CS3230 Assignment 6

Zhuang Jianning
A0214561M

1. In the original naive string-matching algorithm, after checking if P[1..m] == T[s+1.. s+m] we always increment s by 1 to shift the block of m characters in T to the right by 1.

However, when all characters in P are distinct, if a mismatch occurs at the jth element P[j] != T[s+j], we can shift the block to the right by j-1. Since we know P[1] will be distinct from T[s+2 .. s+j-1] as they matched with the other distinct characters in P.

STRING-MATCHING (P, T) * 1-indexed

```
n = T.length
m = P.length
s = 0
while s <= n-m
    for j = 1 to m
        if P[j] != T[s+j]
            break
    if j == m+1
        print pattern found with shift s
    else if j == 1
        s += 1
    else
        s += j-1
```

Since each element in T is compared to an element in P at most twice when P[1] compares with the mismatched jth element in the previous T block again, running time of algorithm is $\underline{O(n)}$ since element comparison is O(1)

2. This seems to be a variation of the 3SUM problem

Determine if there exists $a \in A$, $b \in B$, $c \in C$ st $a+b-c = 2022$

we could store all values of $2022 + c$ in a hash table

Then generate all pairs of elements from A and B

For each pair (a, b) where $a \in A$, $b \in B$, check if $(a+b)$ exists in hash table

```
3SUM(A, B, C)
  hash table H
  for i = 1 to n
     H.add(2022 + C[i])

  for j = 1 to n
    for k = 1 to n
       if H.query(A[j] + B[k])
          return true

  return false
```

since all arithmetic operations and hash functions on array elements can be performed in constant time, adding the n elements from C to hash table takes $O(n)$ time. Generating all n^2 pairs of (a, b) and querying from hash table takes $O(n^2)$ time

Algorithm has total expected runtime $O(n^2)$ //

3.

Prove by induction that $P[X = x_i] = \frac{1}{k}$ for all $i \in \{1 .. k\}$ for all $k \geq 1$

Base care: After receiving first number x_1, $k = 1$

$P[X = x_1] = 1 = \frac{1}{1}$ //

After receiving second number x_2, $k = 2$

$P[X = x_1] = 1 \cdot (1 - \frac{1}{2}) = \frac{1}{2}$ //

$P[X = x_2] = 1 \cdot (\frac{1}{2}) = \frac{1}{2}$ //

Inductive step: Assume after receiving nth number x_n

$P[X = x_i] = \frac{1}{n}$ for all $i \in \{1 .. n\}$

when receiving $(n+1)$th number,

$P[X = x_i] = P[X = x_i \text{ before } (n+1)\text{th number}] \times P[\text{unchanged}]$

$= \frac{1}{n} \times (1 - \frac{1}{n+1})$

$= \frac{1}{n} \times \frac{n}{n+1} = \frac{1}{n+1}$ //

$P[X = x_{n+1}] = \sum_{i=1}^{n} P[X = x_i] \cdot \frac{1}{n+1}$

$= \frac{1}{n+1} \sum_{i=1}^{n} \frac{1}{n}$

$= \frac{1}{n+1}$ //

conclusion: At every stage for all $k \geq 1$, $P[X = x_i] = \frac{1}{k}$ for all $i \in \{1 .. k\}$

X is uniformly sampled from numbers seen.