# Data structures used for chess

A chess program must be able to:

remember what the position is;

generate moves, given a position;

verify moves, to make sure the move to be made is legal;

make clever moves;

## A move

Definitely, a structure describing a chess move should contain the source and

destination square. Information contained in the *move* structure must be enough for the

program to be able to take that move back. Therefore, it should also describe the captured

piece (if any). If the move is a promotion, it must contain the piece that the pawn is

promoted to. Sometimes (it depends on a type of a chessboard representation), also type of

the piece that is moving should be included.

The above data may be put together into a structure having several one byte fields

(e.g.: *source, destination, captured, promoted, moved*). However, the efficiency of the

program often increases if the structure is contained within one 32-bit integer (for example,

6 bits per source / destination field, 4 bits per captured / promoted / moved piece) as it

reduces the memory access overhead for transferring/accessing particular fields of

instances of this often used structure (assuming that arithmetic and logical operations are

performed relatively much faster than memory access – in most hardware environments,

including the most popular PC platform, it is exactly the case).

### A chess position

There are several methods of representing a chess position in the computer

memory. The most natural way seems to be an array 8x8, with each element corresponding

to one of chessboard squares. Value of the elements would indicate state of the square

(empty, occupied by black pawn, occupied by white queen etc.). If one byte was used to

represent one square, then the whole data structure would occupy 64 bytes of computer

memory (I will ignore everything but piece location for now), that is not much. Therefore,

this approach was widely used in the early stages of computer chess development. For

code optimizing reasons, the two dimensional array 8x8 was often replaced with a single

one dimensional array of 64 elements.

Later, the idea was improved by adding two square sentinels at the edges. Sentinel

squares were marked *illegal*. It speeded up move generation, as no check had to be done

each time a move was to be generated to verify if edge of the board was not reached. The

drawback was increase of the structure size to 144 bytes.

