Z Algorithm

This algorithm finds all occurrences of a pattern in a text in linear time. Let length of text be n and of pattern be m, then total time taken is O(m + n) with linear space complexity. Now we can see that both time and space complexity is same as KMP algorithm but this algorithm is Simpler to understand.

What is Z Array?

For a string str[0..n-1], Z array is of same length as string. An element Z[i] of Z array stores length of the longest substring starting from str[i] which is also a prefix of str[0..n-1]. The first entry of Z array is meaning less as complete string is always prefix of itself.

Example:

Index	0	1	2	3	4	5	6	7	8	9	10	11
Text	a	a	b	c	a	a	b	x	a	a	a	z
Z values	X	1	0	0	3	1	0	0	2	2	1	0

How is Z array helpful in Searching Pattern in Linear time?

The idea is to concatenate pattern and text, and create a string “P$T” where P is pattern, $ is a special character should not be present in pattern and text, and T is text. Build the Z array for concatenated string. In Z array, if Z value at any point is equal to pattern length, then pattern is present at that point.

Example:

Pattern P = "aab", Text T = "baabaa"

The concatenated string is = "aab$baabaa"

Z array for above concatenated string is {x, 1, 0, 0, 0,
3, 1, 0, 2, 1}.

Since length of pattern is 3, the value 3 in Z array indicates presence of pattern.

VIDEO REFERENCE

https://www.youtube.com/watch?v=CpZh4eF8QBw

CODE

```python
# Python3 program that implements Z algorithm
# for pattern searching

# Fills Z array for given string str[]
def getZarr(string, z):
    n = len(string)

    # [L,R] make a window which matches
    # with prefix of s
    l, r, k = 0, 0, 0
    for i in range(1, n):

        # if i>R nothing matches so we will calculate.
        # Z[i] using naive way.
        if i > r:
            l, r = i, i

            # R-L = 0 in starting, so it will start
            # checking from 0'th index. For example,
            # for "ababab" and i = 1, the value of R
            # remains 0 and Z[i] becomes 0. For string
            # "aaaaaa" and i = 1, Z[i] and R become 5
            while r < n and string[r - l] == string[r]:
                r += 1
            z[i] = r - l
            r -= 1
        else:

            # k = i-L so k corresponds to number which
            # matches in [L,R] interval.
            k = i - l
```

```
            # if Z[k] is less than remaining interval
            # then Z[i] will be equal to Z[k].
            # For example, str = "ababab", i = 3, R = 5
            # and L = 2
            if z[k] < r - i + 1:
                z[i] = z[k]

            # For example str = "aaaaaa" and i = 2,
            # R is 5, L is 0
            else:

                # else start from R and check manually
                l = i
                while r < n and string[r - l] == string[r]:
                    r += 1
                z[i] = r - l
                r -= 1

# prints all occurrences of pattern
# in text using Z algo
def search(text, pattern):

    # Create concatenated string "P$T"
    concat = pattern + "$" + text
    l = len(concat)

    # Construct Z array
    z = [0] * l
    getZarr(concat, z)

    # now looping through Z array for matching condition
    for i in range(l):

        # if Z[i] (matched region) is equal to pattern
        # length we got the pattern
        if z[i] == len(pattern):
            print("Pattern found at index",
                    i - len(pattern) - 1)
```

```python
# Driver Code
if __name__ == "__main__":
    text = "baabaa"
    pattern = "aab"
    search(text, pattern)

# geeks for geeks inda depkond edhu
```