1 Erdos Number

(C++11) For this question, we used adjacency lists to track whether authors had shared papers. We implemented this as an unordered set, which we kept track of with an unordered map. This unordered map also stored Erdos numbers for each author, which we initialized to 0.

Once we had processed all the input and inserted it into adjacency lists, we ran a BFS starting at the adjacency list describing Erdos' co-authors. As each one of the coauthors were visited, they were assigned a number of 1 and added to a queue. If a coauthor had not yet been visited, then they would have an Erdos number 1 greater than the neighbour from which they were referred.

After this processing, we could answer Erdos number queries - if an author had an Erdos number of 0, this would be because they were never visited by the Erdos BFS, at which point we could say their Erdos number was infinity.

2 Poker Hand

(C++ 11) For this question, we stored the 5 cards as a 64-bit integer. The 8 most-significant bits of this integer were used to denote the different hands the cards could be scored by (e.g. straights, flushes, pairs, etc.). We modified this integer via masks and bitwise operations.

During input, we checked to see if all the input cards had the same suit. If so, we marked the hand as having a flush. Aside from this, suits do not matter to the problem, as it is stated that no suit is ranked higher than another. Having done so, we now only consider the denominations. The cards' denominations are stored in a vector with 13 entries, where each entry $v[i]$ determines the number of cards with denomination i

First, we check for straights using our v vector by seeing if there are 5 consecutive entries in v with values of 1.

Then, we simply load every value into the v vector with bitshifts. Starting from the least significant bit, every four bits describes how many cards have that particular denomination.

After this, we can compare the two numbers.

3 Flea Circus

Heather writeup+comment

4 Check the Check