

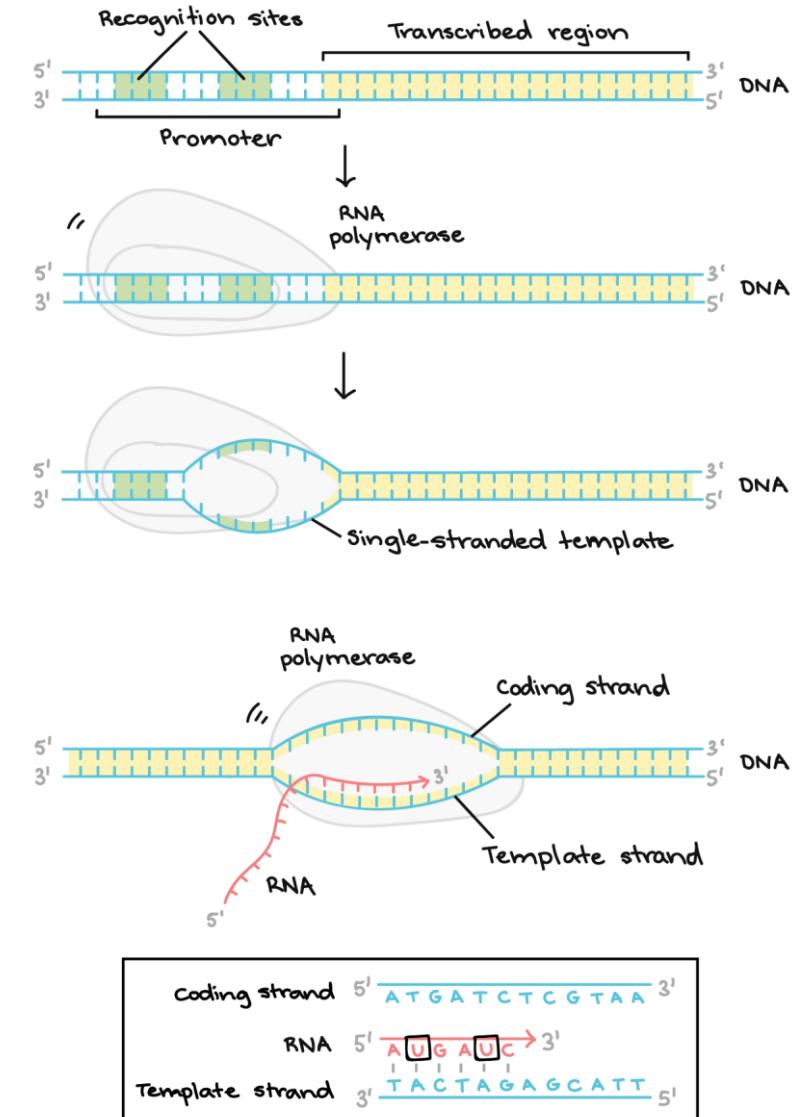
Motif Search

MPA-PRG: Programming in Bioinformatics

Exercise 7

Introduction

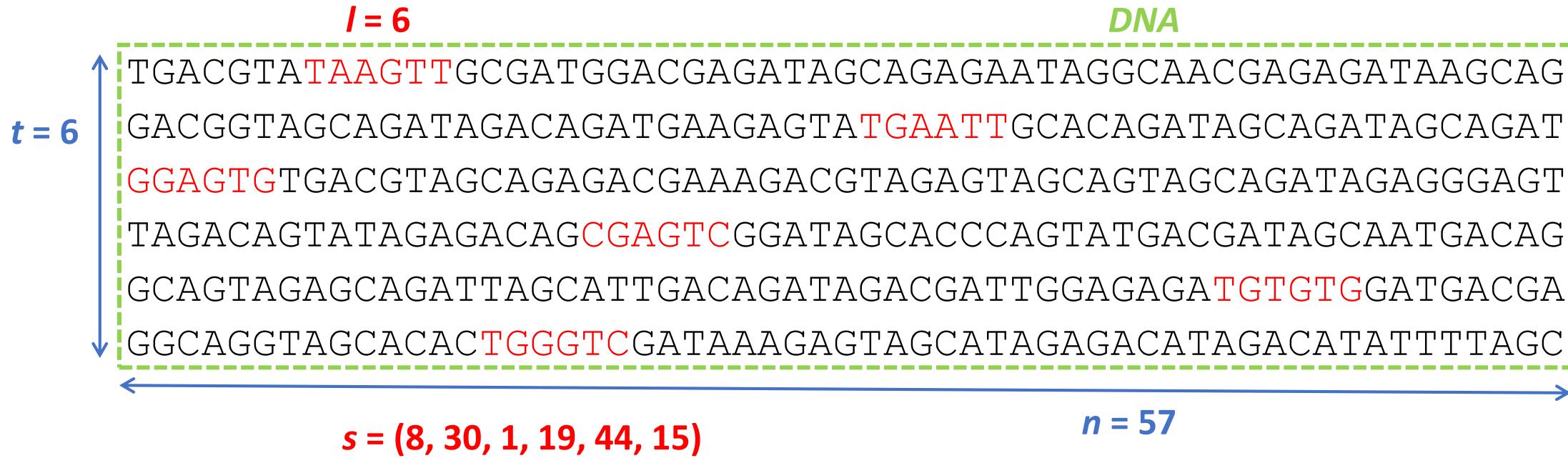
- transcription
 - process of copying DNA to RNA
- transcription factor
 - specific protein
 - binds to a transcription motif
 - transcription initiation and regulation
- transcription factor binding site
 - short segment of DNA (5 – 20 bp)
 - can be found on both strands
 - frequent repetition within the genome



Word-based Methods

- often use exhaustive search
- give globally optimal result
- suitable for short motifs
- can be very fast when implemented with suffix trees
- the motif has to be strongly conserved
- problem with large set of candidates

The Motif Finding Problem



t – the number of DNA sequences

n – the length of each DNA sequence

DNA – a set of sequences, a matrix $t \times n$

l – the length of the motif (l -mer)

s_i – the starting index of the motif in sequence i

$s = (s_1, s_2, \dots, s_t)$ – a vector of starting indexes for t sequences

The Brute-Force Motif Search

- generate all combinations of the vector s
- for each vector s , calculate a frequency profile and a score
- select vector s with the maximum score
- the vector s with the maximum score determines the sequences whose consensus is a candidate motif

The Brute-Force Motif Search

- **function** Score ()
- **function** NextLeaf ()
- **function** BFMotifSearch ()

Task 1

- in R, create a function `Score()`, that calculates the score for a consensus string
- input:
 - an array of starting indexes
 - `DNAStringSet` object of sequences (for example file `seq_score.fasta`)
 - motif length
- output:
 - the score for the consensus string

Alignment matrix

A	T	C	C	G	T	A
G	T	G	C	A	T	A
A	A	G	C	G	T	A
A	T	G	C	G	T	G

Frequency profile

A	3	1	0	0	1	0	3
C	0	0	1	4	0	0	0
G	1	0	3	0	3	0	1
T	0	3	0	0	0	4	0

Consensus string

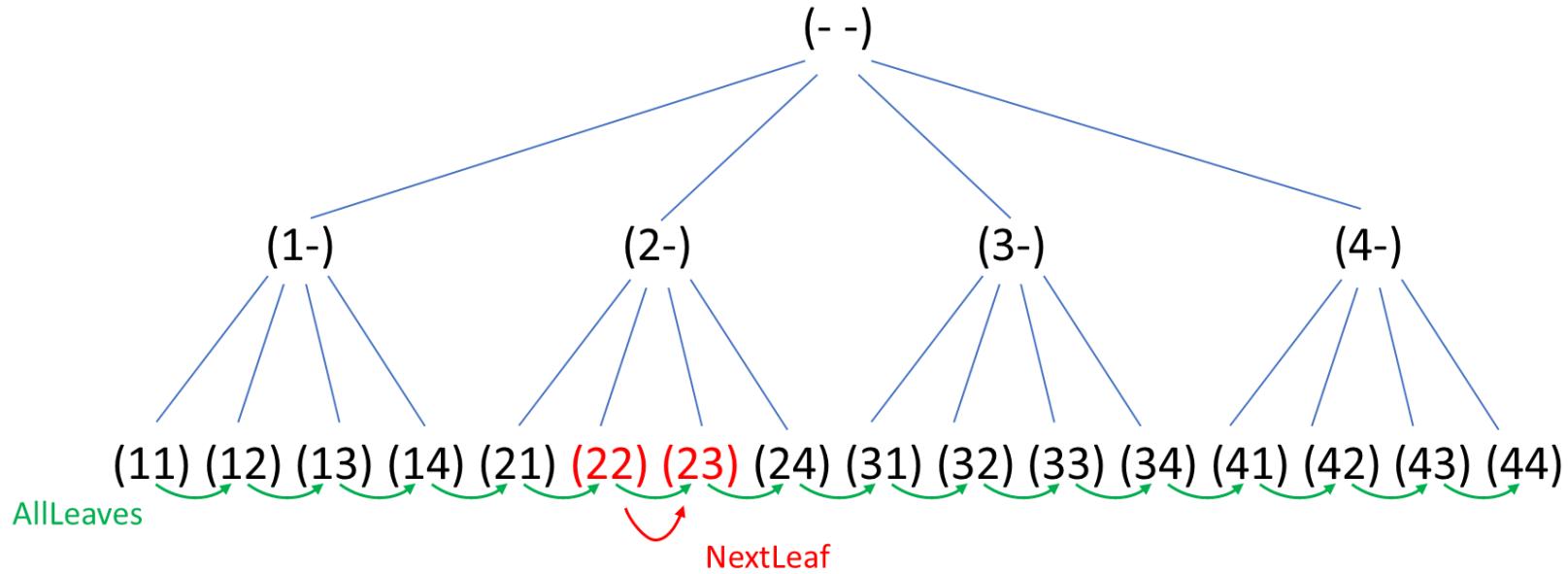
A T G C G T A

Score

$3+3+3+4+3+4+3 = \textcolor{red}{23}$

Task 2

```
NextLeaf(s, t, k)
1   for i ← t to 1
2     if si < k
3       si ← si + 1
4   return s
5   si ← 1
6   return s
```



- input:
 - s an array of starting indexes $s = (s_1 s_2 \dots s_t)$, where t is the number of sequences
 - t number of sequences
 - k $k = n - l + 1$, where n is length of sequences and l is motif length
- output:
 - s an array of starting indexes that corresponds to the next leaf in the tree

Task 3

```
BFMotifSearch(DNA, t, n, l)
1   s ← (1, 1, . . . , 1)
2   bestScore ← Score(s, DNA, 1)
3   while forever
4       s ← NextLeaf(s, t, n - l + 1)
5       if Score(s, DNA, 1) > bestScore
6           bestScore ← Score(s, DNA, 1)
7           bestMotif ← (s1, s2, . . . , st)
8       if s = (1, 1, . . . , 1)
9           return bestMotif
```

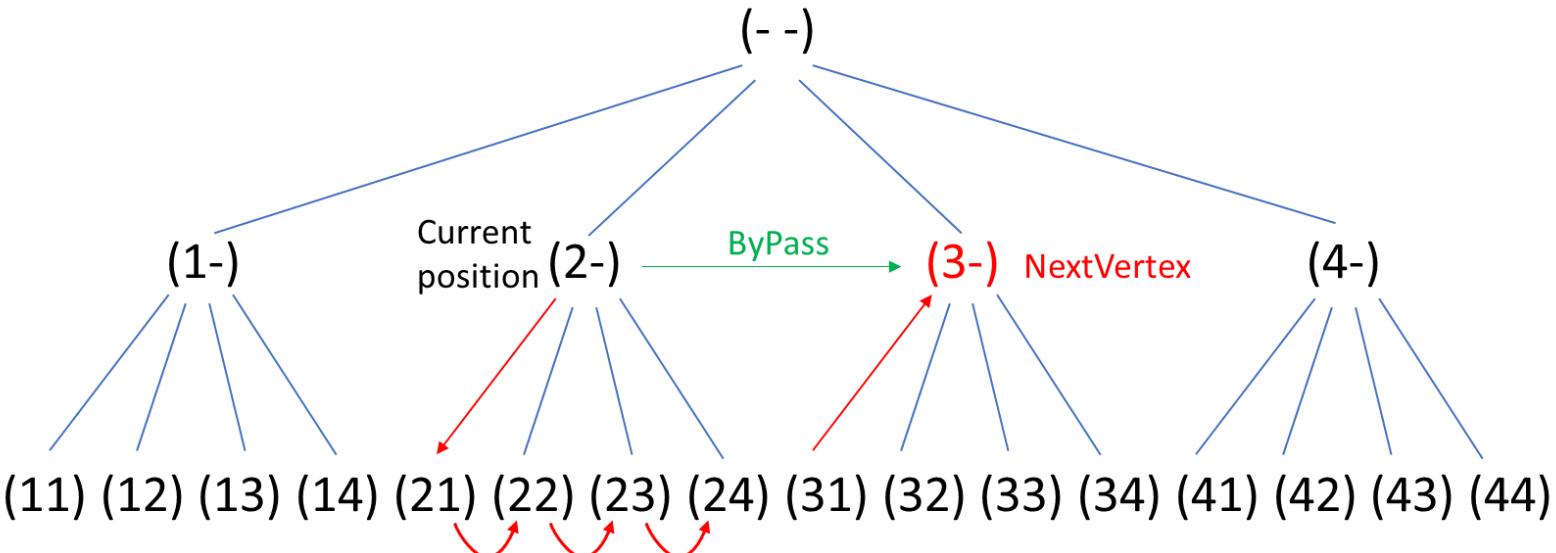
- **input:**
 - DNA DNAStringSet object of sequences (for example file seq_motif.fasta)
 - t number of sequences
 - n length of each sequence
 - l motif length
- **output:**
 - bestMotif an array of starting indexes for each sequence with the best score for the consensus string

The Branch-and-Bound Motif Search

- **function** NextVertex ()
- **function** ByPass ()
- **function** BBMotifSearch ()

Task 4

```
NextVertex(s, i, t, k)
1  if i < t
2    si+1 ← 1
3    return (s, i + 1)
4  else
5    for j ← t to 1
6      if sj < k
7        sj ← sj + 1
8        return (s, j)
9  return (s, 0)
```



- input:

- s an array of starting indexes $s = (s_1 s_2 \dots s_t)$, where t is the number of sequences
- i level of vertex
- t number of sequences
- k $k = n - l + 1$, where n is length of DNA sequences and l is motif length

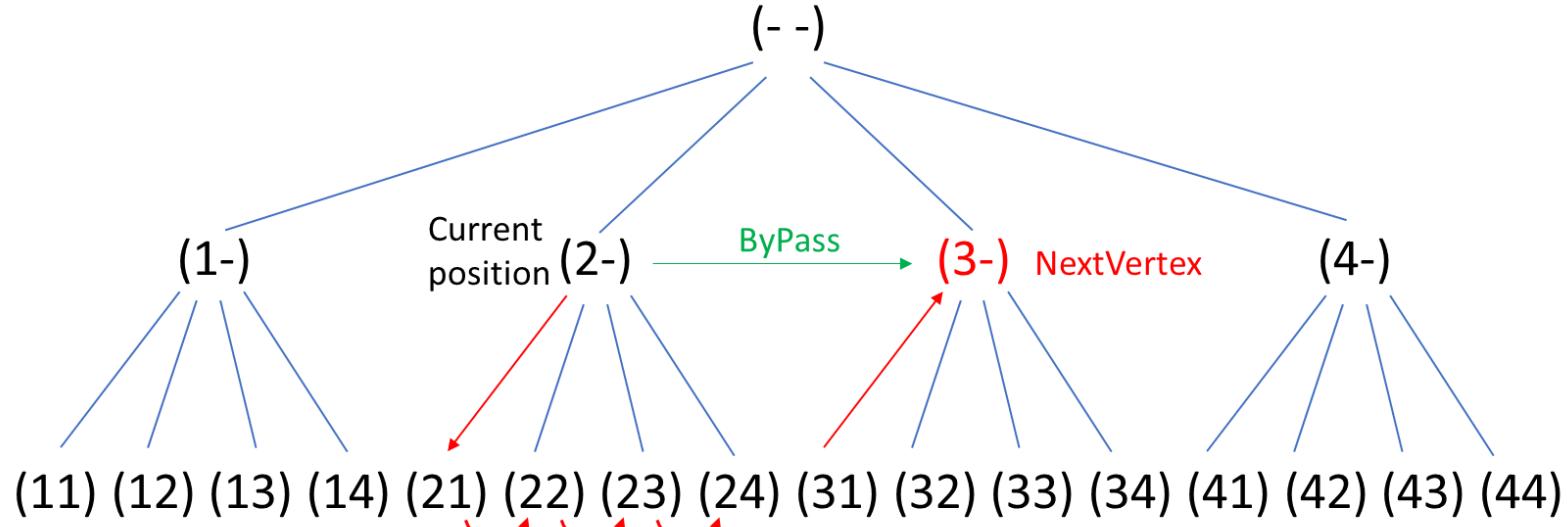
- output:

- s the next vertex in the tree
- current level of vertex

Task 5

```
ByPass(s, i, t, k)
1   for j ← i to 1
2     if sj < k
3       sj ← sj + 1
4     return (s, j)
5   return (s, 0)
```

- input:
 - s an array of starting indexes $s = (s_1 s_2 \dots s_t)$, where t is the number of sequences
 - i level of vertex
 - t number of sequences
 - k $k = n - l + 1$, where n is length of DNA sequences and l is motif length
- output:
 - l -mer of the next leaf after a skip of a subtree
 - current level of vertex



Task 6

- in R, create a function `BBMotifSearch()` according to the following pseudocode
- **input:**
 - DNA `DNAStringSet` object of sequences (for example file `seq_motif.fasta`)
 - t number of sequences
 - n length of each sequence
 - l motif length
- **output:**
 - `bestMotif` an array of starting indexes for each sequence with the best score for the consensus string
- modify function `Score()` to calculate score for the consensus string of the first i sequences of DNA

Task 6

```
BBMotifSearch(DNA, t, n, l)
1  s ← (1, . . . , 1)
2  bestScore ← 0
3  i ← 1
4  while i > 0
5    if i < t
6      optimisticScore ← Score(s, i, DNA, 1) + (t - i) * 1
7      if optimisticScore < bestScore
8        (s, i) ← ByPass(s, i, t, n - 1 + 1)
9      else
10        (s, i) ← NextVertex(s, i, t, n - 1 + 1)
11    else
12      if Score(s, t, DNA, 1) > bestScore
13        bestScore ← Score(s, t, DNA, 1)
14        bestMotif ← (s1, s2, . . . , st)
15        (s, i) ← NextVertex(s, i, t, n - 1 + 1)
16 return bestMotif
```