MiniProject 3 (Time-Space optimization)

Background

You're fed up with starving on your student loans and decide to build a bot that plays a perfect game of poker. The bot is based on a neural network, which is trained by playing the bot against itself for billions of hands ("reinforcement learning", cf. AlphaGo and AlphaZero).
For the bot, you need a powerful algorithm that can determine which poker hand (straight, flush, pair, etc.) when you know the 5 cards of the hand. The algorithm must work absolutely correctly, so check out carefully about the different hand types (https://en.wikipedia.org/wiki/List_of_poker_hands) and how many ways each hand type can be formed by 5 cards (https://en.wikipedia.org/wiki/Poker_probability).

The project

The aim is to implement an algorithm to identify a pair of different types of hands. The algorithm is first implemented "normally", and then more efficiently using lookup tables.

Part 1 Detection of poker straight (1 point)

There are 52 playing cards in the deck. The cards are of four different suits (hearts, diamonds, clubs, spades) and 13 different numerical values (2-14). A single card can therefore be represented by an integer from 0 to 51. Based on the integer, it can be easily calculated for both the suit and the numerical value of the card in question.

A poker hand consists of five cards taken from a deck. One type of hand is a "straight" hand, where the hand contains cards of five consecutive numerical values, e.g. 5, 8, 7, 6, 4 (the order of the cards is irrelevant). If all the cards also happen to be of the same suit, it is a "straight flush". Note that the ace can be either 14 or 1.

See the test program straight.c, which runs through all 5 card permutations. The test program calls for each permutation the function is_straight, which takes as a parameter a 5-element table containing the corresponding card. The function returns the value 1 if the numeric values can be arranged in a straight line, 0 otherwise.

Implement the is_straight function. One option is to first arrange the cards in order of magnitude, which then it is easy to test whether they form a straight line. The ace (1 or 14) should be tested as a special case.

Note down the execution time used by the program.

Part 2 Detection of poker straight using lookup table (2 point)

Improve your direct identification with the search bar. Use the lookup table to quickly check whether the the given cards are a straight.

Tip for implementation:

The numerical values of all the cards in the hand are represented as a vector of bits. The bit is on if the given number value belongs to the hand.

bitti	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
kortti				A	K	Q	J	T	9	8	7	6	5	4	3	2

So first create an empty bit vector with all bits set to 0. Then run the numerical values of the five cards in the hand and set the bit corresponding to that numeric value to 1. Finally, use the bit vector to index the lookup table. The lookup table is pre-stored with a value of 1 if the hand corresponding to that index is a straight hand, 0 otherwise.

To learn more about bit-level operations, see for example: https://en.wikipedia.org/wiki/Bitwise_operations_in_C.

Run the optimized program, and compare the execution time with the original version. Leave the lookup table time spent on initialization out (of consideration).

Also try to find out how much more memory the optimized program uses. What part of memory (stack, heap, code, etc.) consumption increased?

Part 3 Reference counting (2 points)

The "four of a kind" or “quads” have 4 of the same numerical value (the fifth card can be any card). The "full house" has 3 same and 2 of the same numerical value.

Implement the function is_quadsorfullhouse, in the same style as for straight identification. So the function identifies, whether your poker hand is either a four of a kind or a full house (so the same function identifies whether it is one or the other).

Make both a regular and a lookup table version of the function. Make sure that exactly the right number of fours/full hands are identified