KMPplus.java

Below is the syntax highlighted version of KMPplus.java from §5.3 Substring Search.

```
/*************************************************************************
 *  Compilation:  javac KMPplus.java
 *  Execution:    java KMPplus pattern text
 *  Dependencies: StdOut.java
 *
 *  Knuth-Morris-Pratt algorithm over UNICODE alphabet.
 *
 *  % java KMPplus ABABAC BCBAABACAABABACAA
 *  text:    BCBAABACAABABACAA
 *  pattern:          ABABAC
 *
 *  % java KMPplus aabaaaba ccaabaabaabaaabaab
 *  text:    ccaabaabaabaaabaab
 *  pattern:         aabaaaba
 *
 *  % java KMPplus aabaaabb ccaabaabaabaaabaab
 *  text:    ccaabaabaabaaabaab
 *  pattern:                  aabaaabb
 *
 *************************************************************************/

public class KMPplus {
    private String pattern;
    private int[] next;

    // create Knuth-Morris-Pratt NFA from pattern
    public KMPplus(String pattern) {
        this.pattern = pattern;
        int M = pattern.length();
        next = new int[M];
        int j = -1;
        for (int i = 0; i < M; i++) {
            if (i == 0)                                 next[i] = -1;
            else if (pattern.charAt(i) != pattern.charAt(j)) next[i] = j;
            else                                        next[i] = next[j];
            while (j >= 0 && pattern.charAt(i) != pattern.charAt(j)) {
                j = next[j];
            }
            j++;
        }

        for (int i = 0; i < M; i++)
            StdOut.println("next[" + i + "] = " + next[i]);
    }

    // return offset of first occurrence of text in pattern (or N if no match)
    // simulate the NFA to find match
    public int search(String text) {
        int M = pattern.length();
        int N = text.length();
        int i, j;
        for (i = 0, j = 0; i < N && j < M; i++) {
            while (j >= 0 && text.charAt(i) != pattern.charAt(j))
                j = next[j];
            j++;
```

```
        }
        if (j == M) return i - M;
        return N;
    }


    // test client
    public static void main(String[] args) {
        String pattern = args[0];
        String text    = args[1];
        int M = pattern.length();
        int N = text.length();

        // substring search
        KMPplus kmp = new KMPplus(pattern);
        int offset = kmp.search(text);

        // print results
        StdOut.println("M = " + M + ", N = " + N);
        StdOut.println("text:    " + text);
        StdOut.print("pattern: ");
        for (int i = 0; i < offset; i++)
            StdOut.print(" ");
        StdOut.println(pattern);
    }

}
```


Last updated: Wed Aug 26 05:30:12 EDT 2015.