The University of Newcastle
School of Information and Physical Sciences

COMP2230/COMP6230 Algorithms

Tutorial Week 12

21 – 22 October 2021

1. The following is Rabin-Karp algorithm that searches for an occurrence of a pattern *p* in atext *t*. It returns the smallest index *i* such that *t*[*i*..*i*+*m*-1] = *p*, or -1 if no such index exists.

```
Input Parameters: p, t Output
Parameters: None
rabin_karp_search(p, t) {
        m = p.length
        n = t.length
        q = prime number larger than m
        r = $2^{m-1}$ mod q
        // computation of initial remainders
        f[0] = 0
        pfinger = 0
        for j = 0 to m-1 {
                f[0] = (2 * f[0] + t[j]) mod q
                pfinger = (2 * pfinger + p[j]) mod q
        }
        i = 0
        while ($i + m \leq n$) {
                if (f[i] == pfinger)
                        if (t[i..i+m-1] = = p) // this comparison takes time O(m)
                                return i
                f[i + 1] = 2 * (f[i]- r * t[i]) + t[i + m] mod q
                i = i + 1
        }
        return -1
}
```

1.1 Trace the algorithm on pattern “101” and text “011010011001110” with *q=2.* Howmany comparisons between the pattern and text symbols are made?

1.2 Trace the algorithm on pattern “101” and text “011010011001110” with *q=5.* Howmany comparisons between the pattern and text symbols are made?

1.3 Show that the worst case running time of the algorithm is *O(m(n-m+1)).*

2. The following is a pseudocode of the algorithm that computes the shift table for a pattern p, to be used in Knuth–Morris–Pratt(KMP) algorithm; *shift[k]*, $k \in [-1, \ldots, p.length-1]$, is the smallest positive integer s such that $p[0\,..\,k-s] = p[s\,..\,k]$.

```
Input Parameters: p
Output Parameters: shift
knuth_morris_pratt_shift(p,shift) {
        m = p.length
        shift[-1]=1 // if p[0] ≠ t[i] we shift by one position
        shift[0]=1
        i = 1
        j = 0
        while (i + j < m)
                if (p[i+j] = = p[j]) {
                        shift[i+j]=i
                        j = j + 1
                }
                else {
                        if (j = = 0)
                                shift[i]=i+1
                        i = i + shift[j-1]
                        j = max(j-shift[j-1],0)
                }
}
```

2.1 Trace the algorithm on pattern “pappar”.

2.2 Trace the algorithm on pattern “ababbcabab”.

2.3 Show that *knuth_morris_pratt_shift* algorithm correctly computes the shift array in time *O(m)*.