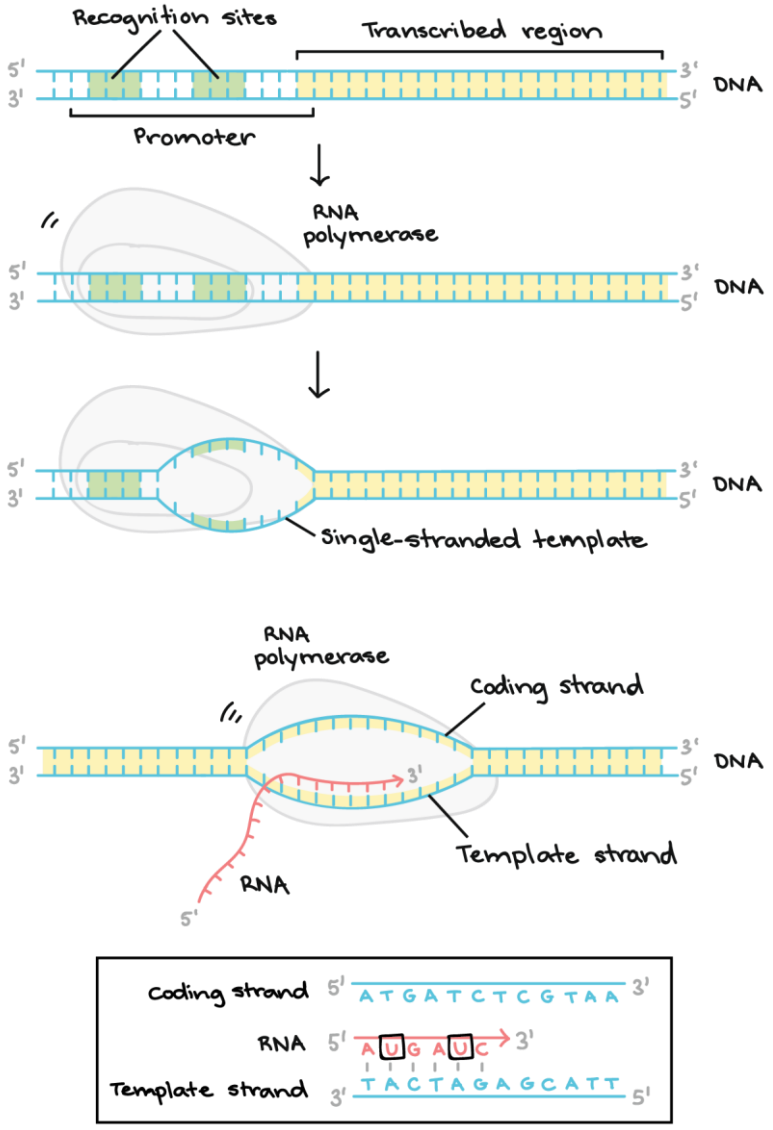
Motif Search

MPA-PRG: Programming in Bioinformatics

Exercise 7

Introduction

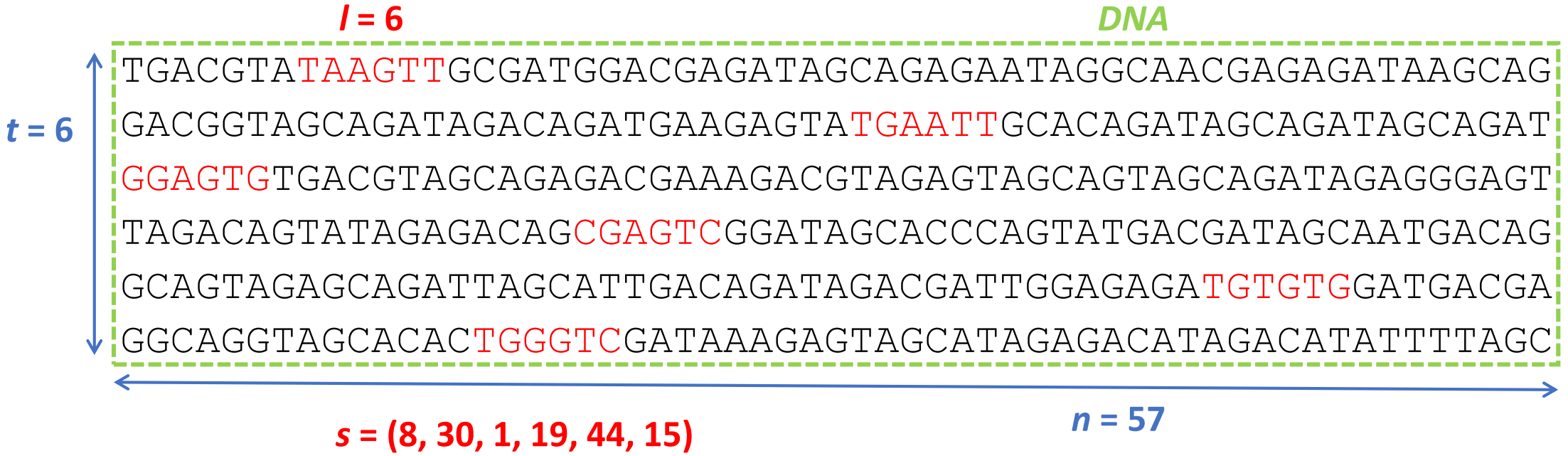
- Transcription
  - process of copying DNA to RNA
- Transcription factor
  - specific protein
  - binds to a transcription motif
  - transcription initiation and regulation
- Transcription factor binding site
  - short segment of DNA (5 – 20 bp)
  - can be found on both strands
  - frequent repetition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



Word-based Methods

- often use exhaustive search
- give globally optimal result
- suitable for short motifs (in eukaryotes; prokaryotes have longer 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
$s = (s_1, s_2, \ldots, s_t)$ – a vector of starting indexes for t sequences

The Motif Finding Problem

For any vector *s* from *DNA*, calculate a frequency profile and the total score of the vector is the sum of maximum frequencies in the profile columns.

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 = 23
```

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 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

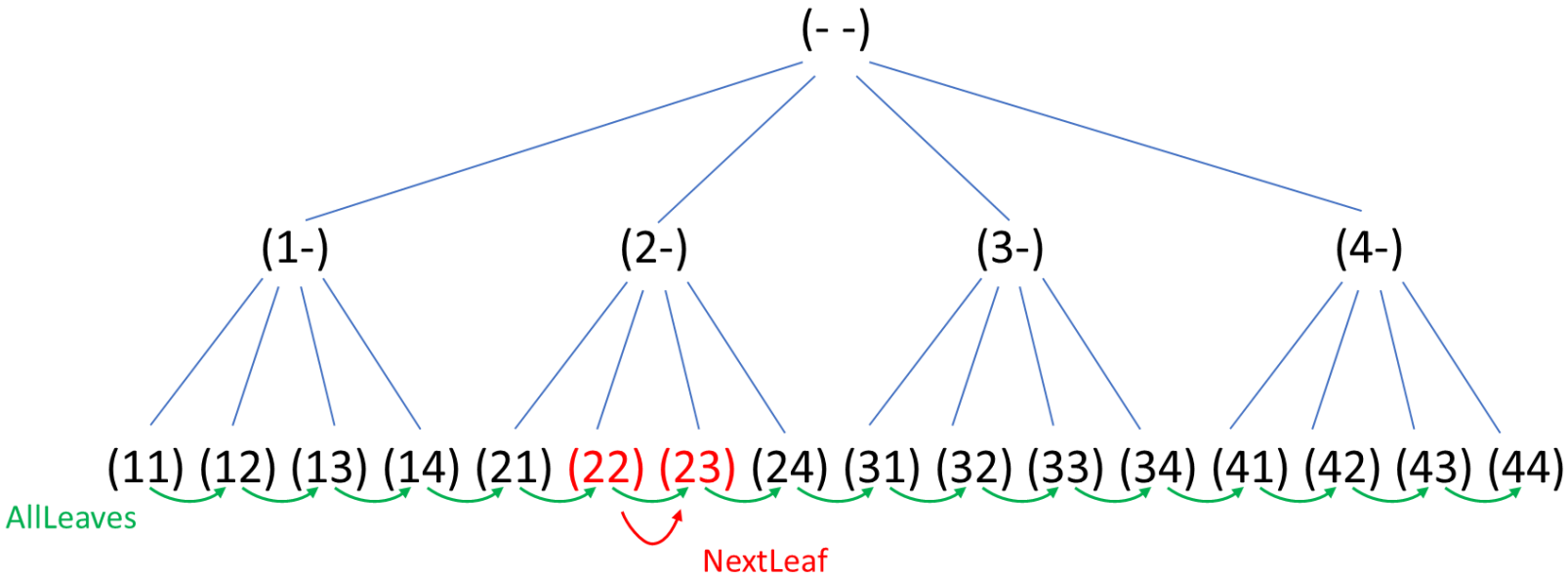
```
A T G C G T A
```

Score

```
3+3+3+4+3+4+3 = 23
```

Task 2

```
NextLeaf(a, L, k)
1    for i ← L to 1
2      if a_i < k
3        a_i ← a_i + 1
4        return a
5      a_i ← 1
6    return a
```



Input:

- *L*-mer $a = (a_1\ a_2\ \ldots\ a_L)$ of starting indexes
- number of DNA sequences
- $k = n - l + 1$, where *n* is length of DNA sequences and *l* is motif length

Output:

- *L*-mer of the next leaf in the tree

Task 3

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

Input:

- `DNAStringSet` of DNA sequences (for example file `seq_motif.fasta`)
- number of DNA sequences
- length of each DNA sequence
- motif length

Output:

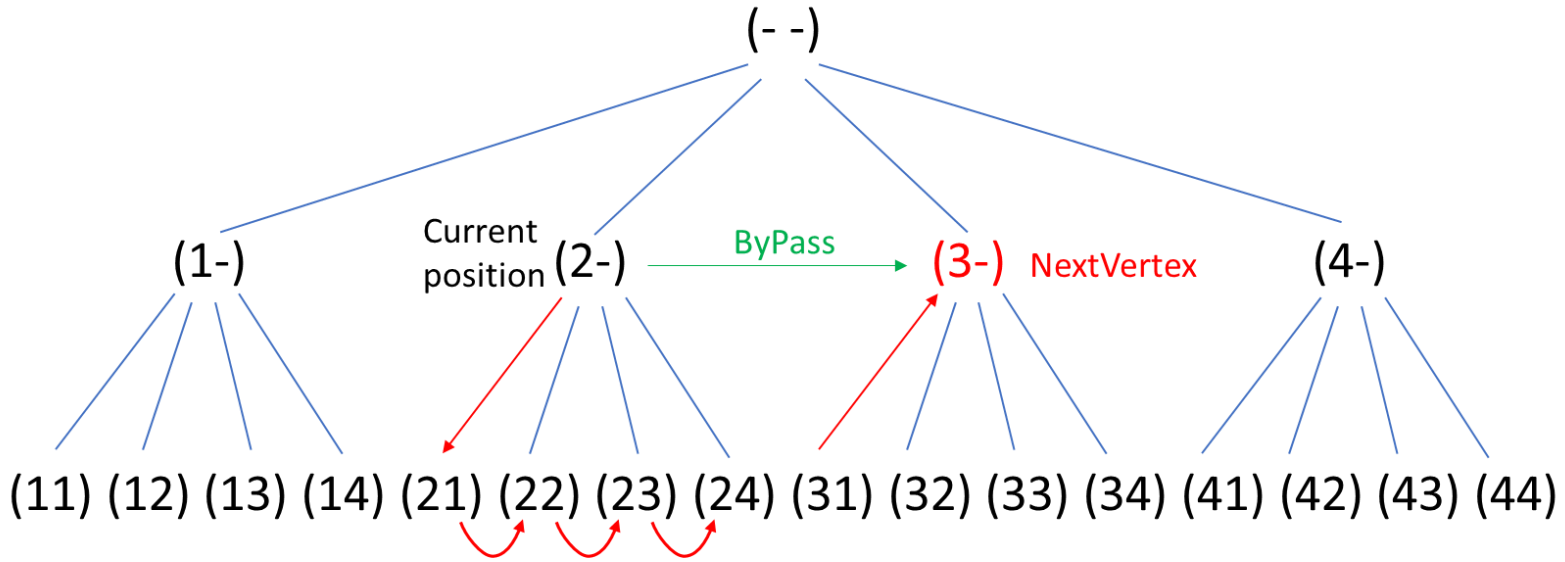
- an array of starting positions for each DNA sequence with the best score for the consensus string

The Branch-and-Bound Motif Search

- Function `NextVertex()`
- Function `ByPass()`
- Function `BBMotifSearch()`

Task 4

```
NextVertex(a, i, L, k)
1    if i < L
2      a_{i+1} ← 1
3      return (a, i + 1)
4    else
5      for j ← L to 1
6        if a_j < k
7          a_j ← a_j + 1
8          return (a, j)
9    return (a, 0)
```



Input:

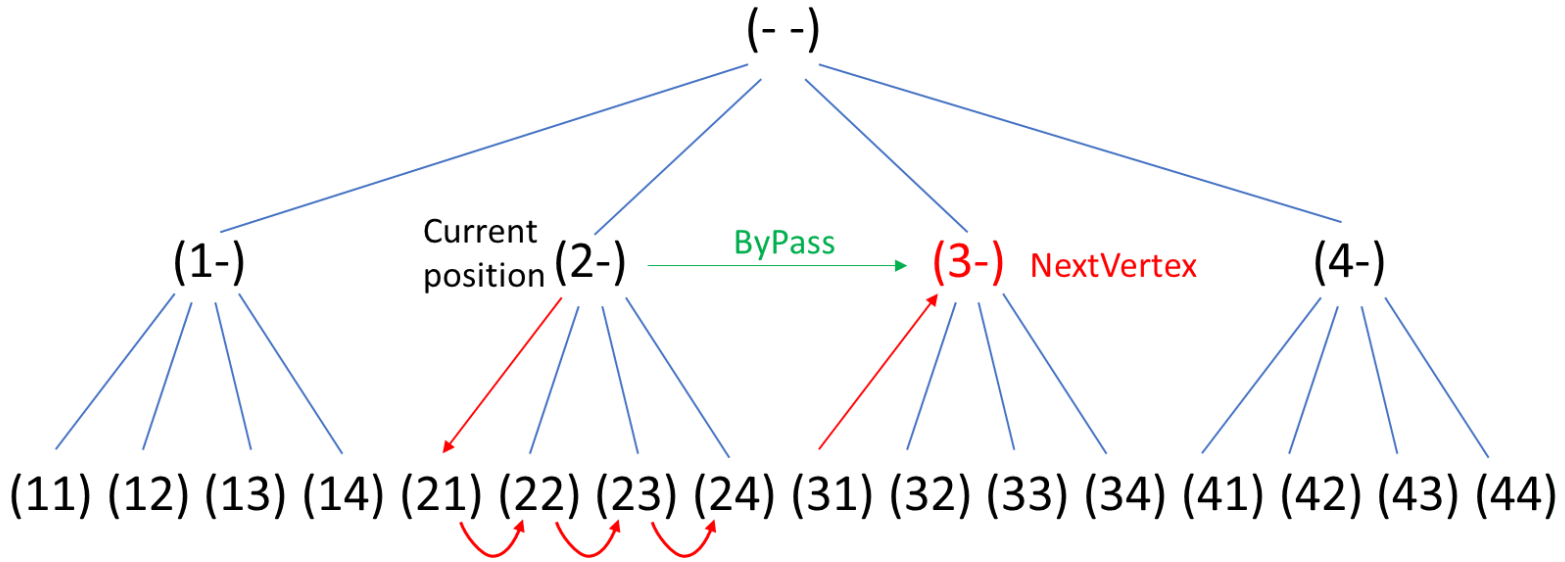
- *L*-mer $a = (a_1\ a_2\ \ldots\ a_L)$ of starting indexes
- level of vertex
- number of DNA sequences
- $k = n - l + 1$, where *n* is length of DNA sequences and *l* is motif length

Output:

- *L*-mer of the next vertex in the tree
- current level of vertex

Task 5

```
ByPass(a, i, L, k)
1    for j ← i to 1
2      if aj < k
3        aj ← aj + 1
4        return (a, j)
5    return (a, 0)
```



Input:

- *L*-mer $a = (a_1\ a_2\ \ldots\ a_L)$ of starting indexes
- level of vertex
- number of DNA sequences
- $k = n - l + 1$, where *n* is length of DNA sequences and *l* is motif length

Output:

- *L*-mer of the next leaf after a skip of the subtree
- current level of vertex

Task 6

- In R, create a function `BBMotifSearch()` according to the following pseudocode.
- Input:
  - `DNAStringSet` of DNA sequences (for example file `seq_motif.fasta`)
  - number of DNA sequences
  - length of each DNA sequence
  - motif length
- Output:
  - an array of starting positions for each DNA sequence with the best score for the consensus string
- Modify function `Score()` to calculate a partial consensus score for first *i* rows 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, l) + (t - i) * l
7        if optimisticScore < bestScore
8          (s, i) ← ByPass(s, i, t, n - l + 1)
9        else
10         (s, i) ← NextVertex(s, i, t, n - l + 1)
11     else
12       if Score(s, t, DNA, l) > bestScore
13         bestScore ← Score(s, t, DNA, l)
14         bestMotif ← (s_1, s_2, ... , s_t)
15       (s, i) ← NextVertex(s, i, t, n - l + 1)
16   return bestMotif
```