In [1]:
```
import numpy as np
```

In [33]:
```
class HopfieldNetwork:
    def __init__(self, size):
        self.size = size
        self.weights = np.zeros((size, size))

    def train(self, patterns):
        num_patterns = len(patterns)
        for pattern in patterns:
            pattern = np.reshape(pattern, (self.size, 1))
            self.weights += np.dot(pattern, pattern.T)
        np.fill_diagonal(self.weights, 0)

    def predict(self, pattern, max_iter=100):
        pattern = np.reshape(pattern, (self.size, 1))
        for _ in range(max_iter):
            old_pattern = pattern.copy()
            pattern = np.sign(np.dot(self.weights, pattern))
            if np.array_equal(pattern, old_pattern):
                return pattern.flatten()
        return("Prediction did not converge within max_iter iterations.")
```

In [34]:
```
# Example usage:
patterns = [
    [1, -1, 1, -1],
    [-1, -1, -1, 1],
    [1, 1, -1, -1]
]
```

In [35]:
```
hopfield_net = HopfieldNetwork(size=len(patterns[0]))
hopfield_net.train(patterns)
```

In [36]:
```
# Predicting from a noisy pattern
noisy_pattern = [1, -1, 1, 1]
retrieved_pattern = hopfield_net.predict(noisy_pattern)
print("Original Pattern:", noisy_pattern)
print("Retrieved Pattern:", retrieved_pattern)
```

```
Original Pattern: [1, -1, 1, 1]
Retrieved Pattern: Prediction did not converge within max_iter iterations.
```