Scoring Perfect Palindromes

- Example sequence: AAAGAGCTCTAA (green line highlights a perfect palindrome)

- Assign prime numbers with sign to nucleotides: A=3; T=-3; G=7; C=-7

- positional score vector :

A A A G A G C T C T A A
[3, 3, 3, 7, 3, 7, -7, -3, -7, -3, 3, 3]

- Cumulative score vector:

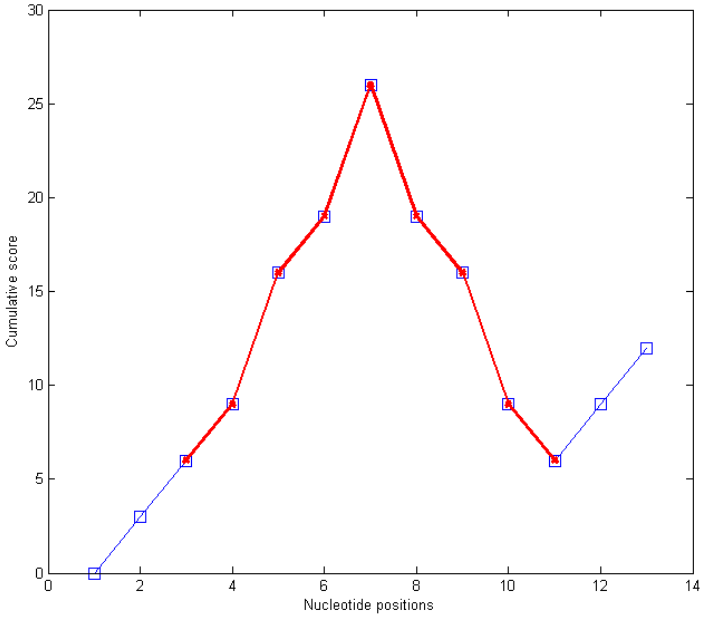
A A A G A G C T C T A A
[0, 3, 6, 9, 16, 19, 26, 19, 16, 9, 6, 9, 12]

(red line highlights a set of symmetric scores, each starting one base before a palindrome (above green lines) and terminating at the end of that palindrome)

- Presence of a perfect IR will not contribute to the cumulative score. Hence, potential palindromic sequences can be detected by searching for the presence of identical cumulative scores.
- The cumulative scores of perfect IRs will show a perfect symmetric pattern.

The cumulative scores for AAAGAGCTCTAA

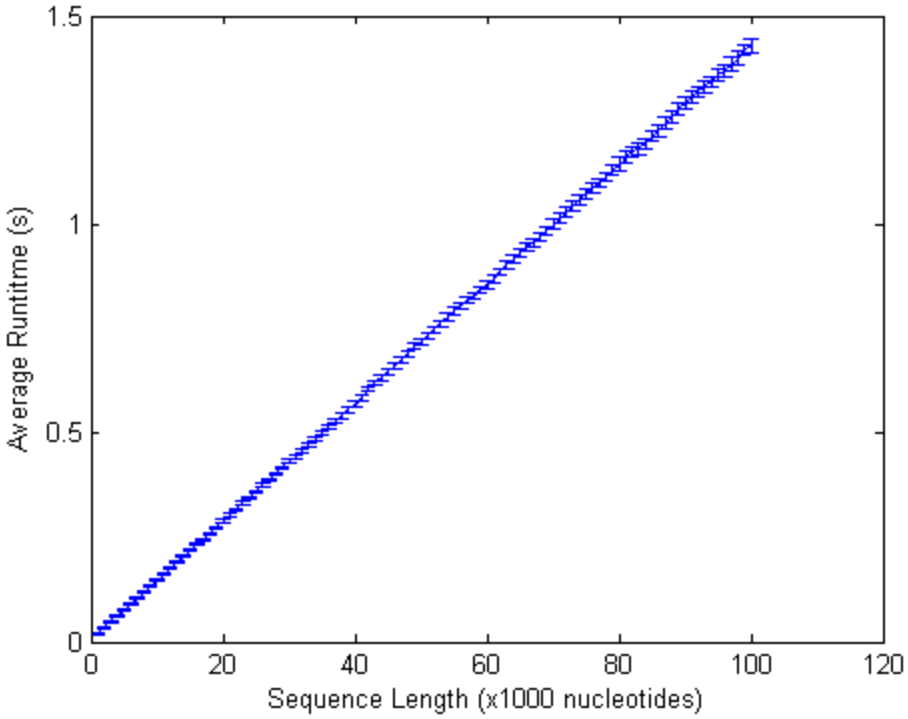
(The redline in the below graph highlights the symmetric scores)



Supplemental Figure S1: Graphic representation of scoring schema.

```
# Function definitions:
#    unique(A) - returns unique values of A
#    find(A == B) - returns the positions of A satisfying A == B

# Set up scoring system and calculate cumulate score
score[‘A’, ‘C’, ‘G’, ‘T’] = [10007, -10007, 10009, -10009]
scoreCumulativeVec[0] = 0
for i ← 1 to length(sequence):
    scoreCumulativeVec[i] = scoreCumulativeVec[i-1] + score[sequence[i]]
scoreCumulativeUniqueVec = unique(scoreCumulativeVec)

# Find the multiple occurrences of the unique cumulative scores
for each score in scoreCumulativeUniqueVec:
    positionVec = find(scoreCumulativeVec == score)
    initialIRposition.append(positionVec)

# Construct all the possible perfect inverted repeats
IRIniCount = 0, seedVec = [0]*length(sequence)
for each IRPositionVec in initialIRposition
   if length(IRPositionVec) > 1:
     for i=1 to length(IRPositionVec)-1:
        for j=i+1 to length(IRPositionVec):
            startPosition = IRPositionVec[i]
            endPosition = IRPositionVec[j] - 1
            IRIniLength = (endPosition - startPosition + 1)
            if IRIniLength <= maxLength and IRIniLength % 2 == 0:
                IRIniCount += 1
                startPositionVecIni[IRIniCount] = startPosition
                endPositionVecIni[IRIniCount] = endPosition
                lengthVecIni[IRIniCount] = IRIniLength
                seedPositionIni = startPosition + (IRIniLength / 2)
                seedVec[seedPositionIni] += 1
            else:
                break

# Validate the possible perfect inverted repeats
IRCount = 1, lengthIniSrtIndex = sort(lengthVecIni,'descend')
for each IRIndex in lengthIniSrtIndex:
    if lengthVecIni[IRIndex] >= minLength:
        seedPositionIni = startPositionVecIni[IRIndex] + (lengthVecIni[IRIndex] / 2);
        if (lengthVecIni[IRIndex] / 2) == seedVec[seedPositionIni]:
            startPositionVec[IRCount] = startPositionVecIni[IRIndex]
            endPositionVec[IRCount] = endPositionVecIni[IRIndex]
            IRCount += 1
            seedVec[seedPositionIni] -= 1
return startPositionVec,endPositionVec
```

Supplemental Figure S2: The pseudocode of our MATLAB program



Supplemental Figure S3. The average runtimes of our MATLAB program with different input sequence lengths. All these tests were performed in an Ubuntu Linux server (1,400 MHz, 96 GB RAM).

Supplemental Table S1. The comparison of perfect inverted repeats detected using tools provided by MATLAB (*palindromes* function), BioPHP and EMBOSS and our proposed algorithm/tool. The comparison is based on selected test cases, which indicates the inability of the other tools in detecting some simple perfect inverted repeat patterns.

Input Sequence	Matlab (*palindromes*)		BioPHP		EMBOSS		Proposed Algorithm	
	Starting Position	Palindromic Sequence	Starting Position	Palindromic Sequence	Starting Position	Palindromic Sequence	Starting Position	Palindromic Sequence
CATATATC	2	ATATAT	2	ATATAT	2	ATATAT	2	ATAT
	4	ATAT	3	TATA	2	ATAT	2	ATATAT
			4	ATAT	4	ATAT	3	TATA
							4	ATAT
AAATTTATA	1	AAATTT	1	AAATTT	1	AAATTT	1	AAATTT
	6	TATA	2	AATT	6	TATA	2	AATT
			6	TATA			6	TATA
ATATATGCGC	1	ATATAT	1	ATATAT	1	ATATAT	1	ATAT
	3	ATAT	2	TATA	1	ATAT	1	ATATAT
	7	GCGC	3	ATAT	3	ATAT	2	TATA
			7	GCGC	7	GCGC	3	ATAT
							7	GCGC