**Bio Informatics Computing**

MUKHASIR SHAH SYED

109137268

Task: median string and motif finding using L-mer tree structure.

Steps of operation:

1. Read input fasta sequence file "HMP-617.fa" (arbitary number of sequences and arbitary length each) and store that in a list(data structure).
2. Traverse through L-mer tree structure and find the median string; using a priority queue of size 5, keep the 5 best median strings and their total\_distances.
3. Using 5 best median strings, find the corresponding motif consensus strings/score/positions; The original appearing order of the DNA sequences should be kept in the reported positions, i.e., s=(s1....st), where s1 is the starting position of motif in DNA sequence1 and so on where st is starting position of motif in DNA sequence t.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Date : Mar-25-2015 Author: Mukhasir Shah Syed

\* Program written will be used to find median string and motif from given set of DNA sequences and the length of the motif by implementing L-mer tree way. We would report the best 5 median strings and their corresponding motif consensus strings and positions.

\* INPUT:

\* A set of DNA sequences (arbitary length each) from "HMP-617.fa" file and motif length (L-mer) of size 6.

\* OUTPUT:

\* We will display 5 best median strings(with total distance), corresonding motif consensus strings(with conseensus scores), and motif positions (for each motif string).

\* Firstly, we will read all DNA sequences from "HMP-617.fa" and place them into a list of key value pair.

\* Secondly, we would traverse through L-mer tree structure by starting at "AAAAAA".

\* We will loop through Vertexes in L-mer tree in each DNA sequence and find out hamming

score and based on conditions if total distance for the vertex is greater than bestdistance of 6-mer then ByPass the vertex, else go to next vertex.

\* If length of vertex is 6 then get the calculation to be done for that 6-mer leaf and then go to next vertex.

\* Place 6-mers and their distance into a dictionary and sort them based on Total distance in ascending order.

\* After sorting is completed, we can find best 5 median strings for which motif consensus string, motif consensus score and motif position in each DNA sequences with less hamming score.

\* TotalDistance() method is used to get all median strings and their hamming score for each substring of size "6" in each DNA sequence provided. We will pass Vertex value and DNA sequence, Score would be update with addition of new score to old score.

\* FindMotifPositions() method will find one motif position in each DNA sequence with low hamming score and place that value and its position in sequence into list. We would pass median value from top 5 median values selected, DNA sequences and List of key value pair.

\* FindConsensus() method is used to find motif consensus score and motif consensus score.

\* ByPass() method is used to traverse through next vertex at same level.

\* NextVertex() method is used to traverse through next vertex starting at next level. If level is at 6 then go to next vertex at same level. If vertex is last child then go to next vertex a level up.

\* To complie/run the code we have to create a Console Application using Visual Studio 2013 and compile/build this application which will create an executable file then run the

".exe" application to execute the code.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MotifFind\_Tree

{

class Program

{

static void Main(string[] args)

{

string HMP\_Label = string.Empty;

string HMP\_Seq = string.Empty;

//Initialize KeyValuePair List for storing database sequences.

List<KeyValuePair<string, string>> HMP\_Data = new List<KeyValuePair<string, string>>();

//Loop throug database file to store the label and bases respectively in key-value pair format.

foreach (string Database in File.ReadLines(@"E:\Bio Informatics\Assignment 5\HMP-617.fa"))

{

if (Database.Contains(">"))

{

if (HMP\_Seq != string.Empty)

{

HMP\_Data.Add(new KeyValuePair<string, string>(HMP\_Label, HMP\_Seq));

HMP\_Seq = string.Empty;

}

HMP\_Label = Database;

}

else

{

HMP\_Seq += Database;

}

}

HMP\_Data.Add(new KeyValuePair<string, string>(HMP\_Label, HMP\_Seq));

//variables used further

string st = string.Empty;

int bestDistance = 0;

int optimisticDistance = 0;

int TotalDist = 0;

string prefix = "AAAAAA";

int x = 6;

bool Flag = true;

st = prefix;

int TreeValue\_Level = prefix.Length;

string bestWord = string.Empty;

//Initialize a List to store visited vertexes of 6-mers

List<string> VisitedNodes = new List<string>();

//Initialize a dictionary to store Median Values processed.

Dictionary<string, int> MedianDistanceData = new Dictionary<string, int>();

//Loop through to check each vertex applicable based on conditions

while (x > 0)

{

//if level of vertex is less than 6 then go through else go into else block.

if (x < 6)

{

//Loop through each DNA sequence by passing vertex from L-mer tree and find out total distance.

foreach (KeyValuePair<string, string> HMPMOtif\_Seq in HMP\_Data)

{

optimisticDistance += TotalDistance(prefix, HMPMOtif\_Seq.Value);

}

st = prefix;

//Check if total distance is greater than best distance then go through and bypass vertex.

//else goto next vertex method to find next vertex in L-mer tree.

if (optimisticDistance > bestDistance)

{

//This method will by pass through next vertex at same level or other level based on vertex position and level.

ByPass(ref st, ref TreeValue\_Level, 6, 4);

prefix = st;

x = TreeValue\_Level;

}

else

{

//This method will go to next vertex in L-mer tree structure.

NextVertex(ref st, ref TreeValue\_Level, 6, 4);

prefix = st;

x = TreeValue\_Level;

}

optimisticDistance = 0;

}

else

{

//Loop through each DNA sequence by passing vertex from L-mer tree and find out total distance.

foreach (KeyValuePair<string, string> HMPMOtif\_Seq in HMP\_Data)

{

TotalDist += TotalDistance(prefix, HMPMOtif\_Seq.Value);

}

VisitedNodes.Add(prefix);

//Add median string and its total distance for respective median string into dictionary.

MedianDistanceData.Add(prefix, TotalDist);

//Check if Total distance is less than best distance then set new total distance as best distance.

//set the median as best word

if (TotalDist < bestDistance)

{

bestDistance = TotalDist;

bestWord = st;

}

if (Flag)

{

bestDistance = TotalDist;

bestWord = st;

Flag = false;

}

//After setting best distance and best word now move to next vertex in L-mer tree structure.

NextVertex(ref st, ref TreeValue\_Level, 6, 4);

prefix = st;

x = TreeValue\_Level;

TotalDist = 0;

}

}

string FinalVisitedNodes = string.Join("\n", VisitedNodes.ToArray());

File.WriteAllText(@"E:\Bio Informatics\Assignment 6\TotalVisited\_6mers.txt", FinalVisitedNodes);

//Linq query to sort data present in disctionary based on value(which is score) in ascending order.

var items = from pair in MedianDistanceData

orderby pair.Value ascending

select pair;

////Initialize KeyValuePair List for storing median values with their score.

List<KeyValuePair<string, int>> med = new List<KeyValuePair<string, int>>(items);

string median\_string = string.Empty;

string motifps = string.Empty;

string cValueFind = string.Empty;

string motifCon\_String = string.Empty;

string FinalValue = string.Empty;

//Initialize KeyValuePair List for storing motif data and their positions.

List<KeyValuePair<string, int>> MotifPositionData = null;

//Loop through 5 best median values.

for (int counter = 0; counter < 5; counter++)

{

motifps = "";

MotifPositionData = new List<KeyValuePair<string, int>>();

//Loop through DNA Sequences

foreach (KeyValuePair<string, string> b in HMP\_Data)

{

//Build a string value with median string and its total distance

median\_string = "median string: " + med[counter].Key + " (total\_distance = " + med[counter].Value + ")";

//This method is used to find motif string and its position in DNA sequence which have low

//hamming distance.

FindMotifPositions(med[counter].Key, b.Value, ref MotifPositionData);

}

cValueFind = string.Empty;

//Loop through all motif strings for a particular median string and form a string to display.

foreach (KeyValuePair<string, int> motifSeq in MotifPositionData)

{

//Concatinate motif strings into one string so that it could be passed to FindConsensus to find

//motif consensus string and motif consensus score.

cValueFind += motifSeq.Key + ";";

//Concatinate position and string to diplay in output.

motifps += motifSeq.Value + "(" + motifSeq.Key + "), ";

motifps = motifps.Trim();

}

//Build format and concatinate final value into the string to be displayed.

motifps = "motif positions/string s=(s1,s2,..,st-1,st): " + "\n\t" + motifps.Substring(0, motifps.Length - 1);

//This method is used to find out motif consensus string and motif consensus score by passing

//all motif strings found above.

motifCon\_String = FindConsensusScore(cValueFind);

//Concatinate all the string to display the final outpout format.

FinalValue += median\_string + "\n" + motifCon\_String + "\n" + motifps + "\n\n";

}

File.WriteAllText(@"E:\Bio Informatics\Assignment 6\ConsensusResult.txt", FinalValue);

}

/// <summary>

/// This method is used to ByPass through Vertexes in L-mer tree structure.

/// </summary>

/// <param name="CurrentLeaf"></param>

/// <param name="level"></param>

/// <param name="length"></param>

/// <param name="maxValue"></param>

public static void ByPass(ref string CurrentLeaf, ref int level, int length, int maxValue)

{

//Convert string into character array

char[] CurLeaf\_int = StringNumber(CurrentLeaf, "N").ToArray();

bool FinalValue = true;

int a = 0;

//Loop through character array and increment the position to next number.

for (a = (level-1); a >= 0; a--)

{

//Check if Value at particular location in character array less than max value (4) then

//increment the value to next value.

if (Convert.ToInt16(CurLeaf\_int[a].ToString()) < maxValue)

{

CurLeaf\_int[a] = Convert.ToChar((Convert.ToInt16(CurLeaf\_int[a].ToString()) + 1).ToString());

FinalValue = false;

break;

}

}

//Change character array into string.

string Final = String.Join("", CurLeaf\_int.ToArray());

//Convert string in number to string in alphabets.

Final = StringNumber(Final, "S");

CurrentLeaf = Final.Substring(0, a + 1);

level = CurrentLeaf.Length;

if (FinalValue)

{

a = Final.Length-1;

CurrentLeaf = Final.Substring(0, a + 1);

level = 0;

}

}

/// <summary>

/// This method is used to traverse through next vertex in L-mer tree structurel

/// </summary>

/// <param name="s"></param>

/// <param name="level"></param>

/// <param name="length"></param>

/// <param name="maxvalue"></param>

public static void NextVertex(ref string s,ref int level, int length, int maxvalue)

{

//Variables to be used in this method

string NextV = string.Empty;

int counter = 0;

bool FinalValue = true;

//Convert string with aplhabets into number character array.

char[] CurLeaf\_int = StringNumber(s, "N").ToArray();

//Check if level is less than length then go to next vertex starting at next level..

if(level<length)

{

//Loop through character array

for(int i=0;i<CurLeaf\_int.Length;i++)

{

NextV += CurLeaf\_int[i];

}

NextV += "1";

level = NextV.Length;

s = StringNumber(NextV, "S");

}

else

{

//loop through character array.

for (counter = length-1; counter >= 0; counter--)

{

//Check if value at certain index is less than max value (4)

//then add "1" to value.

if (Convert.ToInt16(CurLeaf\_int[counter].ToString()) < maxvalue)

{

CurLeaf\_int[counter] = Convert.ToChar((Convert.ToInt16(CurLeaf\_int[counter].ToString()) + 1).ToString());

FinalValue = false;

level = counter + 1;

break;

}

}

//Change character array to String.

string Final = String.Join("", CurLeaf\_int.ToArray());

//Convert string in number to string in alphabets.

Final = StringNumber(Final, "S");

//Set the values in s and level to be returned.

s = Final.Substring(0, counter + 1);

level = s.Length;

if (FinalValue)

{

counter = Final.Length-1;

s = Final.Substring(0, counter + 1);

level = 0;

}

}

}

/// <summary>

/// This method is used to convert string into number format and viceversa.

/// </summary>

/// <param name="value"></param>

/// <param name="type"></param>

/// <returns></returns>

public static string StringNumber(string value, string type)

{

string IntValue = string.Empty;

//Check if type is "N" or "S". If "N" then we will prepare a string with alphabets with numbers

//for string with corresponding alphhabet values.

//If "S" then convert Number string into alphabet format.

if (type == "N")

{

//Convert string to character array

char[] value\_array = value.ToArray();

foreach (char c in value\_array)

{

switch (c)

{

case 'A':

IntValue += "1";

break;

case 'C':

IntValue += "2";

break;

case 'G':

IntValue += "3";

break;

case 'T':

IntValue += "4";

break;

}

}

}

else if(type=="S")

{

//Convert string to character array

char[] value\_array = value.ToArray();

foreach (char c in value\_array)

{

switch (c)

{

case '1':

IntValue += "A";

break;

case '2':

IntValue += "C";

break;

case '3':

IntValue += "G";

break;

case '4':

IntValue += "T";

break;

}

}

}

//return the respective value requested.

return IntValue;

}

/// <summary>

/// This method is used to find out median string and its score. We will add new score to the existing score if that

/// median has already been placed into dictionary.

/// </summary>

/// <param name="medianValue"></param>

/// <param name="SequenceValue"></param>

public static int TotalDistance(string medianValue, string SequenceValue)

{

//Variables to be used in this method

bool setValue = true;

int k = 0;

int score = 0;

int substring\_position = 0;

//Loop through DNA sequence till the last substring string has size "6" and find out hamming score.

for (k = 0; k <= (SequenceValue.Length - medianValue.Length); k++)

{

score = 0;

//Get substring of size of l(L=6)

string tempValue = SequenceValue.Substring(k, medianValue.Length);

//Convert median and substring into character array for comparing each character at same position.

char[] median = medianValue.ToArray();

char[] tmpvalue = tempValue.ToArray();

//Loop through character array to find score of mismatch.

for (int l = 0; l < medianValue.Length; l++)

{

if (median[l] != tmpvalue[l])

{

score = score + 1;

}

}

int maxscore = score;

//We will check if the new hamming score value is less than the old hamming score value then set new value

//into variables so that that can be update into list.

if (maxscore != substring\_position)

{

if (setValue)

{

substring\_position = maxscore;

setValue = false;

}

}

//get the score for each substring.

if (maxscore <= substring\_position)

{

substring\_position = maxscore;

setValue = false;

}

}

//Return score for each sequence for corresponding vertex passed.

return substring\_position;

}

/// <summary>

/// This method is used to find the Motif Consensus Score and Motif Consensus String from all

/// motif strings retrieved for each median string.

/// </summary>

/// <param name="sequences"></param>

/// <returns></returns>

public static string FindConsensusScore(string sequences)

{

string modifyString = string.Empty;

int score = 0;

string consensus\_string = string.Empty;

//We will split string based on splitter ";".

string[] motifstrings = sequences.Split(';');

//Loop through each character present in each string.

for (int j = 0; j < 6; j++)

{

//Loop through each motif string to get character at "i" position to form a string.

for (int i = 0; i < motifstrings.Length; i++)

{

if (motifstrings[i] != "")

{

modifyString += motifstrings[i][j].ToString();

}

}

//We will add occurence of character in each column and get numeric number and add the sum of all value to get consensus score.

score += modifyString.GroupBy(x => x).OrderByDescending(x => x.Count()).First().Count();

//We will get maximun occured character and concatinate all values to get consensus string.

consensus\_string += modifyString.GroupBy(x => x).OrderByDescending(x => x.Count()).First().Key.ToString();

modifyString = "";

}

//Form a value in string to be displayed in output.

string mcs = "motif consensus string: " + consensus\_string + " (consensus\_score = " + score.ToString() + ")";

//return final string.

return mcs;

}

/// <summary>

/// This method is used to find motif positions and motif strings in for the 5 best median strings.

/// </summary>

/// <param name="medianValue"></param>

/// <param name="SequenceValue"></param>

/// <param name="MotifPositions"></param>

public static void FindMotifPositions(string medianValue, string SequenceValue, ref List<KeyValuePair<string, int>> MotifPositions)

{

//Variables used in this method.

bool setValue = true;

string v1 = string.Empty;

int v2 = 0;

int k = 0;

int score = 0;

int substring\_position = 0;

int z = 0;

//Loop through DNA sequence and get substring of sequence size "6" and find out hamming score.

for (k = 0; k <= (SequenceValue.Length - medianValue.Length); k++)

{

score = 0;

//Get substring of size of l(L=6)

string tempValue = SequenceValue.Substring(k, medianValue.Length);

//Convert median and substring into character array for comparing each character at same position.

char[] median = medianValue.ToArray();

char[] tmpvalue = tempValue.ToArray();

//Loop through character array to find score of mismatch.

for (int l = 0; l < medianValue.Length; l++)

{

if (median[l] != tmpvalue[l])

{

score = score + 1;

}

}

int maxscore = score;

//Below written if statement will set minimun hamming score.

if (maxscore != substring\_position)

{

if (setValue)

{

substring\_position = maxscore;

setValue = false;

}

}

//We will check if the new hamming score value is less than the old hamming score value then set new value

//into variables so that that can be update into list.

if (maxscore < substring\_position)

{

substring\_position = maxscore;

v1 = tempValue;

v2 = k;

}

//If hamming score value is zero/same value then we use the below code set first string as value to below

//variables.

if (maxscore == substring\_position)

{

if (z == 0)

{

v1 = tempValue;

v2 = k;

}

z++;

}

}

//Add motif strings and their corresponding score into list.

MotifPositions.Add(new KeyValuePair<string, int>(v1, v2));

}

}

}

***OUTPUT:***

median string: GAAAAA (total\_distance = 309)

motif consensus string: GAAAAA (consensus\_score = 3393)

motif positions/string s=(s1,s2,..,st-1,st):

194(GAAAAA),962(GAAAAA),4(GCAGAA),210(GAAAAA),35(GAAAAA),5(GTTAAA),482(GAAAAA),72(GAAAAA),46(GAAAAA),12(GAAAAA),6(GAGAAA),203(GAAAAA),80(GGTAAA),210(GAAAAA),35(GAAAAA),170(GAATAA),12(GAAAAA),46(GAAAAA),4(GAACAA),408(GAAAAA),165(GAAAAA),662(GAAAAA),819(GAAAAA),69(GAGAAA),66(GAGAAA),313(GAAAAA),162(GAAATA),473(TAAAAA),7(GAACAA),2(GAGTAA),300(GAAAAA),881(GAAAAA),216(GAAAAA),322(GAAAAA),158(GATAAA),189(GAAAAA),14(GAAAAA),140(GCAAAA),69(GAAGAA),28(GAAATA),137(GCAAAA),139(GAAAAA),2(GAAAAA),1085(GAAAAA),2(GAAACA),479(GAATAA),42(GAAAAT),579(GAAAAA),28(GAAAAT),75(GAAATA),2(GAAAAA),44(GACAAA),2(GAAGAA),2(GAAAAA),2(GAAAAT),247(GAAAAA),522(GAAAAG),19(GAGAAA),3(GAAAAA),317(GAAAAA),84(GCAAAA),818(GAAAAA),34(CAAAAA),357(GAAAAA),678(GAAAAA),210(GAAAAA),23(GCAAAA),23(GAAAAA),125(GAAAAA),115(GCAAAA),1014(GAAAAA),2(GAACAA),159(GAAAAA),3(GAAAAA),317(GAAAAA),54(GCACAA),128(GAAAGA),184(GAAAAA),170(CAAAAA),2226(GAAAAA),108(GAAAAA),74(GAACAA),72(GAAACA),2(GAAAAA),297(GAAAAA),17(GAAAAA),38(CAAAAA),10(GAAAAC),186(GAAAAG),2(GAAAAA),2(GAAAAA),66(GCAAAA),66(GAAAAA),6(GAAGAA),2(GAAAAC),2(GATAAA),138(GAAAAT),156(GACAAA),336(GAAAAA),80(GAAAAA),1124(GAAAAA),678(GAAAAA),369(GAAAAA),335(GAAAAA),178(GTAAAA),148(GCAAAA),6(GATAAA),177(GAAAAA),132(GAAATA),42(GAAAAA),243(GAAAAA),525(GAAAAA),45(GACAAA),251(GAAAAA),18(GAAAAA),407(GAAAAA),6(GAAAAC),116(GAAAAA),273(GAAAAA),181(GAAACA),25(GAAAAC),9(GACACA),140(GAAAAC),732(GAAAAA),16(GAGAAC),21(GATAAA),230(GCAAAA),425(GAAAAA),2(GACAAC),849(GAAAAA),2(GAAAAC),75(GAAAAA),12(GAACAA),99(GAATAT),36(GAACAA),326(GAAAAA),217(GAAAAA),703(GAAAAA),10(GAAAAA),388(GAAAAA),83(TAAAAA),159(GAAAAA),102(GAAAAA),17(GAAAAA),44(GAAAAA),185(CAAAAA),555(GAAAAA),2(GAAAAA),245(GAAAAA),62(GAAAAG),44(GAAAAA),42(GAAAAA),6(GATGAA),445(GAAATA),7(GAAAGA),7(GAAAAA),244(GAAAAA),401(GAAAAA),520(GAAAAA),122(GAAACA),42(GAAAAT),567(GAAAAA),1002(GAAAAA),1940(GAAAAA),57(GAAAAA),214(GAAAAA),444(GAAAAA),59(GAAAAA),414(GAAAAA),650(GAAAAA),2(GGAAAA),113(GAAAAA),77(GGAAAA),89(GAAACA),28(GAAAAA),101(GAAAAA),23(GAAAAG),257(GAAAAA),63(GCAAAA),230(GGAAAA),18(GAAAAG),18(GATAAA),11(TAAAGA),60(GAGAAA),107(GAAAAA),183(GAAAAA),2(GAATAA),24(GAAAAA),53(GAATAA),60(GAAAAA),109(GGAAAA),8(GAAAAA),2(GAAAAA),49(GAAAAA),35(GGAAAA),75(GAAATA),87(GAAAAA),2(GAAAAA),122(GACAAA),254(GAAAAA),8(GAAAAA),262(GAGAAA),12(GAAATA),257(GAAAAA),331(GAAAAA),39(GAAAAA),192(GAAAAA),63(CAGAAA),330(GAAAAA),2(GAAATC),118(GCAAGA),42(GCAAAA),105(GGCAAA),2(GAATAA),192(GAAAAA),4(GAAAAA),95(GAAAAA),210(GAAAAA),226(GAAAAA),7(GCAAAA),66(GAAAAA),54(GAAAAA),5(GAAAAT),36(GAAAGA),79(TAAAAA),368(GAAAAA),126(GAAAAA),15(GAAAAT),587(GAAAAA),2(GAAAAA),77(GCAGAA),54(GAAAAA),78(GAAAAA),93(GAAGAA),309(GAAAAA),144(GGAAAA),30(GAAAAA),338(GAAAAA),93(GAAAAA),42(GAAAAA),243(GAGAAA),37(CAAAAA),68(GAATAA),525(GAAAAA),45(GACAAA),251(GAAAAA),18(GAAAAA),407(GAAAAA),6(GAAAAC),110(GCAAAA),116(GAAAAA),273(GAAAAA),181(GAAACA),72(GGAACA),317(GAAAAA),2(GAACAA),849(GAAAAA),2(GAAAAC),75(GAACAA),15(GAACAA),38(GCACAA),280(GAACAA),60(GCAAAA),180(GACAAA),68(GAATAA),804(GAAAAA),131(GAAAAA),28(GAAAAA),155(GCAAAA),792(GAAAAA),245(GAAAAA),95(GAAAAA),17(GAAAAA),642(GAAAAA),488(GAAAAA),444(GAAAAA),245(GAAACA),6(GAAAAG),21(GGAAAA),57(GAAAAA),3(GCAAAA),246(GAAAAA),933(GAAAAA),32(GAAAAA),7(GAAAAA),588(GAAAAA),1140(GAAAAA),732(GAAAAA),42(GAAAAC),263(GAAAAA),144(GTAAAA),336(GAAAAA),570(GAAAAA),98(GACAAA),6(GAAAAT),156(GAAAAC),1041(GAAAAA),236(GAAAAA),2(GAAAAG),1044(GAAAAA),707(GAAAAA),449(GAAAAA),25(CAAAAA),8(GAAAAA),66(GAAAAA),2(GAAAAA),5(GACAAT),225(GACAAA),9(GAAAAT),46(GAAAAA),191(GAAAAA),129(GCAAAA),444(GAAAAA),2(GAAAAA),279(GAACAA),2(GAAAAA),2(GAAAAA),105(GAAAAT),74(GAAAAC),2(GAAAAA),195(GAAAAA),17(GAATAA),276(GAAAAA),2(GAAACG),153(GAAAAA),182(GGAAAA),498(GAAAAA),2(GAAAAA),23(GAAAAC),130(GCAAAA),438(GAAAAA),63(GATAAA),173(CAAAAA),96(GAAAAA),261(GAAATA),359(GAAAAA),62(CAAAAA),131(GAGAAA),2(GAAACA),537(GAAAAA),66(GAAAAT),112(GAAAAA),369(GAAAAA),42(GAAAAG),74(GATAAA),228(GAAAAA),1164(GAAAAA),482(GAAAAA),174(GATAAA),665(GAAAAA),126(GAGAAA),166(GGAAAA),6(GAAATA),22(TAAAAA),459(GAAAAA),234(GAAAAA),201(GAAAAA),18(GAAAAA),245(GGAAAA),83(GAAAAA),482(GAAAAA),879(GAAAAA),840(GAAAAA),16(GAAAAA),596(GAAAAA),702(GAAAAA),222(GAAAAA),93(GATAAA),231(GATAAA),47(GAATAA),870(GAAAAC),47(GGAAAA),338(GAAAAA),309(GAAAAT),2(GAATAA),437(GAAAAA),2(GAAGAA),2(GAAAAA),13(GAAAAA),42(GAGAAA),275(GAAAAA),1037(GAAAAA),615(GAAAAA),209(GAAAAA),741(GAAAAA),1388(GAAAAA),170(GAAAAA),275(GAAAAA),109(GTAAAA),1575(GAAAAA),401(GAAAAA),969(GAAAAA),207(GCAAAA),153(GAAAAA),18(GAAAAA),474(GAAAAA),78(GAAAAA),2(GACTCA),17(GGTAAA),107(GCAAAA),4(GATTCA),425(CAAAAA),159(GAAAAA),22(GAATAA),105(GAAAAA),170(GCAAAA),5(GGAAAA),201(GAACAA),1493(GAAAAA),1095(GAAAAA),94(GAAAAC),143(GCAAAA),16(GTAAAA),102(GAAAAA),885(GAAAAA),1405(GAAAAA),366(GAAAAA),25(GAAATA),503(GAAAAA),148(GTAAAA),377(CAAAAA),204(GCAAAA),105(GAAAAA),2(GAAAAA),129(GAAAAA),36(GAAAAA),239(GAAAAA),330(GAAAAA),171(GATAAA),63(GAAGAA),261(GCAAAA),600(GAAAAA),75(GAAACA),104(GAAACA),44(TAAAAA),93(GAAAAA),293(GAAAAA),75(GACAAA),177(GCAAAA),1527(GAAAAA),101(GAAAAA),95(GAAAGA),2(GAAAAA),6(GAAAAA),245(GAAGAA),181(GAAAAA),1536(GAAAAA),14(GAAACA),2(GAAAGA),351(GAAAAA),25(TAAAAA),62(GAAAAA),87(GAAAAT),117(GAAAAC),263(GAAAAT),702(GAAAAA),42(GATAAA),2(GAAGAA),31(GAATAA),525(GAAAAA),2(GATAAA),262(GAAAAA),117(CAAAAA),5(TAAAAA),317(GATAAA),156(GAAAAT),489(GAAAAA),308(GAAAAA),182(GAAATA),162(GAGAAA),2459(GAAAAA),818(GAAAAA),119(GAAAAG),40(GTAAAA),375(GAAAAG),71(GATGAA),792(GCAAAA),108(GAAAAA),2(GAAAAC),186(GAAAAA),66(GGAAAA),170(GCAAAA),2(GAAAAT),19(GAAAAA),2(GAAAAA),563(GCAAAA),366(GAAAAA),138(GAAAAA),1032(GAAAAA),33(GAAAAA),74(GAAAAT),149(GAAAAA),579(GAAAAA),234(GAAAAA),1092(GAAAAA),27(GAAAAA),232(GAAATA),396(GAAATA),8(GAAAAA),2(GAAAAA),206(GAAAAG),351(GAAAAA),6(GATAAA),447(GAAAAA),150(GAAAAA),2141(GAAAAA),132(GAAAAA),474(CAAAAA),326(GCAAAA),110(GGAAAA),99(GAAAAA),3(GAAAAA),143(GAAATA),300(GAACAA),5(GAAAAA),42(GAGAAA),87(GACAAA),3(GATAAA),149(GAAAAA),51(GAAAAA),114(GCAAAA),78(GAAAAA),40(GGGAAA),2(GATAAA),124(TAAAAA),408(GAAAAA),113(GAAAAA),771(GATAAA),272(GAAAAA),1331(GAAAAA),620(GAAAAA),430(GAAAAA),254(GAAAAA),403(GGAAAA),86(GAAAAA),798(GAAAAA),237(GAAAAA),1248(GAAAAA),270(GAAAAA),47(GGAAAA),258(GAAAAA),752(GAAAAA),45(GAAAAA),1110(GAAAAA),65(GAAAAA),1254(GAAAAA),753(GAAAAA),72(GAAAAA),15(GAAAGA),305(GAAAAC),11(CAAAAA),4(GCAAAA),41(GCAAAA),89(GAATAA),43(GAAATA),1554(GAAAAA),3466(GAAAAA),147(GAAAAA),358(GAAAAA),23(GAAAAA),14(GAAGAA),253(GAAAAA),721(GAGAAA),300(GAGAAA),362(GAAAAA),558(GAAAAA),390(GAAAAA),4(GTAAAA),45(GTAAAA),44(GAAGAA),167(GAAAAA),2(GAATAA),55(GAAAAA),642(GAGAAA),176(GAAAAA),4(GTAAAA),2(GAATAA),855(GAAAAA),50(GAAAAA),245(GAAAAA),6(GAAAAA),1227(GAAAAA),38(CAAAAA),2(GAAAAA),24(GTAAAA),66(GAAAAA),215(GAAAAA),171(GAAAAA),995(GAAAAA),123(GAAGAA),237(GAAGAA),214(GAGAAA),18(GAACAA),84(GTAAAA),120(GAAAAA),14(GAAACA),2(GAAAGA),215(GAAAAG),176(GAAAAA),102(GAAGAA),402(GAAAAA),51(GAAAAT),311(GAAAAA),171(GTAAAA),113(GCAAAA),59(GACAAA),343(GAACAA),473(GAAAAA),96(GAAAAA),23(GAAAAA),369(GAAAAA),2(GAATAA),75(GAAACA),5(GAAGAA),372(GAAAAA),2(GAAGAA),15(GAACTA)

median string: CTGGCA (total\_distance = 345)

motif consensus string: CTGGCA (consensus\_score = 3357)

motif positions/string s=(s1,s2,..,st-1,st):

181(CTGACA),1677(CTGGCA),114(CTGGCA),401(CTGGCA),249(CTGGCA),106(CGGGCA),198(CTGGAA),500(ATGGCA),94(CTGGCA),99(CAGGCA),99(CTGGCA),25(CTGGTA),63(CGGGCA),401(CTGGCA),249(CTGGCA),132(CTGGCA),99(CAGGCA),43(ATGGAA),347(CTGGAA),0(ATGTCA),158(CTGGCA),485(CTGGCA),657(CTGGCA),23(CTGGAA),152(CCGGCA),349(CTGGCA),42(CTTGCA),307(CTGGCA),155(CTGGCA),36(CTGGGG),669(CTGGCA),30(CTGACA),71(CTGGCA),22(CTGGCC),29(CTGGCT),36(CGGGCA),20(CTGGCT),27(CTTGCA),18(CAGGCA),168(CAGGCA),189(CTGGCC),63(ATGGCA),8(CAGGCA),353(CTGGCA),56(CTGCCA),129(CTGGCA),104(CCGGCA),609(CTGGCA),335(CTGGCC),20(TTTGTA),1157(CTGGCA),105(CTGGCA),246(CTGGCA),21(CTGGCG),146(CTGGCA),54(CTGGCC),170(CTGGCA),252(ATGGCA),434(CTGGCA),80(CTCGCA),81(CGGGCA),222(CTGGCA),48(CTGGCA),459(CTGGCA),1428(CTGGCA),224(CTGGCA),11(TTTGCA),43(CTGGCG),21(CTGGCG),21(CTGGCA),149(CTGGCG),227(CTGGGA),25(ATGGCA),69(CTGGCT),80(CTCGCA),109(CTGTCA),147(CTGGGA),439(CTGGCA),40(CTGGCT),108(CTGGGA),48(CTGGCT),239(CGGGCA),403(CTGGCA),240(ATGGCA),244(CTGGTA),54(CTGGCT),0(GTGGCG),701(CTGGCA),635(CTGGCA),39(GTGGCA),364(CCGGCA),36(CTGACA),219(CTGGCA),140(CTGGCA),25(CCTGCA),67(CTGCCA),128(CTGGCA),132(CTGGCC),105(CTGGCA),114(CTGACA),22(TTGGCA),399(CTGGCA),282(CTGGCA),228(CTGGCA),465(CTGGAA),497(CTGGCA),147(TTGGCA),0(ATGGCA),193(CTGGTA),9(CTGGCT),97(CTGCCA),830(CTGGCA),135(CGGGCA),351(CTGGCA),192(CTGGAA),33(CTGGCA),756(CTGGCA),149(CTGGCA),288(CTGGCA),74(GGGGCA),33(CTGGCA),225(CTGGCA),27(CTCGCG),333(CAGGCA),132(CTGGCG),7(CGGGCA),219(CTGGCA),10(GTGGCA),206(CTGGCG),186(CTGGCA),348(CTGGCA),459(CTGGCA),56(CTGGAA),69(TTCGCA),18(CTGGCA),0(ATGGCA),96(CAGGCA),659(CTGGCA),20(CTGGCA),73(CTGGCA),128(CAGGCA),48(CTGGCT),43(CTGCCA),111(CTGGAA),180(CTGGCA),227(CTGGCA),42(CTGACA),26(CGGGCA),273(CTGGCA),461(CTGGCA),8(CTGGCA),106(CGGGCA),0(ATGGCA),138(CTGGCA),60(CTGGCA),48(CTGACA),22(CTGCCC),45(CTGTCA),1539(CTGGCA),98(CTGGCA),147(CTGGCA),60(CTGGAA),155(CTGGCA),990(CTGGCA),77(CTGGAA),61(CTGGCT),162(CTGGTA),18(CTGGCA),830(CTGGCA),516(CTGGCA),66(CTGGCA),495(CTGGCA),149(CTGGCA),36(CTGACA),77(ATGGCA),2(GTCGCA),56(TTGGCG),647(CTGGCA),0(ATGGCA),24(CTGGCC),0(ATGACA),57(CAGGCA),84(CTGTCA),23(CTGGTA),320(GTGGCA),109(CTGGTA),24(CTCTCA),67(CTGGCA),159(CTGACA),1368(CTGGCA),129(CTGGCA),163(CTGCCA),60(CTGGCA),215(CTGGGA),33(CTGGAA),20(TTTGTA),270(CTGGCA),48(CTGACA),30(CTGACA),0(GTGGCA),96(CTGGAA),102(CTGACA),182(CAGGTA),31(CTGTCA),51(CGGGCA),132(CTGGCA),35(CCGGCA),23(CTGGAA),125(CTGGCA),316(CCGGCA),83(CTGCCA),104(CTGGAA),126(CTGGCA),60(CTGCCA),35(CCGGCA),0(ATGGGA),126(CTGGCC),411(CAGGCA),194(CTGGAA),101(CTGGCA),219(CTGGCA),146(CTGGCA),87(CCGGCA),45(CTGGCC),470(CTGGCA),303(CGGGCA),835(CTGGCA),83(CTGGCA),327(CTGGCG),212(CTGGCT),56(CTGGCA),147(CTGGCA),94(CTGCCA),30(CTGGAA),189(CTGGCA),459(CTGCCA),55(CTGCCA),667(CTGACA),195(CTGACA),9(CTGGCT),97(CTGCCA),215(CTGGCA),97(CCGGCA),830(CTGGCA),21(CGGGCA),351(CTGGCA),192(CTGGAA),33(CTGGCA),756(CTGGCA),107(CTGGCA),149(CTGGCA),288(CTGGCA),74(GGGGCA),97(CTGTCA),167(CTGGGA),142(ATGGCA),282(CTGGCA),222(CTGGCA),596(CTGGCA),489(CTGGCA),35(CTGGCA),182(CTGGCG),22(CCGTCA),408(CTGGCA),111(CTGGCA),60(CTGGCG),80(CTGGCC),8(CTGGGA),408(CTGGCA),399(CTGGCA),39(CTGGCA),207(CTGGGA),368(CTGGCA),59(CTGGAA),815(CTGGCA),426(CTGGCA),259(GTGGCA),114(CTGGCA),39(CTGTTA),72(CTGGTA),0(ATGGCA),208(CAGGCA),946(CTGGCA),464(CTGGCA),207(CTGGCA),37(CTGCCA),602(CTGGCA),464(CTGGCA),132(CTGGAA),294(CTGGCT),501(CTGGCA),215(CTGGCA),39(CTGGCT),812(CTGGCA),185(CTGGCA),189(CTGTCA),26(CTGGCT),45(CTGGCA),17(CTGGCG),240(CTGGCA),97(CTGACA),42(CTGGCT),60(CTGGTA),150(CTGGCA),306(CTGGCA),30(CTGGAA),42(CTGCCA),378(CTGGCA),31(CTGGAA),102(CTGGAA),864(CTGGCA),361(CTGGCA),261(CTGGCA),30(CTGGCG),46(CTGGTA),141(CTGGCA),71(GTGGCA),45(CAGGCA),138(GTGGCA),141(CTGGCG),163(CTGGCG),29(CTGGTA),195(CTGGAA),10(TTAGCA),462(CTGGCA),618(CTGGCA),309(CTGGCA),200(CTGGCA),197(CTGGCA),570(CTGGCA),228(CTGGCA),264(CTGGCA),24(CCGGCA),185(CTGGCA),30(CTGGCG),553(CTGGCA),81(CTGGCC),123(CTGGTA),69(CTGGCG),35(CTGGCA),380(CGGGCA),248(CTGGCA),647(GTGGCA),121(CTGGCG),5(CCGGCA),427(CTGGCA),1986(CTGGCA),197(CTGGCA),61(ATGGCA),627(CTGGCA),31(CCGGCA),75(CTGGCA),91(TTGGCA),31(CCGCCA),498(CTGGCA),1503(CTGGCA),100(ATGTCA),195(CTGGAA),229(CTGACA),21(CTGCCA),258(CTGGCA),540(CTGGCA),71(CTGGGA),200(CTGGCT),150(CAGGCA),636(CTGGCA),106(GTGGCA),231(CTGGAA),99(CTGGTA),179(CTGGCA),332(CTGGCA),137(CTGGGA),79(CTGGGA),146(CTGGCT),44(CGGGCA),483(CTGGCA),1121(CTGGCA),66(CTGGCA),578(CTGGCA),324(CTGGCA),114(CTGGAA),197(CTGGCA),981(CTGGCA),39(CTGGCG),449(CTGGCA),25(TTGGCA),308(CTGGCA),167(CTGGCA),626(CTGGCA),137(CTGGCA),545(CTGGCA),84(CTGGCA),336(CTGGCA),222(CTGGCA),456(CTGGCA),912(CTGGCA),57(CAGGCA),4(CTCACA),15(GTGGTA),869(CTGGCA),107(CTGGCT),870(CTGGCA),373(CAGGCA),19(CTGGAA),801(CTGGCA),371(CTGGCA),70(ATGGCA),111(CTGCCA),96(CTGGCG),747(CTGGCA),549(CTGGCA),559(CTGGCA),819(CTGGCA),270(CTGGCA),641(CTGGCA),1347(CTGGCA),64(CTGGCG),234(CTGGCA),567(CTGGCA),208(CTGGCG),909(CTGGCA),314(CTGGCA),1553(CTGGCA),210(CTGGAA),0(ATGTCA),112(CTGGCG),31(TTGGCA),466(CTGGCA),23(CTGGCA),477(CTGGCA),46(TTGGCA),240(CTGGCA),157(CTGGAA),62(CTGGCA),939(CTGGCA),117(CTGGCA),560(CTGGCA),460(CTGGCA),957(CTGGCA),983(CTGGCA),330(CTGGCA),164(CTGGCA),153(CTGTCA),17(GTGGCA),111(CTGGTA),302(CTGGCA),0(ATGGCA),283(GTGGCA),84(CTGGCA),12(CTGGAA),590(CTGGCA),20(CTGTCA),149(CGGGCA),137(CTGGAA),429(CTGGCA),201(CTGGCA),558(CTGGCA),64(CTGGCA),45(CTGTCA),408(CCGGCA),540(CTGGCA),676(CTGGCA),208(CTGGCA),39(CTGGCA),33(CTGGCC),1098(CTGGCA),257(CTGGCA),108(CTGGCG),139(CTGGCG),135(CTGGCA),1436(CTGGCA),296(CTGGCA),57(CTGGCG),1119(CTGGCA),142(CTGGTA),336(CTGGCA),699(CTGGCA),990(CTGGCA),141(CTGGCA),267(CTGGAA),96(CTGGTA),194(CTGGCA),237(CTGGCA),32(CTGGCA),1371(CTGGCA),8(CTGGCA),62(CCGGCA),724(ATGGCA),258(CTGGCA),49(CTGGCC),0(ATGGCA),105(CTGGCA),264(CTGGCA),159(CTGGCA),99(GTGGCA),218(CTGGAA),111(CTGGCA),14(CTGGGA),205(CCGGCA),39(CTGGCA),15(CTGGCT),133(CTGTCA),232(CTGGCG),158(CTGGTA),162(CTGGCG),60(CTGGCC),195(CTGGCA),1358(CTGGCA),234(CTGGCT),783(CTGGCA),56(CTGCCA),136(CTGTCA),527(CTGGCA),354(CTGGCA),283(CTGACA),324(CTGGCA),51(CTGGCG),127(CTGTCA),702(CTGGCA),81(CTGGTA),24(GTAGCA),51(CTGGCA),48(CTGGCA),213(CTGTCA),166(CTGGCA),290(CTGGCC),63(CGGGCA),66(CAGGCA),365(CTGGCA),422(CTGGCA),128(CTGGCA),285(CTGGCA),308(CTGGCA),130(CGGGCA),45(CTTGCA),474(CTGGAA),213(CTGGCA),141(CTGGCA),195(CTGGCA),69(CTGGCA),171(CTGGCA),1448(CTGGCA),21(CTGCCA),648(CTGGCA),240(CTGGCA),54(CTGGAA),78(CTGGCA),63(CTGGCA),68(CTGGCA),38(CGGGCA),203(CTGGCA),162(CAGGCA),7(CTCGTA),11(TTGGCC),472(CTGGCA),821(CTGGCA),201(CTGGCA),91(CTGACA),128(GTGGCA),105(CTGGCA),8(CAGGCA),830(CTGGCA),515(CTGGCA),108(CTTGCA),879(CTGGCA),457(CTGGCA),707(CTGGCA),705(CTGGCA),250(CTGGAA),66(CTGGCC),296(CTGGCA),79(GCGGCA),94(CTGGAA),486(CTGGCA),246(CTGGCA),531(CTGGCA),31(CTGGCG),462(CTGGCA),459(CTGGCA),369(CTGGCA),23(CTGGGA),843(CTGGCA),111(CTGGCT),15(CTGGCA),222(CTGGCA),195(CTGGCA),57(CTGGCA),219(CTGGCA),1332(CTGGCA),161(CTGGCA),187(CTGGCA),153(CTGGCC),11(CCGTCA),468(CTGGCA),9(CTGGCA),519(CTGGCA),274(CTGGCA),199(CTGGGA),378(CTGGCA),196(ATGGCA),672(CTGGCA),259(CTGGCA),225(CTGGCA),187(CTGGCA),170(CTGGAA),42(CTGGAA),225(CTGGCA),878(CTGGCA),190(CAGGCA),0(ATGGCA),69(CTGGCA),404(CTGGCA),30(CTGGCG),101(CTGGCA),61(CTGGCT),118(CGGGCA),45(CAGGCA)

median string: CTGGCG (total\_distance = 348)

motif consensus string: CTGGCG (consensus\_score = 3354)

motif positions/string s=(s1,s2,..,st-1,st):

475(CTGGCG),525(CTGGCG),114(CTGGCA),60(CTGGGG),19(CGGGCG),17(CAGGCG),133(TTGACG),37(CTGACG),94(CTGGCA),66(CTGGAG),25(CTAGCG),564(CTTGCG),25(CTGCCG),171(CTGTCG),19(CGGGCG),66(CTGGCG),66(CTGGAG),27(CAGGGG),241(ATGGCG),173(CTGGTG),69(CTGGCG),128(CTGGGG),64(GTGGCG),40(CTGCCG),188(CTGGCG),349(CTGGCA),42(CTTGCA),80(CTGGGG),155(CTGGCA),36(CTGGGG),135(CTGGCC),663(CTGGCG),71(CTGGCA),281(CTGGCG),13(CTGCCG),1158(CTGGCG),20(CTGGCT),81(CTGGTG),27(CTGCCG),176(CTGGCC),54(CTGGGG),126(CTGCCG),414(CTGGCG),84(CTGGCG),151(CCGGCG),180(CTGGCG),0(GTGGCG),780(CTGGCG),298(CCGGCG),62(TTTGCG),423(CTGGCG),66(CTGGCG),60(CTGGCC),21(CTGGCG),94(GTGGCG),725(CTGGCG),72(CTGGCG),26(GTGCCG),221(CTGGCG),146(CTGGCT),161(CTGGCG),600(CTGGCG),48(CTGGCA),1385(CTGGCG),1610(CTGGCG),202(CTGGCG),72(ATGGTG),43(CTGGCG),21(CTGGCG),21(CTGGCA),149(CTGGCG),78(CTGGAG),137(CTGGGG),18(CTGGTG),146(CTGGCT),169(CTGGCC),447(CTGGGG),162(CTGGGG),40(CTGGCT),1434(CTGGCG),9(CTGTCG),453(CTGACG),67(CTGTCG),22(CAGGCG),33(TTGACG),54(CTGGCT),0(GTGGCG),114(CTGGGG),593(CTGGCG),445(CTGACG),261(CTGGAG),357(CAGGCG),51(CTGGCT),140(CTGGCA),11(CTTTCT),87(CTGGTG),128(CTGGCA),206(CTGGCG),258(CTGGCG),309(CTGGCT),987(CTGGCG),73(ATGGCG),120(CTGGCG),33(CTGTCG),135(CTTGCG),121(CTGGGG),12(ATGGTG),431(CTGGCG),936(CTGGCG),9(CTGGCT),202(CTGGCG),53(CTGGTG),168(CTGGCG),127(CTGGCG),36(GTGGCG),153(CTGGCG),57(CTGGCG),111(CTGGCG),116(CTGGAG),126(GTGGCG),41(CTGGCG),81(CCGGCG),27(CTCGCG),551(CTGGCG),132(CTGGCG),11(CAGGCG),42(CTGGCG),119(CTGGCG),206(CTGGCG),87(CTGGCG),222(CTGGCG),102(CTGGCG),79(CTGGCG),75(CAGGCG),18(CTGGCA),81(CCGGCG),171(CTGGGG),6(CTGGCT),20(CTGGCA),73(CTGGCA),67(CTGATG),48(CTGGCT),13(CTGATG),25(GTGTCG),111(CTGGTG),278(CTGGCG),120(CTGGAG),184(CTGGCG),273(CTGGCA),195(CTGGCT),549(CTGGCG),28(CTGCCG),33(CAGGCG),30(CTGGCG),968(CTGGCG),56(CTGGGG),22(CTGCCC),300(CTGCCG),24(CTGGCT),98(CTGGCA),189(CTGGCG),15(CTGACG),63(CTGGCG),633(CTGGCG),118(CTGGTG),42(CTGGGG),514(CTGGCG),180(CTGGCG),924(CTGGCG),506(CTGGCG),108(CTGGCG),126(CTGGCG),38(CTGGTG),82(CCGGCG),51(CCGGCG),0(ATGTCG),56(TTGGCG),41(CTGGAG),230(ATGGCG),24(CTGGCC),48(CTGACG),57(CAGGCA),102(CTGGCG),119(CTGGCG),567(CTGGCG),304(CTGGGG),60(CAGGCT),67(CTGGCA),317(CTGGCG),233(CTGGCG),93(CTGGCG),45(CAGGCT),60(CTGGCA),231(CGGGCG),221(CTGGCG),62(TTTGCG),95(CTGGCT),36(GTGGCG),171(CTGGCG),453(CTGGCG),126(CAGGCG),193(CTGGCG),40(TTAGCG),152(GTGGCG),344(CGGGCG),312(CTGGCG),212(CTGGCG),108(CTGACG),221(CTGGCG),346(CTGGCG),54(CTGGAG),16(CGAGCG),45(CTGCCG),76(CTGACG),212(CTGGCG),154(CTGCCG),126(CTGGCC),1688(CTGGCG),104(CTGGTG),17(GTGGCG),378(CTGGCG),146(CTGGCA),67(CTGCCG),45(CTGGCC),30(CAGGCG),1032(CTGGCG),835(CTGGCA),83(CTGGCA),327(CTGGCG),134(CTGGAG),11(CTGCCG),701(CTGGCG),321(CTGGCT),593(CTGGCG),231(CTGGCG),126(CTGGGG),45(CTGGTG),190(CTGTCG),139(CTGGTG),618(CTGGCG),202(CTGGCG),63(CAGGCG),137(CTGTCG),53(CTGGTG),168(CTGGCG),113(CGGGCG),36(GTGGCG),153(CTGGCG),57(CTGGCG),133(CTGGCG),111(CTGGCG),116(CTGGAG),126(GTGGCG),149(CTGGCG),264(CAGGCG),155(CTGGCG),87(CTGGCG),348(CTGGCG),614(CTGGCG),264(CTGGTG),35(CTGGCA),182(CTGGCG),185(GTGGCG),402(CTCGCG),141(CTGGCG),60(CTGGCG),42(GTGGCG),23(CTGGCG),20(CTGCCG),135(CTGGCG),39(CTGGCA),73(CTGTTG),195(GTGGCG),1681(CTGGCG),56(CTGGCC),976(CTGGCG),41(CAGGCG),114(CTGGCA),96(CTGGAG),26(CCAGCG),141(CTTGCG),38(CTGGTG),117(CTGCCG),539(CTGGCG),207(CTGGCA),26(CTGCCG),489(CTGGCG),639(CTGGCG),8(CTGGAG),18(ATGGCG),304(CTGGCG),202(CTGGCG),39(CTGGCT),33(CTGGTG),309(CTGGCG),71(CTGTCG),1571(CTGGCG),45(CTGGCA),17(CTGGCG),74(CTGGTG),428(CTGGCG),42(CTGGCT),471(CTGGCC),1096(CTGGCG),141(CTGGGG),37(CTGGCG),180(GTGGCG),130(CTGCCG),31(CTGGAA),192(CTGGCC),91(CTGACG),412(CTGGCG),88(CTGTCG),30(CTGGCG),204(CTGGCT),133(CTGGCG),38(TTGGTG),5(CTGGGC),200(CTGGCG),141(CTGGCG),163(CTGGCG),36(CTTGCG),134(CTGTCG),58(CGGGCT),258(CTGGCG),522(CTGGCG),657(CTGGCG),149(CTGGCG),165(CTGGCG),957(CTGGCG),753(CTGGCG),150(CAGGCG),411(CTGGCG),185(CTGGCA),30(CTGGCG),70(CTCGCG),880(CTGGCG),114(GTGGCG),69(CTGGCG),35(CTGGCA),5(CTGTGG),8(CTGGCG),852(CTGGAG),121(CTGGCG),25(ATGGCG),516(CTGGCG),84(CTGGCG),32(CTGGCT),328(CTGGCG),690(CTGGCG),312(CTGGCG),66(CAGGCG),91(TTGGCA),49(TTAGCG),751(CTGGCG),85(CTGGCG),70(CCGTCG),144(CTGTCG),97(TTGGCG),441(CTGGCG),1012(CTGGCG),897(CTGGCG),135(CTGGCG),1287(CTGGCG),285(CTGCCG),282(CTGGCG),954(CTGGCG),105(CTGGAG),512(CTGGCG),160(CTCGCG),363(CTGGCG),73(CTGCCG),210(CTGGCG),226(CTGGCG),141(CTGGCG),300(CTGGCG),477(CTGGCG),1768(CTGGCG),565(CTGGCG),849(CTGGCG),504(CTGGCG),589(CTGGCG),32(CTGGCG),39(CTGGCG),141(CTGGTG),303(CTGGCG),450(CTGGCG),84(CTGGCG),23(CTGGCT),267(CTGGCG),449(CTGGCG),219(CTGGCG),378(CTGGCG),137(CTGGCG),1230(CTGGCG),660(CTGGCG),450(CTGGCG),50(TTTGCG),13(CTGTGG),238(CTCGCG),107(CTGGCT),180(CTGGCG),37(CAGGCG),19(CTGGAA),24(CTGGCG),88(CTGCCG),153(CTGGCG),73(CTGGTG),96(CTGGCG),288(CTGGCG),175(CTGGCG),810(CTGGCG),228(CTGGCG),282(CTGGCG),66(CTGGCG),1234(CTGGCG),64(CTGGCG),72(CTGGCG),240(CTGGCG),208(CTGGCG),825(CTGGCG),95(CTGGCG),402(CTGGCG),84(ATGGCG),148(GTGGCG),112(CTGGCG),160(TTGGCG),732(CTGGCG),17(CTGGCG),609(CTGGCG),33(CTGATG),70(CCGGCG),366(CTGGCG),328(CTGGCG),969(CTGGCG),405(CTGGCG),1843(CTGGCG),610(CTGGCG),853(CTGGCG),27(CTGGCG),163(CTGGCG),244(CTGGCG),82(CTGCCG),550(CTGGCG),429(CTGGCC),1303(CTGGCG),326(CTGGCG),123(CTGGTG),51(CTGGTG),412(CTGGCG),18(GTGGCG),104(CTGGCG),378(CTGGCG),299(CTGGCG),729(CTGGCG),465(CTGGCG),582(CTGGCG),64(CTGGCA),271(CTGGCG),184(TTGGCG),337(CTGGTG),1072(CTGGCG),187(CTGGCG),196(CTGGCG),405(CTGGCG),1059(CTGGCG),687(CTGGCG),108(CTGGCG),139(CTGGCG),303(CTGGCG),168(CTGGCG),424(CTGGCG),57(CTGGCG),532(CTGGCG),417(CTGGCG),147(CTGGCG),36(CTGGCG),46(CTGGCG),264(CTGGCG),18(CTGCCG),291(CTGGCG),251(CTGGCG),471(CTGGCG),32(CTGGCA),903(CTGGCG),918(CTGGCG),318(CTGGAG),118(CCGGCG),667(CTGGCG),402(CTGGCG),34(CTGGTG),46(CTGGCG),195(CTGGCG),159(CTGGCA),801(CTGGCG),47(CTGCCG),335(CTGGCG),143(CTGGCG),654(CTGGCG),39(CTGGCA),935(CTGGCG),228(CTGGCG),232(CTGGCG),177(CTGGCG),162(CTGGCG),114(CTGGCG),268(CTGGCG),741(CTGGCG),141(TTGGCG),375(CTGGCG),169(CTGGCG),112(CTGCCG),84(CTGGCG),73(CGGGCG),103(CTGCCG),849(CTGGCG),51(CTGGCG),325(CTGGCG),636(CTGGCG),209(CTGGCG),67(CTGTCG),0(ATGGCG),31(GTGGCG),705(CTGGCG),234(CTGGCG),540(CTGGCG),120(CTGGAG),375(CTGGCG),105(CTGGCG),400(CTGGCG),128(CTGGCA),25(CTGTCG),55(CTGGGG),525(CTGGCG),174(CTGGCC),491(CTGGCG),76(CTGTCG),147(CTGGCG),99(CTGGCG),225(CTGGCG),19(CTGCCG),303(CTGGCG),675(CTGGCG),165(CTGGCG),18(CTGGCC),957(CTGGCG),446(CTGGCG),63(CTGGCA),68(CTGGCA),293(CTGGCG),203(CTGGCA),31(CTCGCG),69(ATGGCG),11(TTGGCC),494(CTGGCG),836(CTGGCG),485(CTGGCG),318(CTGGAG),417(CTGGGG),105(CTGGCA),113(CTGGCT),854(CTGGCG),398(CTGGCG),78(ATGGCG),477(CTGGCG),48(CTGGCT),483(CTGGCG),21(CTTGCG),23(CTCGGG),199(CTGGCG),306(CTGGCG),174(ATGGCG),0(ATGACG),276(CTGGCG),102(CTGTCG),91(TTGGCG),31(CTGGCG),624(CTGGCG),60(CTGGAG),702(CTGGCG),135(CTGGCG),170(CTGGCG),111(CTGGCT),450(CTGGCG),462(CTGGCG),870(CTGGCG),110(CTGGCG),10(CTGCCG),1009(CTGGCG),28(CTTGCG),375(CTGGCG),252(CTGGCG),39(CTCGCT),27(CTGGGG),461(CTGGCG),78(CTGGCT),718(CTGGCG),165(CTGGAG),378(CTGGCA),360(CTGGCG),54(CTGGCG),63(CTGTCG),161(CTGGCT),96(CTTGCG),344(CTGGCC),90(CTGGTG),225(CTGGCA),312(CTGACG),260(GTGGCG),0(ATGGCA),69(CTGGCA),10(CTGACG),30(CTGGCG),477(CTGGCG),61(CTGGCT),183(CTGACG),143(TTGGCG)

median string: CTGGAA (total\_distance = 353)

motif consensus string: CTGGAA (consensus\_score = 3349)

motif positions/string s=(s1,s2,..,st-1,st):

63(CAGGAA),206(CTGGAA),231(CTGGAA),1039(CTGGAA),351(CTGGAA),106(CGGGCA),198(CTGGAA),574(CTGGAA),43(ATGGAA),9(CAGGAA),99(CTGGCA),200(CTGGAA),63(CGGGCA),1039(CTGGAA),351(CTGGAA),42(CTGGAA),9(CAGGAA),43(ATGGAA),347(CTGGAA),42(CTTGAA),158(CTGGCA),92(GTGGAA),240(CTGGAA),23(CTGGAA),90(CAGGAA),110(CTGGAT),159(CTGGAA),11(GTGGAA),38(CTGGGA),87(CTGCAA),206(CTGGAA),951(CTGGAA),71(CTGGCA),60(CAGGAA),302(CTGGAA),159(CTGCAA),182(CTGGAA),72(CTGGAA),78(CTGGAT),657(CTGGAA),273(ATGGAA),41(CGGGGA),35(CTGGAA),1553(CTGGAA),77(CTGTAA),129(CTGGCA),39(CTTGAA),159(CTGAAA),187(CGGGAA),73(GTGAAA),2093(CTGGAA),54(CTGGGA),432(CTGGAA),46(CTGGTA),146(CTGGCA),59(CTGGAA),21(CTGAAA),29(CCGGTA),746(CTGGAA),225(CTGGAG),17(CTTGAA),222(CTGGCA),48(CTGGCA),1019(CTGGAA),132(CTGGAA),126(ATGGAA),184(CCGGAA),303(CTGGAA),93(CAGGAA),21(CTGGCA),1038(CTGGAA),396(CTGGAA),60(CTGGAT),0(GTGGAA),225(CTGGAG),73(CTGCAG),147(CTGGGA),338(TTGGAA),147(CTAGAA),1904(CTGGAA),65(CCGGAA),136(CCGGAA),130(GTGGAA),82(CTGTAA),112(CTGTAA),0(TTGGAA),6(CTTAAA),342(CTGCAA),1523(CTGGAA),114(CAGGAA),64(CTGCAA),15(CTGCAA),507(CTGGAA),66(CGGGAA),0(GTGAAA),16(CTGATA),78(CTGGTA),7(CGAGAA),105(CTGGCA),189(CTGAAA),885(CTGGAA),942(CTGGAA),230(CAGGAA),332(CTGGAA),465(CTGGAA),216(CAGGAA),620(CTGGAA),48(CTGGTA),130(CTGAAA),132(GTGGAA),117(CAGGAA),879(CTGGAA),21(CGGGAA),218(CTGGAA),192(CTGGAA),33(CTGGCA),72(CCGGAA),129(CCGGAA),138(CTGGAA),13(CAGAAA),33(CTGGCA),141(CCGGAA),214(CTGAAA),30(CCGGAA),162(CTGGAA),70(CTGGAA),219(CTGGCA),42(CTGGGA),215(CTGGAA),846(CTGGAA),45(CAGGAA),531(CTGGAA),56(CTGGAA),13(CCGGAT),51(CTGGAA),141(CCGGAA),192(CTGGAA),282(CAGGAA),20(CTGGCA),37(CAGGAA),155(CTGTAA),207(CTGGAA),0(TTGGAT),111(CTGGAA),4(CTGAAA),12(GTGGAA),120(CTGGAG),428(CTGGGA),89(CTGGTA),77(CTGCAA),41(CTGGAA),39(CCGGAA),42(CTGGAA),111(CTGGAG),49(CCGGAA),30(CTGAAA),36(CTGGAT),2(GTGGAA),1127(CTGGAA),98(CTGGCA),254(CTGGAA),60(CTGGAA),1236(CTGGAA),693(CTGAAA),77(CTGGAA),102(CTGGAA),162(CTGGTA),18(CTGGCA),245(CTGGAA),120(CTGGAA),0(ATGGAA),312(CTGGAA),75(CTGGAA),170(CCGGAA),18(CCGGAA),98(CCGGAA),138(CCGGAA),16(CTGAAA),61(CTGCAA),14(CTGGAC),16(CTGAAA),163(ATGGAA),47(CAGGAA),23(CTGGTA),387(CTGGAA),109(CTGGTA),53(CGGGAA),67(CTGGCA),3(CTGGAC),489(CTGGAA),3(CTGCAA),51(CTTGAA),807(CTGGAA),28(CTGCAA),33(CTGGAA),73(GTGAAA),81(CCGGAA),114(CTGGAT),224(CTGGGA),504(CTGGAA),96(CTGGAA),0(ATGGAA),206(ATGGAA),7(CAGGAA),328(CCGGAA),36(GTGGAA),74(CTGGTA),23(CTGGAA),39(CTCGAA),45(CTGGAT),54(CTGGAG),104(CTGGAA),126(CTGGCA),109(CGGGAA),74(CTGGTA),12(CAGGAA),649(CTGGAA),660(CTGGAA),194(CTGGAA),101(CTGGCA),507(CTGGAA),6(CAGGAA),3(ATGAAA),208(CTGAAA),353(CTGGAA),383(CTGGAA),268(CTGGGA),83(CTGGCA),495(CTGGAA),49(CTGAAA),56(CTGGCA),332(CTGGAA),111(CTTGAA),30(CTGGAA),31(CAGGAA),96(CTGGAT),28(CTGAAA),96(CCGGAA),288(CTGGAA),711(CTGGAA),117(CAGGAA),204(CTGGAA),6(CTGTAT),879(CTGGAA),21(CGGGCA),218(CTGGAA),192(CTGGAA),33(CTGGCA),72(CCGGAA),68(CGGGAA),129(CCGGAA),138(CTGGAA),13(CAGAAA),62(CTGTAT),167(CTGGGA),164(CTGGAA),846(CTGGAA),45(CAGGAA),531(CTGGAA),387(CTGGAA),28(TTGGAA),99(CCGGAA),115(CTGGAC),83(CTGGAA),111(CTGGCA),90(CTGGAA),36(CTTGAA),213(CTGGAA),193(CTGGTA),126(ATGGAA),39(CTGGCA),92(CAGGAA),1116(CTGGAA),59(CTGGAA),114(CTGGTA),599(CTGGAA),76(CTGTAA),114(CTGGCA),96(CTGGAG),207(CTGGAA),191(CTGGAA),462(CTGGAA),930(CTGGAA),144(CTGGAA),43(ATGGAA),358(CCGGAA),258(CTGGAA),63(CAGGAA),132(CTGGAA),261(CTGAAA),55(CCGGAA),68(CTGGAA),761(CTGGAA),905(CTGGAA),48(CCGGAA),258(CTGGAA),234(CTGGAA),521(CTGGAA),116(CTGGAT),1071(CTGGAA),158(CTGGAA),50(CTGCAA),108(CTGGAA),618(CTGGAA),45(CTGGAC),30(CTGGAA),125(CTGGTA),410(CTGGAA),31(CTGGAA),102(CTGGAA),792(CTGGAA),93(CTTGAA),468(CTGGAA),998(CTGGAA),46(CTGGTA),486(CTGGAA),417(CTGGAA),206(CTGCAA),50(CTGCAA),20(CTGGAT),504(CTGGAA),471(CTGGAA),195(CTGGAA),0(ATGAAA),69(CTGGAA),255(CTGGAA),213(CTGGAA),134(CTGGGA),384(CTGGAA),987(CTGGAA),246(CTGGAA),81(CTGGAA),99(CTGGAC),351(CTGGAA),287(CTGGAT),486(CTGGAA),420(CTGGAA),123(CTGGTA),228(CTGGAA),123(CTGGAA),34(CTGGAC),436(CTGGAA),1206(CTGGAA),40(CTGAAA),5(CCGGCA),177(CTGGAA),1050(CTGGAA),456(CTGGAA),420(CTGGAA),876(CTGGAA),164(CTGGAA),75(CTGGCA),3(CATGAA),64(CCAGAA),51(CTGGAA),420(CTGGAA),9(TTTGAA),195(CTGGAA),470(CTGGAA),147(CTGGAA),1194(CTGGAA),981(CTGGAA),24(CAGGAA),1502(CTGGAA),486(CTGGAA),155(CTGGAA),162(CTGGAA),231(CTGGAA),84(CTGGAT),78(CTGGAA),699(CTGGAA),45(ATGGAA),267(CTGGAA),64(CTGATA),519(CTGGAA),483(CTGGCA),8(CTGGAA),593(CTGGAA),606(CTGGAA),591(CTGGAA),114(CTGGAA),795(CTGGAA),507(CTGGAA),567(CTGGAA),759(CTGGAA),990(CTGGAA),117(CTGGGA),12(CTGGAA),567(GTGGAA),483(CTGGAA),168(CTGGAA),489(CTGGAA),786(CTGGAA),441(CTGGAA),782(CTGGAA),348(CTGGAA),720(CTGGAA),4(CTCACA),0(ATGGAT),477(CAGGAA),0(GTGGGA),1569(CTGGAA),24(CTCGAA),19(CTGGAA),611(CTGGAA),66(CGGGAA),729(CTGGAA),213(CTGGAG),326(CTGGAA),1092(CTGGAA),564(CTGGAA),312(CTGGTA),468(CTGGAA),749(CTGGAA),866(CTGGAA),1218(CTGGAA),684(CTGGAA),61(ATGGAA),24(CTGGTA),273(CTGGAA),128(CTTGAA),267(CTGGAA),159(CTGGAA),210(CTGGAA),35(CGGGAA),150(CTGGAT),184(ATGGAA),90(CTGGAA),1095(CTGGAA),276(CTGGAA),171(CTGGAA),45(CTGGAA),157(CTGGAA),62(CTGGCA),287(CTGGAA),318(CTGGAA),225(CTGGAA),1308(CTGGAA),87(CTGGAA),93(GTGGAA),1734(CTGGAA),234(CTGGAA),357(CCGGAA),144(CTGGAT),22(CGGGAA),312(CTGGAA),119(CTGGTA),267(CTGGAT),54(GTGGAA),12(CTGGAA),285(CTGGTA),48(CTGGAT),320(CTGGAA),137(CTGGAA),78(CTGGAG),351(CTGGAA),537(CTGGAA),64(CTGGCA),205(CTGAAA),202(CTGAAA),113(CTGTAA),1343(CTGGAA),138(CTGGAA),39(CTGGCA),45(CTGGAG),612(CCGGAA),219(CTGGAA),129(CTGGAA),0(ATGGAA),135(CTGGCA),330(CTGGAA),5(CGGGAA),17(CGAGAA),474(CTGGAA),222(CTGGAA),120(CTGGAG),71(CTGGAT),1005(CTGGAA),345(CTGGAA),267(CTGGAA),513(CTGGAA),210(CTGGAA),24(ATGGAA),279(CTGGAA),1131(CTGGAA),249(CTGGAA),363(CTGGAA),111(CTGGAT),405(CTGGAA),72(CTGGAG),65(CTGGAA),219(CTGGAA),576(CTGGAA),108(CTGGAT),138(CTGGAA),218(CTGGAA),111(CTGGCA),675(CTGGAA),408(CTGGAA),135(CTGGAA),219(CTGGTA),166(CTGGAA),39(GTGGAA),84(CTGAAA),510(CTGGAA),897(CTGGAA),42(CTGGAA),591(CTGGAA),93(CTGCAA),987(CTGGAA),87(CTGAAA),0(GTGGAA),507(CTGGAA),222(CAGGAA),402(CTGGAA),591(CTGGAA),170(CTGGAT),241(TTGGAA),147(CTGAAA),150(CTGGAA),112(CTGCAA),543(CTGGAA),48(CTGGCA),88(CTGGAC),166(CTGGCA),342(CTGGAA),120(CTGGAG),79(CTGGAG),600(CTGGAA),1277(CTGGAA),128(CTGGCA),84(CTGGAG),251(CTGGAA),7(CTGCAA),244(CTGTAA),474(CTGGAA),366(CTGGAA),122(CTGGAA),3078(CTGGAA),45(CTGGAA),141(CTGGAG),1437(CTGGAA),441(CTGGAA),30(CAGGAA),5(CAGGAA),54(CTGGAA),362(CTGGAA),135(CTGGAA),12(CGGGAA),381(CTGGAA),132(CTGGAT),357(CTGGAA),7(CTCGTA),22(ATGGAG),507(CTGGAA),24(CTGGAA),296(CTGGAA),145(CTGAAA),132(CAGGAA),105(CTGGCA),8(CAGGCA),95(CTTGAA),84(CTGGAA),438(CTGGAA),359(CTGGAA),370(CTGCAA),261(CTGGAA),218(CTGTAA),250(CTGGAA),4(CTGCTA),165(CTGAAA),162(CTGGAT),94(CTGGAA),1452(CTGGAA),506(CTGGAA),573(CTGGAA),143(CTGGAA),306(CTGGAT),479(CTGGAA),408(CTGGAA),23(CTGGGA),1085(CTGGAA),177(CTGGAA),15(CTGGCA),186(CCGGAA),3(CTGGAA),57(CTGGCA),278(CTGGAA),219(CAGGAA),119(CTGGGA),120(CTGGAC),57(CGGGAA),15(CAGGAA),147(CTGGTA),9(CTGGCA),12(CTGAAA),244(ATGGAA),411(CTGGAA),398(CTGGAA),66(ATGGAA),503(CTGGAA),107(CTGGTA),333(CTGGAA),187(CTGGCA),170(CTGGAA),42(CTGGAA),164(ATGGAA),135(CTTGAA),90(CTTGAA),0(ATGGCA),539(CTGGAA),27(CTTGAA),168(CTGGAA),125(CTGGAA),168(CAGGAA),9(CTGAAA),152(CTGGAA)

median string: TGCTGG (total\_distance = 353)

motif consensus string: TGCTGG (consensus\_score = 3349)

motif positions/string s=(s1,s2,..,st-1,st):

31(CGCTGG),434(TGCTGG),112(TGCTGG),962(TGCTGG),274(TGCTGG),52(TGTTGG),196(TGCTGG),187(TGATGG),20(TGCTTG),64(TTCTGG),97(TGCTGG),65(TTCTGG),9(TGTTGG),58(TGCTGG),274(TGCTGG),40(TACTGG),64(TGCTGG),20(TGCTCG),70(GGCTGG),78(GGCTGG),67(TCCTGG),47(TGCTGC),829(TGCTGG),7(GGCTGG),221(TGCTGT),526(TGCTGG),17(AGATGG),151(TGTTGG),144(TGCGGG),29(TCCTGA),133(TGCTGG),19(AGCTGG),31(TGCTTG),20(AGCTGG),14(TGCCGG),709(TGCTGG),9(GGCCGG),79(TTCTGG),25(TGCTGC),655(TGCTGG),31(TGCTCG),61(TGATGG),868(TGCTGG),772(TGCTGG),97(TGCCGG),127(TGCTGG),160(TGGTGG),181(TGCTGG),150(TGCTGT),123(TGCTGT),730(TGCTGG),514(TGCTGG),13(TGATGG),452(TGCTGG),89(TGATGG),52(TACTGG),70(TGCTGG),27(TGCCGG),529(TGCTGG),170(TGCCGG),109(TACTGG),462(TGCTGG),46(TGCTGG),128(TGTTGG),448(TTCTGG),222(TCCTGG),70(TGATGG),12(TTCTGG),19(CACTGG),19(TACTGG),632(TGCTGG),394(TGCTGG),58(ATCTGG),19(TGGTGG),170(TGCCGG),25(TCCCGG),89(TTCTGG),437(TCCTGG),406(TGCTGG),1306(TGCTGG),7(TGCTGT),59(TGCGGG),31(TGCTGC),332(TGCTGG),401(TGCTGT),52(TTCTGG),20(TATTGG),112(TTCTGG),320(TGCTGG),35(TGCTGT),62(TGCTGC),6(TACTGG),49(TGCTGG),130(TGCTGA),24(TCCTGC),85(TTCTGG),76(GGCTGG),148(TGCTGT),154(TGCTGG),1(TGTTGG),883(TGCTGG),61(TGCTTG),334(TGCTGG),13(TGCTGG),327(TACTGG),6(AGCTGG),491(TGATGG),107(TGCTGC),314(TGCTGG),1(TGTTTG),223(TGCTGG),322(TGCTGG),166(GGCTGG),397(TGCTGG),8(TGCCGG),67(TGCTGG),55(TGCTGG),109(CGCTGG),268(TGCTGG),148(TCCTGG),39(GGCTGG),93(TGCCGG),16(TGTTGC),901(TGCTGG),130(TGCTGG),68(CGCTGG),83(TGCTGG),331(TGCTGG),233(TGCCGG),85(TGCTGG),68(TGCCGG),100(CGCTGG),54(CGCTGG),1(TGCTAC),16(TGCTGG),93(TGCCGG),89(TGCTGT),95(TTCTGG),3(TGCCCG),71(TGCTGG),45(TCCTGC),103(TGCTGG),237(TGCTTG),77(TGCTGC),125(TGCTGC),89(TGTTGG),58(TGCTGT),172(TGCTGG),16(AGCTGG),16(TGATGG),313(TGCTGG),29(TGCCGG),40(TGCTGG),6(TGCGGG),58(TGCTGG),258(TGCTGG),34(TGCTGG),43(TGCTGT),22(TGCTGG),1(TGCTGT),49(TGCTGA),28(TGCTGA),61(TGCTGG),217(TGCTGG),1(TGCTGC),40(TTCTGG),160(TTCTGG),196(TGCTGG),373(TGCTGG),3454(TGCTGG),274(TGCTGG),124(AGCTGG),53(TGCCGG),80(TGCCGG),88(CGCTGG),17(TGCAGT),116(TGCGGG),538(TGCTGG),221(TGCTGC),22(TGCTGG),107(TGTTGG),161(TGATGG),1(TGCAGA),162(TGCTGG),16(TACTGG),992(TGCTGG),14(TGCCCG),83(TGATGG),1(TGCTGG),1210(TGCTGG),1(TGCTGC),173(TGATGG),637(TGCTGG),44(TGCAGG),7(GGCTGG),123(TGCTGT),67(TGCCGG),79(TACTGG),118(TGCTGA),535(TGCTGG),111(GGCTGG),191(TGCTGG),29(CGCTGA),61(AACTGG),268(TGATGG),71(TGCTGA),118(TGCTGG),21(AGCTGG),365(TGCTGG),344(TGCTGG),52(TGCTGG),35(TGCTGC),124(TGCTGG),58(TGCTGC),118(TGCTGG),37(TTGTGG),506(TGCTGG),40(CGCTGG),30(TGGTGG),61(GGCTGG),49(TGCTGG),136(TGCTGA),1(TGATGA),368(TGCTGG),304(TGCTGG),88(TGCTGA),206(TGCTGA),112(TGCTGG),280(TGCTGG),132(TACTGG),12(TGCCGG),712(TGCTGG),92(TGCTGC),123(TGCTGG),127(TGCTGG),65(TGTTGG),46(TGGTGG),17(TGATGG),137(TACTGG),1039(TGCTGG),223(TGCTGG),202(TGCTGG),15(TGCTGT),291(TGCTGG),166(GGCTGG),397(TGCTGG),8(TGCAGG),67(TGCTGG),55(TGCTGG),3(TGCTGC),109(CGCTGG),34(TGCTGG),148(TCCTGG),190(TGCTGG),235(TGCTGG),59(TGCTGA),85(TGCTGG),262(TGCTGG),55(TGCAGG),94(TGCTGA),388(TGCTGG),59(TGCAGG),71(TGCCGG),354(TACTGG),61(TGCGGG),580(TGCTGG),60(TGCCGG),6(GGCTGG),21(TGCCGG),622(TGCTGG),37(AGCTGG),274(TGCTTG),624(TGCTGG),175(TGCTGG),46(TGCAGG),53(TGTTGG),254(TGCTTG),112(TGCTGG),89(TGCTGT),129(TGGTGG),39(CGCTGT),39(TGGTGG),1646(TGCTGG),142(TGCTGG),142(TGCTGA),27(TGCCGG),483(TGCTGG),114(TGCTGC),102(TGCTGT),72(TGCCGG),302(TGCTGG),200(TGCTGG),5(TGCTGA),31(TCCTGG),476(TGCTGG),28(TGCTCG),157(TGCTGG),325(TGCTGG),15(TTCTGG),238(TTCTGG),41(TACTGG),17(TGCGGG),58(TCCTGG),41(TGCTGT),43(CGCTGG),1007(TGCTGG),514(TGCTGG),131(TGCCGG),127(TTCTGG),100(TGCTGG),236(TGCTGA),335(TGATGG),389(TGCTGA),28(AGCTGG),410(TGCTGG),131(TGCTGG),39(TGGTGT),3(TACTGG),31(TACTGG),43(TGCTGT),112(TGCTTG),156(TGCTGT),40(TGTTGG),56(TGCGGG),460(TGCTGG),201(TGCTGG),655(TGCTGG),40(TTCTGG),1(TGCTGA),568(TGCTGG),362(TGCTGG),502(TGCTGG),155(TGCTGG),197(TGCTGG),46(TGCTGT),76(TGCTGG),212(TGCTGC),1(TGCTGA),418(TGCTGG),241(TGCTGG),227(TGCGGG),195(TACTGG),56(TGATGT),119(AGCTGG),3(TGCCGG),685(TGCTGG),82(TGCTGG),454(TGCTGG),418(TGCTGG),10(TGTTGG),895(TGCTGG),415(TGCTGG),1(TGCATG),1(TGCATG),847(TGCTGG),1289(TGCTGG),40(TTGTGG),52(TGATGG),418(TGATGG),62(TGCAGG),246(TGCTGG),83(TGCCGG),1003(TGCTGG),711(TGCTGG),394(TGCTGG),49(TGCCGG),760(TGCTGG),103(TGCTGG),169(TGCTGG),499(TGCTGG),298(TGCTGG),463(TTCTGG),184(TGCTGG),133(TGTTGG),446(TGCAGG),454(TGCTGG),109(TGCTGG),1357(TGCTGG),625(TGCTGG),224(TGCTGG),502(TGCTGG),520(TGCTGG),538(TGCTGG),565(TGCTGG),757(TGCTGG),301(TGCTGG),448(TGCTGG),10(TGCTGG),121(TGCTGA),427(TGCTGG),235(TGCTGG),250(TGCTGG),55(TGCTGG),220(TGCTGG),550(TGCTGG),76(TGCTGG),94(CGCTGG),52(TGCGCG),11(TGCTGT),559(TGCTGG),38(TAATGG),211(TGCTGG),259(TGCTGG),17(ACCTGG),170(TGCTGG),325(TGCTGG),31(TGCTGG),211(TGCTGG),154(TGCTGG),376(TGCTGG),1063(TGCTGG),310(TGCTGG),466(TGCTGG),373(TGCTGG),496(TGCTGG),331(TGCTGG),775(TGCTGG),538(TGCTGG),701(TGCTGG),532(TGCTGG),242(TGCTGC),265(TGCTGG),175(TGCTGG),249(TTCTGG),33(TGCGGG),511(TGCTGG),340(TGCTGG),464(TGCTGG),247(TGCTGG),49(TGCTGG),34(TGATGG),43(TGCTGG),88(TGCTGG),19(TACTGG),161(TGCTGG),253(CGCTGG),854(TGCTGG),458(TGCTGG),532(TGCTGG),25(TGCTGG),265(TGCTGG),112(TGCTTG),238(TTCTGG),142(CGCTGG),160(TGATGG),217(TGCTGG),2131(TGCTGG),121(TGCTGG),49(TGCTGG),667(TGCTGG),283(TGCTGG),46(TACTGG),25(TGCTGA),13(TGCTGC),399(TGCTGG),349(TGCTGG),121(TGCTGG),62(TGCTGG),80(CGCTGG),257(TGCGGG),34(TGATGG),64(TGCTGT),163(TGCTTG),37(TGCTGG),43(CGCTGG),997(TGCTGG),121(TGCAGG),50(CGCTGG),370(TGCAGG),301(AGCTGG),166(TGCTGG),379(TGCTGG),55(CGCTGG),472(TGCTGG),478(TGCTGG),145(TGCTGG),34(TGCTGG),187(TGCTGG),262(TGCTGG),22(CGCTGG),229(TGCTGG),415(TGCTGG),43(TGCTGC),277(TGCTGG),1369(TGCTGG),154(TGCTGG),176(TGATGG),109(TGCTGG),445(TGCTGG),70(TGCTGG),32(GGCTGG),302(TGCTGG),241(TGCTGG),106(TGCTGG),311(TGCTGG),84(CGCTGG),109(TGCTGG),214(TGCTGG),31(TGCTGT),20(TGCAGG),56(TGATGG),85(TGCTGG),230(TGCTGG),550(TGCTGG),262(TGCTGG),226(TGCTGG),868(TGCTGG),1102(TGCTGG),1(TGCTGG),328(TGCTGG),73(TGCTGA),76(CGCTGG),130(TGCTGG),197(TGATGG),52(TGATGG),224(TGCTGG),168(GGCTGG),208(TGCTGT),634(TGCTGG),207(GGCTGG),1(TGCGTG),341(TGCTGG),22(TGCTGG),157(TGCTGG),457(TGCTGG),340(TGCTGG),118(AGCTGG),11(TGCCGG),232(TGCTGG),700(TGCTGG),193(TGGTGG),42(CGCTGG),47(TCCTGG),166(GGCTGG),43(TGCTTG),2(GGCTGG),211(TGCTGG),253(TGCTGG),115(TGCTGG),43(TGCTGG),169(TGCTGG),271(TGCTGG),346(TGCTGG),163(TGCTGG),238(TGCTGG),1051(TGCTGG),577(TGCTGG),61(TGCTGG),150(TGCCGG),484(TGCTGG),55(TGCAGG),355(TGCTGG),11(TACAGG),85(TGCTGA),115(TGCTGG),1215(TGCTGG),3337(TGCTGG),1(TGCTTC),16(TGCTTG),103(TTCTGG),6(TTCAGG),406(TGCTGG),280(TGCTGG),398(TGATGG),152(TGCTGG),343(TGCTGG),402(TGCTGG),61(TGCTGG),21(TACTCG),64(TACTGG),51(TGCTTG),160(CGCTGG),229(TGATGG),484(TGCTGG),406(TGCTGG),1042(TGCTGG),141(CGCTGG),185(TGTTGG),58(CGCTGG),688(TGCTGG),133(TACTGG),1045(TGCTGG),156(TTCTGG),28(TGATGG),220(TGCTGG),1(TGCTGG),68(TGCTGG),80(TGCAGG),860(TGCTGG),30(TGCGGG),373(TGCTGG),151(TGCTGG),124(TGCTGA),115(TGCTGG),253(TGCTGG),265(TTCTGG),418(TGCTGG),154(TTCTGG),61(CGCTGG),16(TGCTGG),884(TGCTGG),93(TTCTGG),163(GGCTGG),89(TGCTGT),124(TGCTGG),91(TGGTGG),193(TGCTGG),46(TGCTTG),228(TGCTGG),65(TGCTGT),67(TACTGG),334(TGCCGG),68(TGCTGG),28(CGCTGG),34(TGATGG),115(TGCCGG),170(TGCCGG)