3. Gene prediction

- All genes end on a stop codon
- A simple algorithm for gene prediction
- Searching for start and stop codons
- **Predicting all the genes in a sequence**
- Making the predictions more reliable
- Boyer-Moore algorithm
- Index and suffix trees
- Probabilistic methods
- Benchmarking the prediction methods
- Gene prediction in eukaryotic genomes

```
indexStop1, indexStop2, indexStart, indexDNA, iGene: integer
Gene: array [1:*, 1:2] of integer
indexDNA ← iPhase
iGene ← 1
repeat
  indexStop1 ← NextStopCodon (indexDNA)
  if indexStop1 > 0 then
    indexStop2 ← NextStopCodon (indexStop1+3)
    Length = IndexStop2 - IndexStop1 + 1
    if Length ≥ LengthMin then
          IndexStart <- NextStartCodon (indexStop1+3)
          if indexStart > O and indexStart < IndexStop2 then
               Gene [iGene, 1] ← indexStart
               Gene [iGene, 2] ← indexStop2
               iGene ← iGene +1
          endif
    endif
    endif
    indexDNA ← max (indexStop2, indexStop1, indexDNA+3)
  until (indexStop1 = 0) or (indexDNA > LengthSequence)
```

Search the genes on 6 sequences

- Apply the algorithm on the 3 phases of each strand

```
for iPhase from 1 to 3 do
    Genes ← GenePredictor (DNASequence, IPhase)
    GenesReverse ← GenePredictor
                    (ComputeReverseCompSequence
                            (DNASequence, Length),
                    iPhase)
endfor
```

- In bacterial genomes, coding sequences do not overlap on phases and reverse strand

Compute the reverse complementary sequence

```
ACCGTCTGGTACCCGTATTGCTTATGGCATATA
```

```
T
```

Compute the reverse complementary sequence

ACCGTCTGGTACCCGTATTGCTTATGGCATATA

TA

Compute the reverse complementary sequence

ACCGTCTGGTACCCGTATTGCTTATGGCATATA

TAT

Compute the reverse complementary sequence

ACCGTCTGGTACCCGTATTGCTTATGGCATATA

TATATGCCATAAGCAATACGGGTACCAGACGGT

Compute the reverse complementary sequence

```
function ComputeReverseCompSequence (DNAsequence: character string [1:*],
length: integer) return character string [1:*]
    ReverseCompSequence: character string [1:*]
    i, j: integer
    j ← 1
    for i from length downto 1
        case DNAsequence[i] of
            "A": ReverseCompSequence [j] ← "T"
            "C": ReverseCompSequence [j] ← "G"
            "G": ReverseCompSequence [j] ← "C"
            "T": ReverseCompSequence [j] ← "A"
        j ← j+1
    endfor
    return ReverseCompSequence
end ComputeReverseCompSequence
```