Constructing KMP Failure func

→ bootstrapping process that compares the pattern to itself as in the KMP algo
→ each time we have 2 chars match, we set f(j) = k+1

```
def compute_kmp_fail(P):
    m=len(P)
    fail = [0] * m                  // presume overlap of 0 everywhere
    j=1, k=0
    while j < m:                    //compute f(j) during this pass
        if P[j] == P[k]:            // k+1 characters match thus far
            fail[j] = k+1
            j+=1
            k+=1
        elif k > 0:                 // k follows matching prefix
            k = fail[k-1]
    else
        j+=1                        // no match found starting at j
    return fail
```

Performance

→ excluding failure function, KMP is proportional in running time to the # of iterations of the while loop
→ one of the following cases occurs at each iteration of the loop:

1.) if T[j] = P[k], then j and k increase by 1; thus, s does not change
2.) if T[j] ≠ P[k] and k > 0, j does not change and s increases by at least 1, since s changes from j-k to j-f(k-1)
3.) if T[j] ≠ P[k] and k = 0, then j increases by 1 and s increases by 1, since k does not change