230×230, UF2 to 260×260, while UF3 and UF4 could handle maps up to 3500×3500. This matches, at least qualitatively, our theoretical discussion which gave $O(n^2)$ time estimates for the first two UF instances (where n would correspond to the total number of cells in the map) and $O(n \log n)$ estimates for the latter pair.

It seems natural to expect that, as the water probability increases, the time required for each type of UF implementation should increase since more union operations will be needed. To test this, each UF type was timed on a map where it would take roughly 1 second to handle the 50% water case but with water probabilities ranging from 0.1 to 0.9 in steps of 0.2. The times taken were then normalised so all the values at 0.5 corresponded to 1. These results are shown below.



The results more or less confirm our initial assumption, though it is notable that the UF2 implementation is much more strongly affected by the flooding probability than the others are. Also of note is the drop for UF1 from 0.7 to 0.9. This was confirmed by more extensive investigations, though the reason remains a mystery.

Question 2

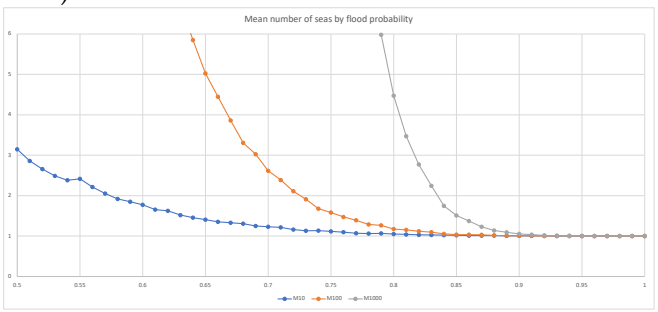
Consider 10×10, 100×100, *and* 1000×1000 *maps. Conduct experiments to try and determine the proportion of water needed on each so that on average there are two or fewer seas.*

> **Aside:** Two answers are given below, just because of the two different interpretations of "on average". Either one alone would be sufficient.

In this experiment a single UF type was used as the question is about the structure of the map, which is not affected by the underlying UF implementation. UF3 was used as it seemed to be the fastest in our first experiments.

Version 1

The experiment was quite straightforward. At each water probability from 0.5 to 1.0 in steps of 0.01 I produced 1000 maps of each size, flooded by that probability and computed the mean number of seas in each case. Those results are presented in the graph below (where the vertical axis has been trimmed so that the transition to two or fewer seas can be seen).

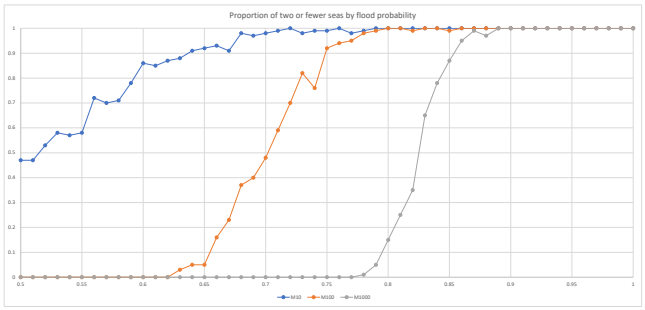


This suggests that the proportions of water needed in the three cases are roughly 0.57 in the 10×10, 0.73 in the 100×100 case and 0.84 in the 1000×1000 case. The fact that these increase is not surprising as, in larger maps, there is more space for isolated seas to form, and so presumably a larger proportion of water is required to join almost all of them together.

This experiment took a couple of hours to run (almost entirely to handle the 1000×1000 cases), but I occupied that time by writing this report.

Version 2

I chose to interpret "on average" as meaning "at least 50% of the time there were two or fewer seas". So, at each water probability from 0.5 to 1.0 in steps of 0.01 I produced 100 maps of each size, flooded by that probability and computed the proportion of maps in each case that had two or fewer seas.



The results are a little rough, and could be smoothed by doing more trials, but the trends are clear enough. They suggest that the proportion of water needed in the 10×10 case to get two or fewer seas at least 50% of the time is roughly 0.52 while it's 0.70 in the 100×100 case and 0.83 in the 1000×1000 case. The fact that these increase is not surprising as, in larger maps, there is more space for isolated seas to form, and so presumably a larger proportion of water is required to join almost all of them together.

It is notable that the steepness of the transition from rarely having two or fewer seas to almost always having two or fewer seas increases markedly as the map size increases. To some extent, this is unavoidable as at probability 1 we definitely have only one sea – but it does seem to be even sharper than that would require.

Question 3

Suppose that we also wanted to count the islands. How would you do that - and what changes to the code might be needed?

The main issue in counting islands is to decide on what counts as adjacency. In a 2×2 map with diagonally adjacent land and water cells, the two water cells are considered to be part of the same sea, and so presumably the two land cells should be separate islands.

The basic process of counting islands is the same – loop over all the land cells in a union-find implementation, taking the union of any pair that are adjacent and then count the number of groups. But, as noted above, the code for adjacency would need to be changed so that for this case we were counting as adjacent only cells that share an edge, not cells that simply touch at a corner.