

```

1  #include <iostream>
2  #include<ostream>
3  #include <iomanip>
4  #include <fstream>
5  #include <cstdlib>
6  #include <vector>
7  #include <time.h>
8  #include <sstream>
9
10
11
12  typedef std::vector<int> SimilarityVector;
13  typedef std::vector<SimilarityVector> SimilarityMatrix;
14  typedef std::vector<std::string> SequenceVector;
15
16  timespec timespec_diff(timespec start, timespec end);
17  void printMatrix(const SimilarityMatrix &matrix);
18  void sequenceAlgo(SimilarityMatrix *similarityMatrix, SequenceVector *sequences);
19
20
21  /**
22   * Main function.
23   *
24   */
25  int main(int argc, char *argv[])
26  {
27
28      std::ifstream files[2];
29
30      //files[0].open("C://Krishna/HPC-Files/Prog-Assign2/sequence.txt");
31      files[0].open("C://Krishna/HPC-Files/Prog-Assign2/HIV-1_db.fasta");
32      //files[1].open("C://Krishna/HPC-Files/Prog-Assign2/unknown.txt");
33      files[1].open("C://Krishna/HPC-Files/Prog-Assign2/HIV-1_Polymerase.txt");
34      if(!(files[0]&&files[1]))
35      {
36          std::cerr<<"Unable To load the file";
37          exit(EXIT_FAILURE);
38      }
39
40      // read files
41      SequenceVector sequences(2);
42
43      std::string line;
44      for(int i = 0; i < 2; ++i)
45      {
46          std::ifstream &file = files[i];
47          std::string &sequence = sequences[i];
48          while(getline(file, line))
49          {
50              if((line.size() > 0) && (line[line.size() - 1] == '\r'))
51                  line.resize(line.size() - 1);
52              sequence += line;
53          }
54      }
55
56      // check the sequence size is >= the sample size
57      int rows = sequences[0].size() + 1;
58      int cols = sequences[1].size() + 1;
59      if(rows < cols)
60      {
61          exit(EXIT_FAILURE);
62      }
63
64      // create the matrix, setting all cells to -1
65      SimilarityMatrix similarityMatrix(rows, SimilarityVector(cols, -1));
66

```

```

67
68 // set the first row and first column to 0
69 for(int r = 0; r < rows; ++r)
70     similarityMatrix[r][0] = 0;
71 for(int c = 0; c < cols; ++c)
72     similarityMatrix[0][c] = 0;
73
74 // start timing
75 struct timespec start, finish;
76 clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &start);
77
78 sequenceAlgo(&similarityMatrix, &sequences);
79
80
81
82
83 clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &finish);
84
85
86     printMatrix(similarityMatrix);
87
88 std::cerr << "Total time (nanoseconds): " << timespec_diff(start, finish).tv_nsec << std::endl;
89
90     return 0;
91 }
92
93
94
95 void sequenceAlgo(SimilarityMatrix *similarityMatrix, SequenceVector *sequences)
96 {
97     int rows = similarityMatrix->size();
98     int cols = (*similarityMatrix)[0].size();
99
100
101
102 // Sequencing the Matrix
103 for(int j=1 ; j< rows;j++)
104 {
105
106
107     for (int i = 1 ; i <=cols; i++)
108     {
109
110         int row = j;
111         int col = i;
112
113
114
115
116         int options[3];
117
118         // Algorithm goes here
119         options[0] = (*similarityMatrix)[row - 1][col - 1] + ((*sequences)[0][row - 1] == (*sequences)[1][
col - 1] ? 1 : -1);
120         options[1] = (*similarityMatrix)[row][col - 1] - 2;
121         options[2] = (*similarityMatrix)[row - 1][col] - 2;
122
123         int value = 0;
124         for(int o = 0; o < 3; ++o)
125             if(options[o] > value)
126                 value = options[o];
127
128
129         (*similarityMatrix)[row][col] = value;
130     }
131

```

```

132
133
134
135
136     }
137 }
138
139
140 void printMatrix(const SimilarityMatrix &matrix)
141 {
142     for(SimilarityMatrix::const_iterator rowit = matrix.begin(); rowit != matrix.end(); ++rowit)
143     {
144         for(SimilarityVector::const_iterator colit = rowit->begin(); colit != rowit->end(); ++colit)
145             std::cout << std::setw(3) << *colit << ' ';
146         std::cout << '\n';
147     }
148     std::cout << std::flush;
149 }
150
151
152
153
154 timespec timespec_diff(timespec start, timespec end)
155 {
156     timespec temp;
157     if ((end.tv_nsec - start.tv_nsec) < 0)
158     {
159         temp.tv_sec = end.tv_sec - start.tv_sec-1;
160         temp.tv_nsec = 1000000000 + end.tv_nsec - start.tv_nsec;
161     }
162     else
163     {
164         temp.tv_sec = end.tv_sec - start.tv_sec;
165         temp.tv_nsec = end.tv_nsec - start.tv_nsec;
166     }
167     return temp;
168 }

```