- **Take the first half of this word and combine it with the second half of the other word**

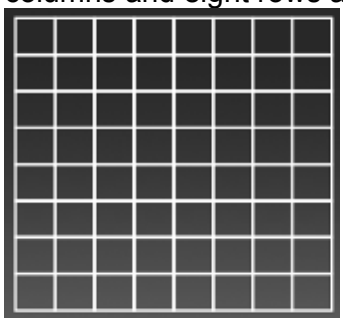
Notice that we are generating new individuals from the best ones by using crossover. The simplest way to perform this crossover is to combine the head of one individual to the tail of the other, as shown in the diagram below.

In the 32-bit word problem, the (two-parent, no mutation) approach, if it succeeds, is likely to succeed much faster because up to half of the bits change each time, not just one bit. However, with no mutation, it may not succeed at all. By pure bad luck, maybe none of the first (randomly generated) words have (say) bit 17 set to 1. Then there is no way a 1 could ever occur in this position. Another problem is lack of genetic diversity. Maybe some of the first generation did have bit 17 set to 1, but none of them were selected for the second generation. The best technique in general turns out to be a combination of both, i.e., crossover with mutation.

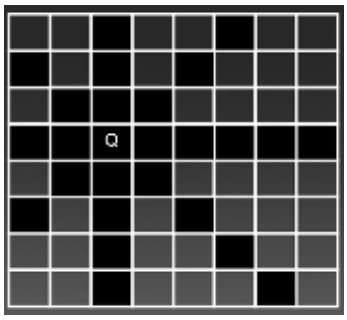
3.7 Eight Queens Problem

Let us now solve a famous problem which will be discussed under GA in many famous books in AI. Its called the Eight Queens Problem.

The problem is to place 8 queens on a chess board so that none of them can attack the other. A chess board can be considered a plain board with eight columns and eight rows as shown below.

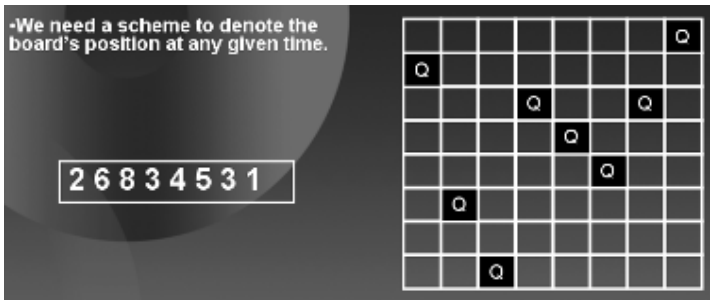


The possible cells that the Queen can move to when placed in a particular square are shown (in black shading)



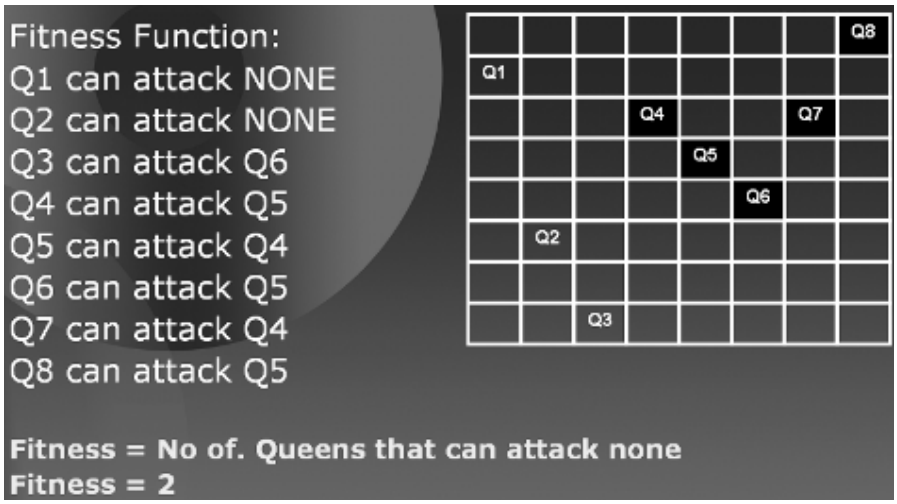
We now have to come up with a representation of an individual/ candidate solution representing the board configuration which can be used as individuals in the GA.

We will use the representation as shown in the figure below.



Where the 8 digits for eight columns specify the index of the row where the queen is placed. For example, the sequence 2 6 8 3 4 5 3 1 tells us that in first column the queen is placed in the second row, in the second column the queen is in the 6^{th} row so on till in the 8^{th} column the queen is in the 1^{st} row.

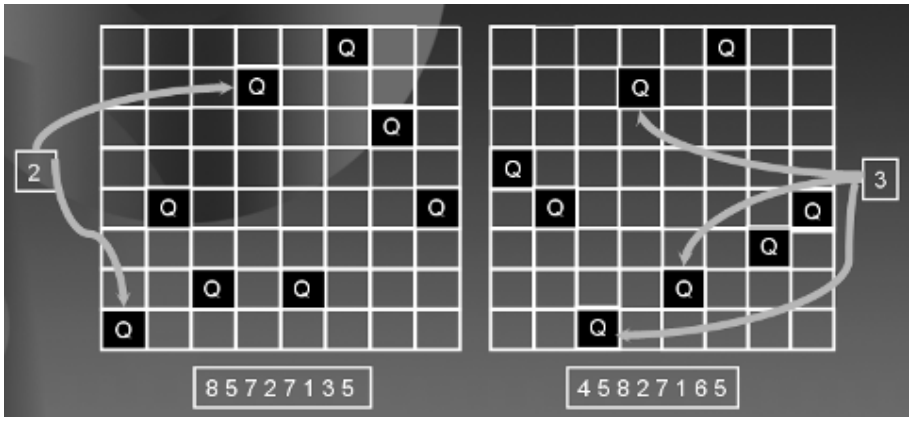
Now we need a fitness function, a function by which we can tell which board position is nearer to our goal. Since we are going to select best individuals at every step, we need to define a method to rate these board positions or individuals. One fitness function can be to count the number of pairs of Queens that are not attacking each other. An example of how to compute the fitness of a board configuration is given in the diagram on the next page.



So once representation and fitness function is decided, the solution to the problem is simple.

- **Choose initial population**
- **Evaluate the fitness of each individual**
- **Choose the best individuals from the population for crossover**

Let us quickly go though an example of how to solve this problem using GA. Suppose individuals (board positions) chosen for crossover are:

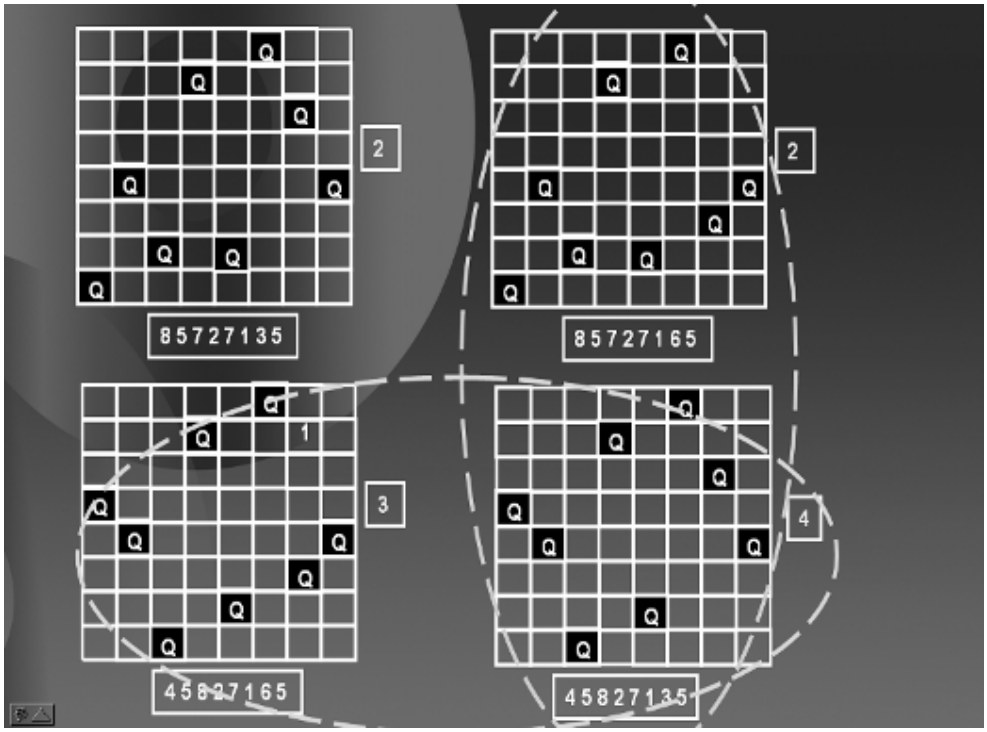


Where the numbers 2 and 3 in the boxes to the left and right show the fitness of each board configuration and green arrows denote the queens that can attack none.

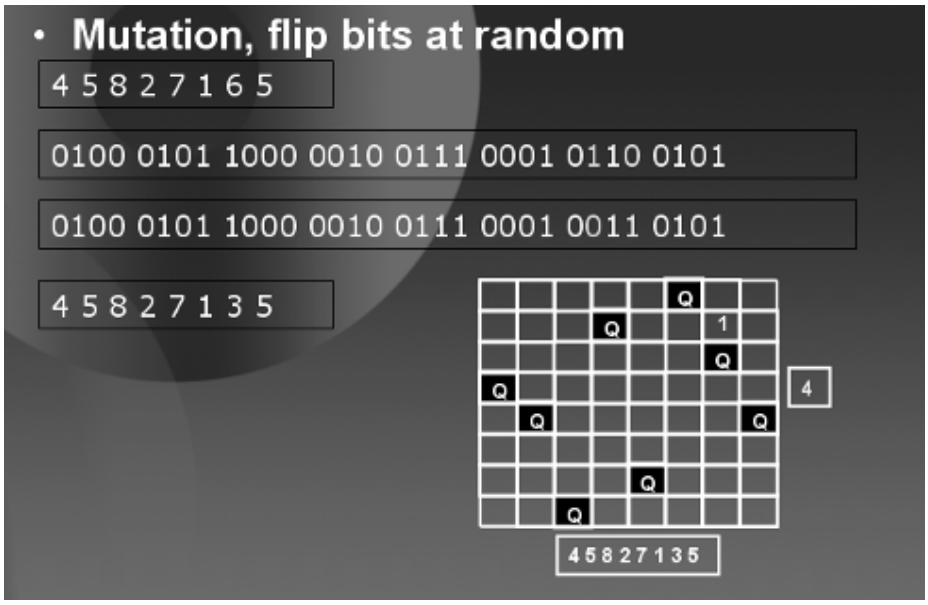
The following diagram shows how we apply crossover:

The individuals in the initial population are shown on the left and the children generated by swapping their tails are shown on the right. Hence we now have a total of 4 candidate solutions. Depending on their fitness we will select the best two.

The diagram below shows where we select the best two on the bases of their fitness. The vertical over shows the children and the horizontal oval shows the selected individuals which are the fittest ones according to the fitness function.



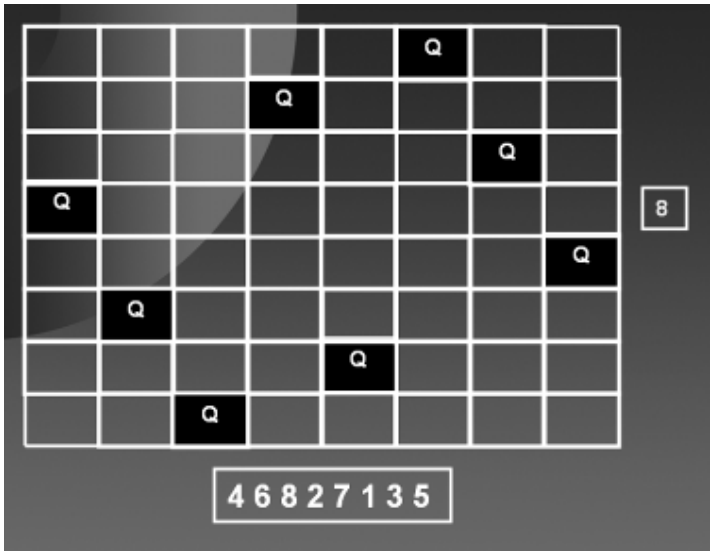
Similarly, the mutation step can be done as under.



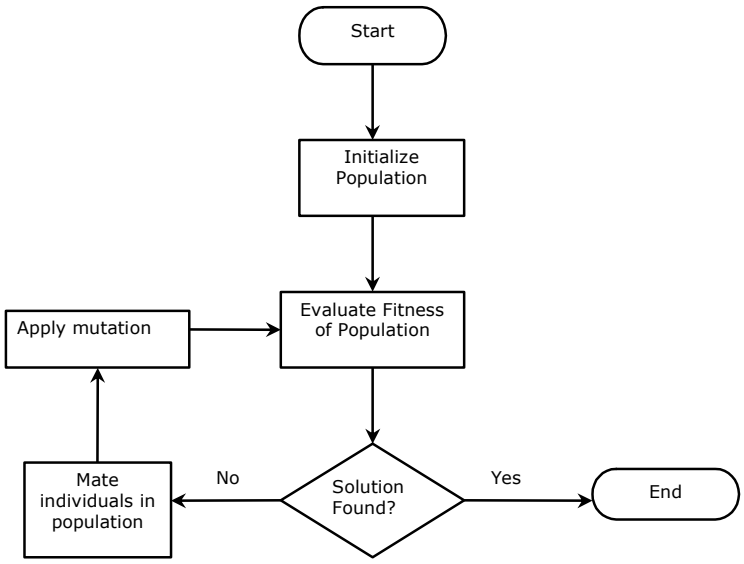
That is, we represent the individual in binary and we flip at random a certain number of bits. You might as well decide to flip 1, 2, 3 or k number of bits, at random position. Hence GA is totally a random technique.

This process is repeated until an individual with required fitness level is found. If no such individual is found, then the process is repeated till the overall fitness of the population or any of its individuals gets very close to the required fitness level. An upper limit on the number of iterations is usually used to end the process in finite time.

One of the solutions to the problem is shown as under whose fitness value is 8.



The following flow chart summarizes the Genetic Algorithm.



You are encouraged to explore the internet and other books to find more applications of GA in various fields like:

- Genetic Programming
- Evolvable Systems
- Composing Music
- Gaming
- Market Strategies
- Robotics
- Industrial Optimization

and many more.