Finding a pattern in a text
The idea is similar to Z-Algorithm
Re-use "information" from previous windows.

Look for pat in text[0...]

pat: a a a a

pat found in position 0 of text!

Now, we start to look for pat in text[1...]

Finding a pattern in a text
The idea is similar to Z-Algorithm
Re-use "information" from previous windows.

Look for pat in text[0...]

pat: a a a a

pat found in position 0 of text!

Now, we start to look for pat in text[1...]

If we save "information" from previous iteration, we then already know that text[1...3] = pat[0...2]

So we only need to compare text[4] = part[3]

Finding a pattern in a text
The idea is similar to Z-Algorithm
Re-use "information" from previous windows.

Look for pat in text[0...]

pat: a a a a

pat found in position 0 of text!

Now, we start to look for pat in text[1...]

If we save "information" from previous iteration, we then already know that text[1...3] = pat[0...2]

So we only need to compare text[4] = part[3]

How many characters to skip?

To know this, we construct a vector lps of pat.size()

lps[i] = size of longest prefix of pat[0...i] which is also a suffix of pat[0...i]

pat: a a a a

lps[3] = longest prefix of <math>lps[0...3] which is also a suffix of pat[0...3]



lps[3] = 3

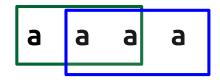
How many characters to skip?

To know this, we construct a vector lps of pat.size()

lps[i] = size of longest prefix of pat[0...i] which is also
a suffix of pat[0...i]

pat: a a a a

lps[3] = longest prefix of <math>lps[0...3] which is also a suffix of pat[0...3]



lps[3] = 3

i	longest prefix/suffix in pat[0i]	size	
0		lps[0] =	
1		lps[1] =	
2		lps[2] =	
3		lps[3] =	6

How many characters to skip?

To know this, we construct a vector lps of pat.size()

lps[i] = size of longest prefix of pat[0...i] which is also a suffix of pat[0...i]

i	<pre>longest prefix/suffix in pat[0i]</pre>	size
0		lps[0] =
1		lps[1] =
2		lps[2] =
3		lps[3] =
•••		•••
8		lps[8] =
9		lps[9] =
10		lps[10] =

7

```
pat: a a a c a a a a 0 1 2 3 4 5 6 7 lps:
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
        for(int i = 0; i < pat.size(); i++)</pre>
 4
 5
            lps.push back(0);
 6
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
10
        lps[0] = 0;
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
19
                 lps[i] = len;
20
                 i++;
21
22
            else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
30
                     lps[i] = 0;
31
                     i++;
32
33
34
35
        return lps;
36
                                                   8
37 }
```

```
pat:
            a
                    C
                а
                       a
                                  а
                               6
                   3
                       4
        0
lps:
        0
i=1:
pat:
        a
            a
      len i
pat[i] = pat[len]
len = 1
lps[i] = 1
i++;
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
        for(int i = 0; i < pat.size(); i++)</pre>
 4
 5
6
            lps.push back(0);
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
10
        lps[0] = 0;
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
19
                 lps[i] = len;
20
                 i++;
21
22
            else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
30
                     lps[i] = 0;
31
                     i++;
32
33
34
35
36
        return lps;
                                                    9
37 }
```

```
pat:
                a 2
                    C
            a
                       a
                                  а
                    3
                               6
            1
                       4
lps:
        0
i=2:
pat:
        a
            a
                a ..
        0
           len i
pat[i] = pat[len]
len = 2
lps[i] = 2
i++;
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
        for(int i = 0; i < pat.size(); i++)</pre>
 4
 5
6
            lps.push back(0);
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
10
        lps[0] = 0;
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
19
                 lps[i] = len;
20
                 i++;
21
22
            else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
30
                     lps[i] = 0;
31
                     i++;
32
33
34
35
36
        return lps;
                                                    10
37 }
```

```
pat:
            a
                   C
                       a
                a
                           a
                                  а
                               6
            1
                       4
                           5
lps:
        0
i=3:
pat:
        a
            a
                   C
            1
        0
               len i
pat[i] != pat[len]
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
        for(int i = 0; i < pat.size(); i++)</pre>
 4
 5
6
             lps.push back(0);
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
10
        lps[0] = 0;
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
19
                 lps[i] = len;
20
                 i++;
21
22
             else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
30
                     lps[i] = 0;
31
                     i++;
32
33
34
35
        return lps;
36
                                                    11
37 }
```

```
pat:
                         C
                    a
                             a
                             4
lps:
           0
i=3:
pat:
           a
               a
                  len i
 pat[i] != pat[len]
 len = lps[len - 1] = 1
 Move back pointer "len" to the
 last character of previous
 longest/prefix suffix
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
        for(int i = 0; i < pat.size(); i++)</pre>
 4
 5
             lps.push back(0);
 6
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
10
        lps[0] = 0;
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
                 lps[i] = len;
19
20
                 i++;
21
22
             else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
                     lps[i] = 0;
30
31
                     i++;
32
33
34
35
36
        return lps;
                                                    12
37
```

```
pat:
                   C
                a
                       a
                               6
                       4
                           5
lps:
        0
i=3:
pat:
        a
                   3
           len
                    i
pat[i] != pat[len]
len = lps[len - 1] = 0
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
        for(int i = 0; i < pat.size(); i++)</pre>
 4
 5
            lps.push back(0);
 6
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
10
        lps[0] = 0;
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
19
                 lps[i] = len;
20
                 i++;
21
22
            else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
30
                     lps[i] = 0;
31
                     i++;
32
33
34
35
36
        return lps;
                                                    13
37 }
```

```
pat:
                   C
                       a
               a
                          a
                              a
                                 a
               2
                              6
                   3
                       4
                          5
lps:
        0
                   0
i=3:
pat:
        a
            a
               a
                   C
            1
                   3
       len
                    i
pat[i] != pat[len] and len == 0
lps[3] = 0
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
        for(int i = 0; i < pat.size(); i++)</pre>
 4
 5
            lps.push back(0);
 6
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
10
        lps[0] = 0;
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
                 lps[i] = len;
19
20
                 i++;
21
22
            else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
30
                     lps[i] = 0;
31
                     i++;
32
33
34
35
36
        return lps;
                                                    14
37 }
```

```
pat:
                   C
                       a
            a
                a
                           a
                              a
                                 а
                              6
                   3
                       4
                          5
lps:
                   0
        0
i=4:
pat:
        a
            а
                a
                       a
                   C
            1
                   3
                       4
       len
 pat[i] = pat[len]
 len = 1
 lps[i] = 1
 i++;
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
        for(int i = 0; i < pat.size(); i++)</pre>
 4
 5
6
            lps.push back(0);
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
10
        lps[0] = 0;
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
                 lps[i] = len;
19
20
                 i++;
21
22
            else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
30
                     lps[i] = 0;
31
                     i++;
32
33
34
35
        return lps;
36
                                                    15
37 }
```

```
pat:
               a
                   C
           a
                      а
                          a
                             a a
                      4
               2
                  3
                         5
        0
               2
                      1
        0
Lps:
i=5:
pat:
        a
            a
                      a
                          a ...
                   C
                   3
                      4
          len
pat[i] = pat[len]
len = 2
lps[i] = 2
i++;
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
        for(int i = 0; i < pat.size(); i++)</pre>
 4
 5
6
             lps.push back(0);
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
10
        lps[0] = 0;
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
19
                 lps[i] = len;
20
                 i++;
21
22
             else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
30
                     lps[i] = 0;
31
                     i++;
32
33
34
35
36
        return lps;
                                                    16
37 }
```

```
pat:
                    a
                        C
               a
                             а
                                  a
                                          а
                                                           std::vector<int> computeLPS(const std::string& pat)
                             4
                    2
                        3
                                 5 6
          0
                                                       2
                                                       3
                                                              std::vector<int> lps;
                    2
                                      3
lps:
          0
                             1
                                                              for(int i = 0; i < pat.size(); i++)</pre>
                                                       4
                                                       5
6
                                                                  lps.push back(0);
                                                       7
                                                              // length of longest prefix/suffix in pat
                                                              int len = 0;
                                                       8
                                                       9
                                                      10
                                                              lps[0] = 0;
                                                      11
i=6:
                                                      12
                                                              int i = 1;
                                                      13
                                                      14
                                                              while( i < pat.size() )</pre>
pat:
           a
               a
                             a
                                  a
                                       а
                         C
                                                      15
                             4
               1
                        3
                                  5
                                       6
                                                      16
                                                                   if( pat[i] == pat[len] )
                                                      17
                                                      18
                                                                      len++;
                                                      19
                                                                      lps[i] = len;
                                                      20
                                                                       i++;
                                                      21
                                                      22
                                                                  else // pat[i] != pat[len]
                  len
                                                      23
                                                      24
                                                                       if(len != 0)
                                                      25
                                                      26
                                                                          len = lps[len - 1];
 pat[i] = pat[len]
                                                      27
 len = 3
                                                      28
                                                                      else // len == 0
 lps[i] = 3
                                                      29
 i++;
                                                      30
                                                                          lps[i] = 0;
                                                      31
                                                                          i++;
                                                      32
                                                      33
                                                      34
                                                      35
                                                      36
                                                               return lps;
                                                      37 }
                                                                                                       17
```

```
pat:
                    a
                         C
                             a
                                           a
                                                           std::vector<int> computeLPS(const std::string& pat)
                             4
                    2
                        3
                                 5
                                       6
          0
                                                        2
                                                        3
                                                               std::vector<int> lps;
                    2
lps:
                             1
           0
                                                               for(int i = 0; i < pat.size(); i++)</pre>
                                                        4
                                                        5
6
                                                                   lps.push back(0);
                                                        7
                                                               // length of longest prefix/suffix in pat
                                                               int len = 0;
                                                        8
                                                        9
                                                       10
                                                               lps[0] = 0;
                                                       11
i=7:
                                                       12
                                                               int i = 1;
                                                       13
                                                       14
                                                               while( i < pat.size() )</pre>
pat:
           a
               a
                    a
                              a
                                  a
                                       a
                                            a
                                                       15
                              4
                                       6
                1
                         3
                                  5
                                                       16
                                                                   if( pat[i] == pat[len] )
                                                       17
                                                       18
                                                                       len++;
                                                       19
                                                                       lps[i] = len;
                                                       20
                                                                       i++;
                                                       21
                                                       22
                                                                   else // pat[i] != pat[len]
                       len
                                                       23
                                                       24
                                                                       if(len != 0)
                                                       25
 pat[i] != pat[len]
                                                       26
                                                                           len = lps[len - 1];
                                                       27
                                                       28
                                                                       else // len == 0
                                                       29
                                                       30
                                                                           lps[i] = 0;
                                                       31
                                                                           i++;
                                                       32
                                                       33
                                                       34
                                                       35
                                                       36
                                                               return lps;
                                                                                                        18
                                                       37 }
```

```
pat:
                                   a a
                          4
                  2
                     3
                             5 6
         0
lps:
                          1
i=7:
pat:
                          a
                                       а
                          4
                      3
                                   6
                               5
                 len
 pat[i] != pat[len]
 len = lps[len - 1] = 1
     = lps[3 - 1] = 2
 Move back pointer "len" to the
 last character of previous
 longest/prefix suffix
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
        for(int i = 0; i < pat.size(); i++)</pre>
 4
 5
             lps.push back(0);
 6
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
        lps[0] = 0;
10
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
                 lps[i] = len;
19
20
                 i++;
21
22
             else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
30
                     lps[i] = 0;
31
                     i++;
32
33
34
35
36
        return lps;
37
                                                    19
```

```
pat:
               a
                   C
                       a
                                 a
                      4
               2
                   3
                         5 6
        0
                              3
                                 3
                       1
        0
Lps:
i=7:
pat:
        a
            a
                       a
                          а
                              a
                                  a
                   C
                       4
                              6
            1
                   3
                          5
              len
pat[i] = pat[len]
len = 3
lps[i] = 3
i++;
```

```
std::vector<int> computeLPS(const std::string& pat)
 2
 3
        std::vector<int> lps;
 4
        for(int i = 0; i < pat.size(); i++)</pre>
 5
6
             lps.push back(0);
 7
        // length of longest prefix/suffix in pat
        int len = 0;
 8
 9
10
        lps[0] = 0;
11
12
        int i = 1;
13
14
        while( i < pat.size() )</pre>
15
16
             if( pat[i] == pat[len] )
17
18
                 len++;
19
                 lps[i] = len;
20
                 i++;
21
22
             else // pat[i] != pat[len]
23
24
                 if(len != 0)
25
26
                     len = lps[len - 1];
27
28
                 else // len == 0
29
30
                     lps[i] = 0;
31
                     i++;
32
33
34
35
36
        return lps;
                                                    20
37 }
```