ALGORITHM 5.6 Knuth-Morris-Pratt substring search

```
public class KMP
{
   private String pat;
   private int[][] dfa;

   public KMP(String pat)
   {  // Build DFA from pattern.
      this.pat = pat;
      int M = pat.length();
      int R = 256;
      dfa = new int[R][M];
      dfa[pat.charAt(0)][0] = 1;
      for (int X = 0, j = 1; j < M; j++)
      {  // Compute dfa[][j].
         for (int c = 0; c < R; c++)
            dfa[c][j] = dfa[c][X];            // Copy mismatch cases.
         dfa[pat.charAt(j)][j] = j+1;         // Set match case.
         X = dfa[pat.charAt(j)][X];           // Update restart state.
      }
   }

   public int search(String txt)
   {  // Simulate operation of DFA on txt.
      int i, j, N = txt.length(), M = pat.length();
      for (i = 0, j = 0; i < N && j < M; i++)
         j = dfa[txt.charAt(i)][j];
      if (j == M) return i - M;  // found (hit end of pattern)
      else        return N;      // not found (hit end of text)
   }

   public static void main(String[] args)
   // See page 769.
}
```

The constructor in this implementation of the Knuth-Morris-Pratt algorithm for substring search builds a DFA from a pattern string, to support a `search()` method that can find the pattern in a given text string. This program does the same job as the brute-force method, but it runs faster for patterns that are self-repetitive.

```
% java KMP AACAA AABRAACADABRAACAADABRA
text:    AABRAACADABRAACAADABRA
pattern:             AACAA
```