3. Prédiction des gènes

- Tous les gènes se terminent sur un codon stop
- Un algorithme simple de prédiction de gènes
- À la recherche des codons start et stop
- **Prédiction de tous les gènes d'une séquence**
- Comment améliorer la qualité des prédictions ?
- L'algorithme de Boyer-Moore
- Index et arbre des suffixes
- Des méthodes probabilistes à la rescousse
- Comment évaluer la qualité de prédiction des méthodes ?
- La prédiction de gènes dans les génomes eucaryotes

```
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)
```

Rechercher les gènes dans les 6 séquences

- Appliquer l'algorithme dans les 3 phases des 2 brins

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

Rechercher les gènes dans les 6 séquences

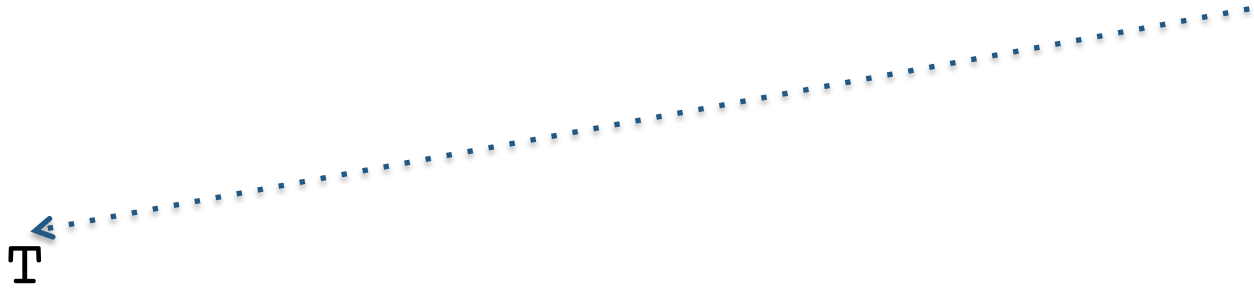
- Appliquer l'algorithme dans les 3 phases des 2 brins

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

- Dans les génomes bactériens, les régions codantes ne se recouvrent pas entre elles, et ne se retrouvent pas en face à face sur les brins

Calculer la séquence complémentaire inversée

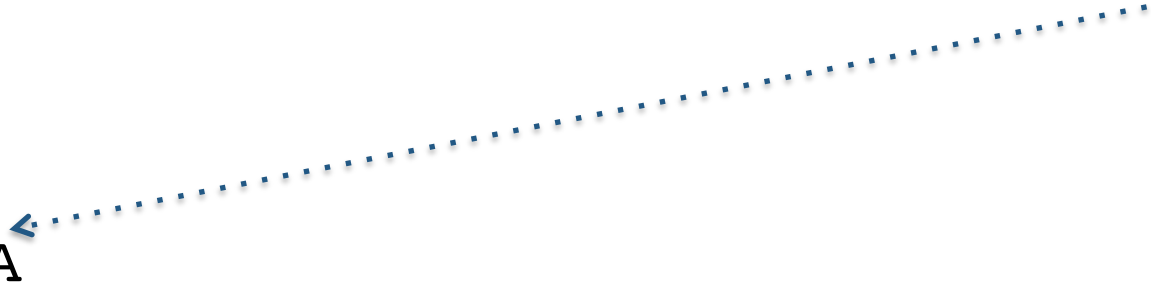
ACCGTCTGGTACCCGTATTGCTTATGGCATATA



T

Calculer la séquence complémentaire inversée

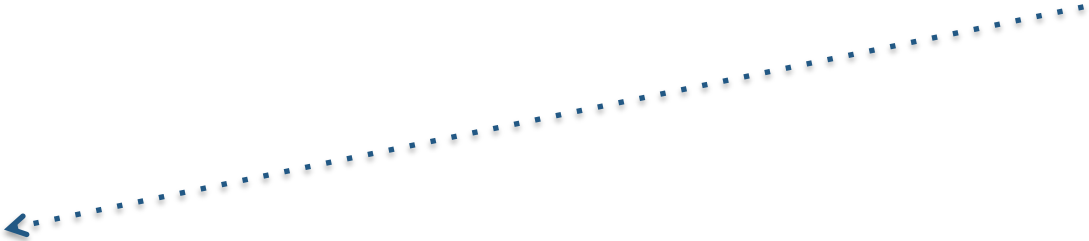
ACCGTCTGGTACCCGTATTGCTTATGGCATATA



TA

Calculer la séquence complémentaire inversée

ACCGTCTGGTACCCGTATTGCTTATGGCATATA



TAT

Calculer la séquence complémentaire inversée

ACCGTCTGGTACCCGTATTGCTTATGGCATATA

TATATGCCATAAGCAATACGGGTACCAGACGGT

Calculer la séquence complémentaire inversée

```
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
```