CSE 494/598: Algorithms in Computational Biology – Assignment
Exact Pattern Matching

Ajay Kannan - 1219387832

2. Biological sequences are often circular as in many bacterial genomes. Scientists simply cut the genome sequence at either an arbitrary point or at a origin of replication. Genome assemblers (a program that can stitch short sequences into a large genome sequence) also do this by breaking the circular sequence at a random location and report it as a linear sequence. If an identical circular sequence is broken at a different positions, they can often result in different linear sequences. Given two linear sequences X and Y, explain how we can use Z-algorithm to check if X and Y come from an identical circular sequence?

Ans) We can arrange the X and Y sequences, as described in the Z algorithm, but duplicating one of them more. For example,

X $ Y Y

More elaborately, let's take GAGGCATT as X and ATTGAGGC as Y. As mentioned above, we perform z algorithm in the the full string,

GAGGCATT $ ATTGAGGC ATTGAGGC

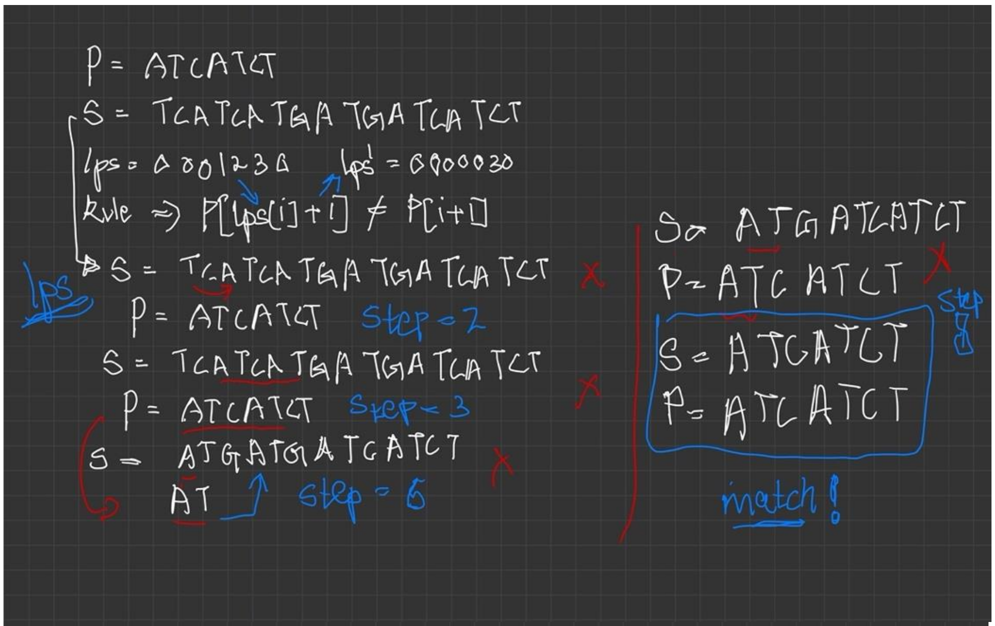
Calculating Z array - 00000000 0 00080110 00050110.

- ➔ len(max(z)) = 8 == len(P) = 8
- ➔ Matching circular pattern.

The reason why we are taking YY is if the pattern is the same as X, we don't know the when the sequence starts. But if the pattern starts, stops before ending the sequence. So, we take two Ys to match the circular pattern.

3. Is this a valid claim?

Ans)



P = ATCATCT

S = TCATCATGATGATCATCT

lps = 0001230 lps' = 0000030

Rule ⇒ P[lps[i]+1] ≠ P[i+1]

lps

S = TCATCATGATGATCATCT ✗

P = ATCATCT step = 2

S = TCATCATGATGATCATCT ✗

P = ATCATCT step = 3

S = ATGATGATCATCT ✗

AT step = 6

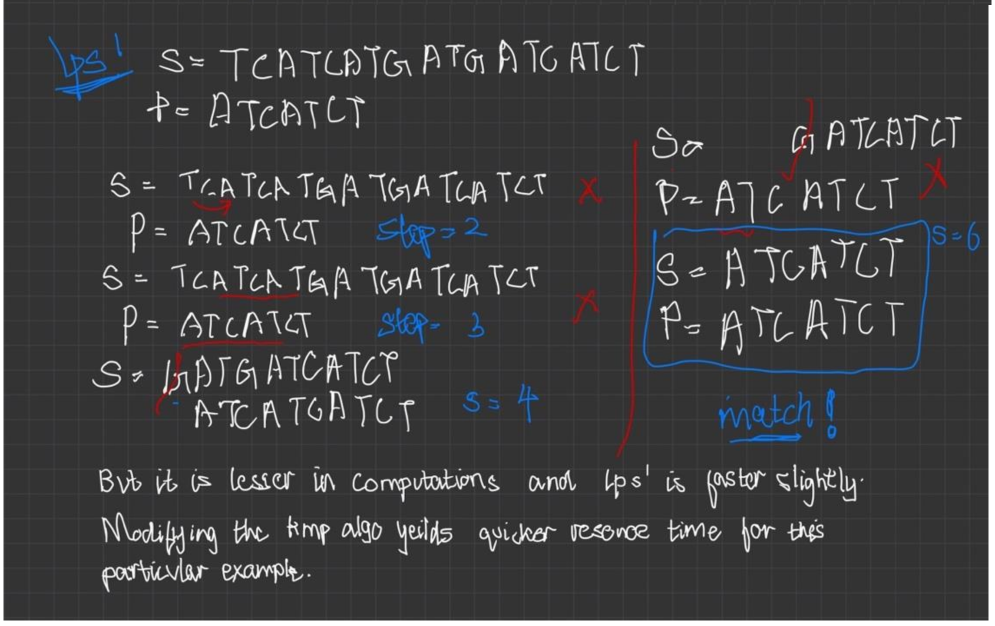
S = ATGATCATCT

P = ATCATCT ✗ step 8

S = ATCATCT

P = ATCATCT

match!



lps' S = TCATCATGATGATCATCT

P = ATCATCT

S = TCATCATGATGATCATCT ✗

P = ATCATCT step = 2

S = TCATCATGATGATCATCT ✗

P = ATCATCT step = 3

S = ATGATCATCT

ATCATGATCT S = 4

S = GATCATCT

P = ATCATCT ✗ S = 6

S = ATCATCT

P = ATCATCT

match!

But it is lesser in computations and lps' is faster slightly. Modifying the kmp algo yeilds quicker resonce time for this particular example.

4. And) KMP modified -

```
def computeLPSArray(pat, N, z):
    l, r, k = 0, 0, 0
    for i in range(1, n):
        if i > r:
            l, r = i, i
            while r < n and string[r - l] == string[r]:
                r += 1
            z[i] = r - l
            r -= 1
        else:
            k = i - l
            if z[k] < r - i + 1:
                z[i] = z[k]
            else:
                l = i
                while r < n and string[r - l] == string[r]:
                    r += 1
                z[i] = r - l
                r -= 1

      # Applying LPS' algo on Z values

      lps = [0] * len(z)
      for i in range(len(z)):
            if z[i] != 0:
                  for j in range(1,z[i]+1):
                        lps[i+j] = j
            elif z[len(z)-1] != 0:
                  lps[i] = z[i]

      z = lps
      z.reverse()
      for i in range(len(z)-1):
            if z[i] + 1 == z[i+1]:
                  z[i] = 0
      z.reverse()
      return z

pat = "ABABCABAB"
print(computeLPSArray(pat, len(pat), [0]*len(pat)))
```