FASTA Files Honors Project

Alyssandra M. Cordero 05/10/2019

Listing 1: PatternMatch Class - Sequence 1

```
1  /*****************************************************************************
2   * The PatternMatch class has the purpose of finding at what shift a ↩
       certain pattern is found inside of a
3   * FAFSA sequence.
4   *****************************************************************************/
5  public class PatternMatch{
6   public static void main(String [] args){
7    char [] textA; //textA char[] declaration.
8    char [] patternA; //patternA char[] declaration.
9    //String text that contains the FAFSA sequence.
10   String text = " ↩
        GTTCTGGACGTACTGTGTCAGTGTGTCGATACCCCGGCGCATATCGACGGGTTTTACGACCAGGAATACG"+
11    "TTATCAGGCGTCAGCATGGCGAAGAGCCCGGAAAACATCGGTTAACTGAGAAGGCTGGCAGCACATCCGG"+
12    "ATACCTCCGGGAAGGAAAAGTGTGACAGGCTCATCCGACAATGGTCTGCCATCAGCCATACCGGGAGCGC"+
13    "CAGACACTGAAACTGGAATAATTTCAGGTGCTCTGGCTCGTTTTTCGGCTTTTGCGACATCCTGCGGCCA";
14   //String pattern that contains the pattern to find in the sequence.
15   String pattern = "GGC";
16   //textA array initialization.
17   textA = store(text);
18   //patternA array initialization.
19   patternA = store(pattern);
20   System.out.println("Pattern occours at shift:"+ boyerMooreSA(patternA, ↩
       textA));//boyerMooreSA method call.
21  }
22  /*************************************************************************
23   * The store method, store a String into a char array.
24   * @param s The String with the element to store in the array.
25   * @return output char[] with the chars of the s.
26   *************************************************************************/
27  public static char[] store(String s){
28   char [] output = new char [s.length()];//output initialization.
29   //for loop that goes trough the char [] storing each character of the ↩
       String into each [] position.
30   for(int i = 0; i < s.length(); i++){
31    output[i] = s.charAt(i);
32   }
33   return output;
34  }
35  /*************************************************************************
```

```
36     * The boyerMooreSA method is used to find the shift position of a pattern inside a sequence.
37     * @param pattern char[] that contains the pattern to look for in the sequence.
38     * @param text char[] that contains the characters of the text file that contains the sequence.
39     * @return i that is the shift where the pattern was found.
40     * @return -1 if the pattern was not found in the sequence.
41     *************************************************************************/
42   public static int boyerMooreSA(char[] pattern, char[] text){
43     int patternSize = pattern.length; //The patternSize variable contains the lenght of the pattern array.
44     int textSize = text.length;    //The textSize variable contains the lenght of the text array.
45     int i = 0, j = 0;              // i and j initialization.
46
47     while((i+patternSize) <= textSize){ //Keep looping until textSize > (i+patternSize).
48       j = patternSize - 1;       //Initialize j to patternSize -1.
49       while (text [i + j] == pattern[j]){ //Keep looping until text [i+j] == pattern[j].
50         j--;                     //While (text[i+j] == pattern) keep decrementing j.
51         if(j < 0)                //if(j < 0) return counter.
52           return i;
53       }
54       i++;                       //increment counter.
55     }
56     return -1;                   //retun -1 (pattern not found).
57   }
58 }
```

Listing 2: PatternMatch Class - Sequence 2

```
1  /*************************************************************************
2   * The PatternMatch2 class has the purpose of finding at what shift a certain pattern is found inside of a
3   * FAFSA sequence.
4   *************************************************************************/
5  public class PatternMatch2{
6   public static void main(String [] args){
7     char [] textA; //textA char[] declaration.
8     char [] patternA; //patternA char[] declaration.
9     //String text that contains the FAFSA sequence.
10    String text = "QIKDLLVSSSTDLDTTLVLVNAIYFKGMWKTAFNAEDTREMPFHVTKQESKPVQMMCMNNSFNVATLPAE"+
11      "KMKILELPFASGDLSMLVLLPDEVSDLERIEKTINFEKLTEWTNPNTMEKRRVKVYLPQMKIEEKYNLTS"+
```

```
12      "VLMALGMTDLFIPSANLTGISSAESLKISQAVHGAFMELSEDGIEMAGSTGVIEDIKHSPESEQFRADHP"+
13      "FLFLIKHNPTNTIVYFGRYWSP";
14    //String pattern that contains the pattern to find in the sequence.
15    String pattern = "DLE";
16    //textA array initialization.
17    textA = store(text);
18    //patternA array initialization.
19    patternA = store(pattern);
20    System.out.println("Pattern occours at shift:"+ boyerMooreSA(patternA, ↩
          textA));//boyerMooreSA method call.
21  }
22  /**************************************************************************
23    * The store method, store a String into a char array.
24    * @param s The String with the element to store in the array.
25    * @return output char[] with the chars of the s.
26    *************************************************************************/
27  public static char[] store(String s){
28    char [] output = new char [s.length()];//output initialization.
29    //for loop that goes trough the char [] storing each character of the ↩
          String into each [] position.
30    for(int i = 0; i < s.length(); i++){
31      output[i] = s.charAt(i);
32    }
33    return output;
34  }
35  /**************************************************************************
36    * The boyerMooreSA method is used to find the shift position of a ↩
          pattern inside a sequence.
37    * @param pattern char[] that contains the pattern to look for in the ↩
          sequence.
38    * @param text char[] that contains the characters of the text file ↩
          that contains the sequence.
39    * @return i that is the shift where the pattern was found.
40    * @return -1 if the pattern was not found in the sequence.
41    *************************************************************************/
42  public static int boyerMooreSA(char[] pattern, char[] text){
43    int patternSize = pattern.length; //The patternSize variable contains ↩
          the lenght of the pattern array.
44    int textSize = text.length;    //The textSize variable contains the ↩
          lenght of the text array.
45    int i = 0, j = 0;              // i and j initialization.
46
47    while((i+patternSize) <= textSize){ //Keep looping until textSize > ↩
          (i+patternSize).
48      j = patternSize - 1;       //Initialize j to patternSize -1.
49      while (text [i + j] == pattern[j]){ //Keep looping until text [i+j] ↩
            == pattern[j].
```

```
50        j--;                        //While (text[i+j] == pattern) keep ↩
              decrementing j.
51        if(j < 0)                   //if(j < 0) return counter.
52          return i;
53       }
54       i++;                         //increment counter.
55     }
56     return -1;                     //retun -1 (pattern not found).
57   }
58 }
```

Listing 3: PatternMatch3 Class

```
1  /*****************************************************************************
2    * The PatternMatch3 class has the purpose of finding at what shift a ↩
         certain pattern is found inside of a
3    * FAFSA sequence.
4    *****************************************************************************/
5  public class PatternMatch3{
6    public static void main(String [] args){
7      char [] textA; //textA char[] declaration.
8      char [] patternA; //patternA char[] declaration.
9      //String text that contains the FAFSA sequence.
10     String text = ↩
           "MDSKGSSQKGSRLLLLLVVSNLLLCQGVVSTPVCPNGPGNCQVSLRDLFDRAVMVSHYIHDLSS"+
11       "EMFNEFDKRYAQGKGFITMALNSCHTSSLPTPEDKEQAQQTHHEVLMSLILGLLRSWNDPLYHL"+
12       "VTEVRGMKGAPDAILSRAIEIEEENKRLLEGMEMIFGQVIPGAKETEPYPVWSGLPSLQTKDED"+
13       "ARYSAFYNLLHCLRRDSSKIDTYLKLLNCRIIYNNNC";
14     //String pattern that contains the pattern to find in the sequence.
15     String pattern = "LLL";
16     //textA array initialization.
17     textA = store(text);
18     //patternA array initialization.
19     patternA = store(pattern);
20     System.out.println("Pattern occours at shift:"+ boyerMooreSA(patternA, ↩
           textA));//boyerMooreSA method call.
21   }
22   /*************************************************************************
23     * The store method, store a String into a char array.
24     * @param s The String with the element to store in the array.
25     * @return output char[] with the chars of the s.
26     *************************************************************************/
27   public static char[] store(String s){
28     char [] output = new char [s.length()];//output initialization.
29     //for loop that goes trough the char [] storing each character of the ↩
           String into each [] position.
30     for(int i = 0; i < s.length(); i++){
```

```
31      output[i] = s.charAt(i);
32    }
33    return output;
34  }
35  /**********************************************************************
36    * The boyerMooreSA method is used to find the shift position of a ↩
          pattern inside a sequence.
37    * @param pattern char[] that contains the pattern to look for in the ↩
          sequence.
38    * @param text char[] that contains the characters of the text file ↩
          that contains the sequence.
39    * @return i that is the shift where the pattern was found.
40    * @return -1 if the pattern was not found in the sequence.
41    **********************************************************************/
42  public static int boyerMooreSA(char[] pattern, char[] text){
43    int patternSize = pattern.length; //The patternSize variable contains ↩
          the lenght of the pattern array.
44    int textSize = text.length;    //The textSize variable contains the ↩
          lenght of the text array.
45    int i = 0, j = 0;              // i and j initialization.
46
47    while((i+patternSize) <= textSize){ //Keep looping until textSize > ↩
          (i+patternSize).
48      j = patternSize - 1;       //Initialize j to patternSize -1.
49      while (text [i + j] == pattern[j]){ //Keep looping until text [i+j] ↩
            == pattern[j].
50        j--;                     //While (text[i+j] == pattern) keep ↩
              decrementing j.
51        if(j < 0)                //if(j < 0) return counter.
52          return i;
53      }
54      i++;                       //increment counter.
55    }
56    return -1;                   //retun -1 (pattern not found).
57  }
58 }
```

Listing 4: PatternMatch4 Class

```
1  /**********************************************************************
2    * The PatternMatch4 class has the purpose of finding at what shift a ↩
         certain pattern is found inside of a
3    * FAFSA sequence.
4    **********************************************************************/
5  public class PatternMatch4{
6    public static void main(String [] args){
7      char [] textA; //textA char[] declaration.
```

```
8     char [] patternA; //patternA char[] declaration.
9     //String text that contains the FAFSA sequence.
10    String text = " ↩
          ADQLTEEQIAEFKEAFSLFDKDGDGTITTKELGTVMRSLGQNPTEAELQDMINEVDADGNGTID"+
11      "FPEFLTMMARKMKDTDSEEEIREAFRVFDKDGNGYISAAELRHVMTNLGEKLTDEEVDEMIREA"+
12      "DIDGDGQVNYEEFVQMMTAK";
13    //String pattern that contains the pattern to find in the sequence.
14    String pattern = "QLT";
15    //textA array initialization.
16    textA = store(text);
17    //patternA array initialization.
18    patternA = store(pattern);
19    System.out.println("Pattern occours at shift:"+ boyerMooreSA(patternA, ↩
          textA));//boyerMooreSA method call.
20  }
21  /**************************************************************************
22    * The store method, store a String into a char array.
23    * @param s The String with the element to store in the array.
24    * @return output char[] with the chars of the s.
25    *************************************************************************/
26  public static char[] store(String s){
27    char [] output = new char [s.length()];//output initialization.
28    //for loop that goes trough the char [] storing each character of the ↩
          String into each [] position.
29    for(int i = 0; i < s.length(); i++){
30      output[i] = s.charAt(i);
31    }
32    return output;
33  }
34  /**************************************************************************
35    * The boyerMooreSA method is used to find the shift position of a ↩
          pattern inside a sequence.
36    * @param pattern char[] that contains the pattern to look for in the ↩
          sequence.
37    * @param text char[] that contains the characters of the text file ↩
          that contains the sequence.
38    * @return i that is the shift where the pattern was found.
39    * @return -1 if the pattern was not found in the sequence.
40    *************************************************************************/
41  public static int boyerMooreSA(char[] pattern, char[] text){
42    int patternSize = pattern.length; //The patternSize variable contains ↩
          the lenght of the pattern array.
43    int textSize = text.length;    //The textSize variable contains the ↩
          lenght of the text array.
44    int i = 0, j = 0;              // i and j initialization.
45
46    while((i+patternSize) <= textSize){ //Keep looping until textSize > ↩
```

```
        (i+patternSize).
47      j = patternSize - 1;        //Initialize j to patternSize -1.
48      while (text [i + j] == pattern[j]){ //Keep looping until text [i+j] ↩
          == pattern[j].
49        j--;                      //While (text[i+j] == pattern) keep ↩
            decrementing j.
50        if(j < 0)                 //if(j < 0) return counter.
51          return i;
52      }
53      i++;                        //increment counter.
54    }
55    return -1;                    //retun -1 (pattern not found).
56  }
57 }
```

Listing 5: PatternMatch5 Class

```
1  /*****************************************************************************
2   * The PatternMatch5 class has the purpose of finding at what shift a ↩
        certain pattern is found inside of a
3   * FAFSA sequence.
4   *****************************************************************************/
5  public class PatternMatch5{
6   public static void main(String [] args){
7     char [] textA; //textA char[] declaration.
8     char [] patternA; //patternA char[] declaration.
9     //String text that contains the FAFSA sequence.
10    String text = " ↩
          LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV"+
11      "EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG"+
12      "LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL"+
13      "GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX";
14    //String pattern that contains the pattern to find in the sequence.
15    String pattern = "CLY";
16    //textA array initialization.
17    textA = store(text);
18    //patternA array initialization.
19    patternA = store(pattern);
20    System.out.println("Pattern occours at shift:"+ boyerMooreSA(patternA, ↩
          textA));//boyerMooreSA method call.
21  }
22  /*****************************************************************************
23    * The store method, store a String into a char array.
24    * @param s The String with the element to store in the array.
25    * @return output char[] with the chars of the s.
26    *****************************************************************************/
27  public static char[] store(String s){
```

```
28    char [] output = new char [s.length()];//output initialization.
29    //for loop that goes trough the char [] storing each character of the ↩
          String into each [] position.
30    for(int i = 0; i < s.length(); i++){
31      output[i] = s.charAt(i);
32    }
33    return output;
34  }
35  /**************************************************************************
36    * The boyerMooreSA method is used to find the shift position of a ↩
          pattern inside a sequence.
37    * @param pattern char[] that contains the pattern to look for in the ↩
          sequence.
38    * @param text char[] that contains the characters of the text file ↩
          that contains the sequence.
39    * @return i that is the shift where the pattern was found.
40    * @return -1 if the pattern was not found in the sequence.
41    **************************************************************************/
42  public static int boyerMooreSA(char[] pattern, char[] text){
43    int patternSize = pattern.length; //The patternSize variable contains ↩
          the lenght of the pattern array.
44    int textSize = text.length;    //The textSize variable contains the ↩
          lenght of the text array.
45    int i = 0, j = 0;              // i and j initialization.
46
47    while((i+patternSize) <= textSize){ //Keep looping until textSize > ↩
          (i+patternSize).
48      j = patternSize - 1;       //Initialize j to patternSize -1.
49      while (text [i + j] == pattern[j]){ //Keep looping until text [i+j] ↩
            == pattern[j].
50        j--;                     //While (text[i+j] == pattern) keep ↩
              decrementing j.
51        if(j < 0)                //if(j < 0) return counter.
52          return i;
53      }
54      i++;                       //increment counter.
55    }
56    return -1;                   //retun -1 (pattern not found).
57  }
58  }
```

Listing 6: LookingForMutation Class

```
1  /**************************************************************************
2    * The LookingForMutation class is designed to find a mutation pattern ↩
         on a file with a FAFSA sequence.
3    **************************************************************************/
```

```
4  import java.io.*;                                    //io library to handle ↩
       files.
5  import java.util.Scanner;                            //Scanner library for the ↩
       file Scanner.
6  public class LookingForMutation{
7   public static void main(String [] args) throws IOException{
8
9     String fileName = "TW_mouse_mutation.txt"; //String variable fileName ↩
         that holds the name of the file.
10    File file = new File(fileName);
11    Scanner inputFile = new Scanner(file);       //Scanner inputFile to go ↩
         trough the file.
12    char [] textA;                               //textA array ↩
         initialization.
13    char [] patternA;                            //patternA array ↩
         initialization.
14
15    String text = "";                            //String text ↩
         initialization .
16    String line = inputFile.nextLine();
17    while(inputFile.hasNext()){ //while loop that goes trugh the file ↩
         until the file does not have a next line.
18     text += line;          //Store each line of the .txt file in the ↩
          String text by concatination.
19     line = inputFile.nextLine();
20    }
21    text += line;                  //Read the last lane of the file.
22    inputFile.close();             //Close the file.
23
24    String pattern = "GCA";           //String pattern that holds the ↩
         pattern.
25    //Text array initialization. Stores each character of the text file in ↩
         a position of the array.
26    textA = store(text);
27    //Pattern array initialization. Stores each character of the pattern ↩
         in a position of the array.
28    patternA = store(pattern);
29    System.out.println("Pattern occours at shift:"+boyerMooreSA(patternA, ↩
         textA)); //boyerMooreSA method call.
30  }
31  /*************************************************************************
32    * The store method, store a String into a char array.
33    * @param s The String with the element to store in the array.
34    * @return output char[] with the chars of the s.
35    ************************************************************************/
36  public static char[] store(String s){
37    char [] output = new char [s.length()];//output initialization.
```

```
38    //for loop that goes trough the char [] storing each character of the ↩
         String into each [] position.
39    for(int i = 0; i < s.length(); i++){
40      output[i] = s.charAt(i);
41    }
42    return output;
43  }
44  /*****************************************************************************
45    * The boyerMooreSA method is used to find the shift position of a ↩
         pattern inside a sequence.
46    * @param pattern char[] that contains the pattern to look for in the ↩
         sequence.
47    * @param text char[] that contains the characters of the text file ↩
         that contains the sequence.
48    * @return i that is the shift where the pattern was found.
49    * @return -1 if the pattern was not found in the sequence.
50    *****************************************************************************/
51  public static int boyerMooreSA(char[] pattern, char[] text){
52    int patternSize = pattern.length; //The patternSize variable contains ↩
         the lenght of the pattern array.
53    int textSize = text.length;    //The textSize variable contains the ↩
         lenght of the text array.
54    int i = 0, j = 0;              // i and j initialization.
55
56    while((i+patternSize) <= textSize){ //Keep looping until textSize > ↩
         (i+patternSize).
57      j = patternSize - 1;       //Initialize j to patternSize -1.
58      while (text [i + j] == pattern[j]){ //Keep looping until text [i+j] ↩
           == pattern[j].
59        j--;                     //While (text[i+j] == pattern) keep ↩
             decrementing j.
60        if(j < 0)                //if(j < 0) return counter.
61          return i;
62      }
63      i++;                       //increment counter.
64    }
65    return -1;                   //retun -1 (pattern not found).
66  }
67  }
```