Question 1

Using prime numbers for 2-D pattern matching

Some Results that we know are:

- $M(\varepsilon) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ *(i)*
- $M(0) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ *(ii)*
- $M(1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ *(iii)*
- For non-empty strings x and y, $M(xy) = M(x)M(y)$ *(iv)*
- $M(x)$ is defined for all $x \in \{0,1\}^*$ *(v)*
- $M(x) = M(y) \Rightarrow x = y$ *(vi)*

(a)

Note that $M(z)$ a string z of length k can be calculated recursively as:

Algorithm 1:

```
1 def compute_M(z, k):
2     if(k = 1)
3         if(z[0] = 0)
4             return M(0)
5         else
6             return M(1)
7     else
8         return M(z[0])*compute_M(z+1, k−1)
```

Time Complexity of above algorithm $= O(k)$. So now we have a method to compute fingerprint in form of 2×2 matrix, in $O(k)$ where k is length of string. First we compute figerprint of pattern in $O(m)$ time, and the we compute fingerprint for all continuous strings of length m ($n - m + 1$ such strings) in text. and then we comapare these $n - m + 1$ fingerprints with fingerprint of pattern. By result (vi) we know that if fingerprint are equal, then so is the actual string. Now we need to find an efficient way to compute fingerprint of all the strings of length m in text.

Computation of M for all string in text of length m

let text (t) = $t_1 t_2 \ldots . t_n$, M_i be the fingerprint(M) of $t[i, m + i - 1]$. Now how can we compute M_{i+1} efficiently from M_i? Note that M_{i+1} is fingerprint of $t[i+1, m+i] = t_{i+1}t_{i+2} \ldots t_{m+i}$, and $t[i, m+i-1] = t_i t_{i+1} \ldots t_{m+i-1}$.

$$M_{i+1} = M(t_{i+1})M(t_{i+2}) \ldots M(t_{m+i})$$

$$M_1 = M(t_i)M(t_{i+1}) \ldots M(t_{m+i-1})$$

hence we get:

$$M_{i+1} = M(t_i)^{-1} * M_i * M(t_{m+i})$$

Since any operation involving aithematics is assumed to be in $O(1)$, the above computation can be done in constant time, with memoization.When we have the fingerprint for all M_i, we just compare them with fingerprint of pattern, and report if we find a collision.

Algorithm 2:

```
1  T := text of length n
2  P := pattern of length m
3  M[1:n−m+1] := list to store fingerprints of T
4  N[0:1] := list to store fingerprint of base case
5
6  N[0] = {1, 0, 1, 1} # base case
7  N[1] = {1, 1, 0, 1} # base case
8
9  M_P := compute_M(P, m)
10 M[1] = compute_M(T, m)
11
12 for i in (2,n−m+1)
13     M[i] = matrix_inverse(N[ T[ i − 1 ] ]) ∗ M[i − 1] ∗ N[ T[ m + i − 1 ] ]
14
15 for i in (1,n−m+1)
16     if (M[i] = M_P)
17         return i
```

Time Complexity:

Note that compute_M takes $O(k)$ time for a string of length k. So computation of M_P would take $O(m)$ time, similar to this computation of M[1] would also take $O(m)$ time. Since computation of M[i] from M[i - 1] takes $O(1)$ time, time complexity of computing M[] is $O(n-m)$, and in the end we just do a $O(1)$ comparison $n-m+1$ time.

$$T_{n,m} = O(m) + O(m) + O(n-m) + O(n-m+1)$$

$$T_{n,m} = O(2n+1) \approx O(n+m) \text{(since m < n)}$$

(b)

Let $M(P)$ be the fingerprint of pattern. The only case when the algorithm in (b) will give wrong answer, is when the pattern and string don't match, but $M_i = M(P)$. Or $M_i - M(P) = 0 \mod p$, which means that all the 4 entries of the matrix $M_i - M(P) = M(E)$ are zero modulo p.
So the event when error will occur is defined as: $M(E)_{11}, M(E)_{12}, M(E)_{21}, M(E)_{22} = 0 \mod p$ and P, T[i:i+m-1] don't match.
Let $P(\delta)$ denote the error probability (note that below equalities are given $\mod p$)

$$P(\delta) = P(M(E)_{11} = 0 \cap M(E)_{12} = 0 \cap M(E)_{21} = 0 M(E)_{22} = 0)$$

$$P(\delta) \leq P(M(E)_{11} = 0) \text{ since } \delta \text{ is a subset of this event}$$

Let $\pi(x)$ denotes the no of primes less than or equal to x.
Let $X(a)$ be the no of primes which divide a.
Clearly,

$$a \geq 2^{X(a)}$$

Thus,

$$X(a) \leq \log(a)$$

Lemma 1: Probability that $A \ mod \ p = 0$ when p is a random prime no from $(2, t)$ is less than or equal to $\frac{log(A}{\pi(t)}$.

Proof. Since p is selected randomly uniformly from all primes from $(2, t)$,

$$P(A \ mod \ p = 0) = \frac{X(A)}{\pi(t)}$$

Thus,

$$P(A \ mod \ p = 0) = \frac{\log(A)}{\pi(t)}$$

□

Note that $F_n < 2^n$ where F_n is n^{th} finbonacci number, also the terms of $M(Z)$ are bounded by F_n (given in problem).
we know:

$$M(Z)_{11} < 2^n$$

By above lemma we know that:

$$P(\delta) \leq \frac{\log(2^n)}{\pi(t)}$$

$$P(\delta) \leq \frac{n}{\pi(t)}$$

Now,

$$\pi(t) \approx \frac{t}{log(t)}$$

Thus,

$$P(\delta) \leq \frac{n * log(t)}{t}$$

Take $t > 5 * n^5 log(n)$. Then,

$$P(\delta) \leq n^{-4}$$

Thus for $t > 5 * n^5 log(n)$, probability that fingerprint will match, provided actual string don't match, would be less that n^{-4}.

(c)

In this problem we use the fingerprinting approach, we compute fingerprint of matrices of size $m \times m$, same way as we did above.First we flatten the 2D matrix into a 1D matrix of size m^2, by appending 2^{nd} row after 1^{st} row, and so on. This computation for each matrix in text of size $m \times m$, needs to be computed efficienty, to achieve $O(n^2)$ Complexity.
Note that here also our fingerprint is a 2D matrix of size 2×2.

Algorithm 1:

```
T := n x n matrix in which we need to search for pattern
P := pattern (dimension m x m)
p := prime number choosen uniformly and randomly from (2, t)
N[0:1] := list to store fingerprint of base case

N[0] = {1, 0, 1, 1} # base case
N[1] = {1, 1, 0, 1} # base case

# first we compute the fingerprint of all continuous strings of length m in
# each row (n−m+1 such string in each row)
F := 2D list to store above fingerprint
```

```
12 # F[i][j] stores fingerprint of string in ith row starting at index j
13
14 for i in (1,n)
15     F[i][1] = compute_M(T[i], m) mod p
16 for i in (1,n)
17     for j in (2,n−m+1)
18         F[i][j] = matrix_inverse(N[T[i][j − 1]]) ∗ F[i][j − 1] ∗ N[T[i][m + j − 1]] mod p
19
20 # store fingerprint of pattern
21 P_new = flatted(P)
22 M_P = compute_M(P_new, m∗m) mod p
23
24 # Now we compute fingerprint of all continuous matrices of size m x m using F[i][j]
25 G := 2D list to store this fingerprint
26
27 # G[i][j] denote fingerprint of matrix with left top at i,j and right bottom at i+m−1, j+m−1
28 # Since fingerprint is calculated by flattening, hence G[i][j] = F[i][j]∗F[i+1][j]∗...F[i+m−1][j]
29
30 # we do a column wise traversal
31 for j in (1, n−m+1)
32     G[1][j] = F[1][j]∗F[2][j]∗...F[m][j] mod p
33 for j in (1, n−m+1)
34     for i in (2, n−m+1)
35         G[i][j] = matrix_inverse(F[i − 1][j]) ∗ G[i − 1][j] ∗ F[i+m−1][j] mod p
36
37 # fingerprint matching
38 for i in (1, n−m+1)
39     for j in (1, n−m+1)
40         if (G[i][j] = M_P)
41             return (i,j)
```

Time Complexity

Computing matrx F[i][1] for all $i \in (1, n)$ would take $O(m) * n$ time. Computing rest of the entries of F would take $O(1)$ time per entry, so a total of $O(n*(n-m))$. Calculation of M for P would take $O(m^2)$ time. Computing G[1][j] for all $j \in (1, n-m+1)$ would take $O(m) * (n-m+1)$ time. And lastly computation of remaining entries of G would take $O(1)$, for each entry, so total $O((n-m) * (n-m+1))$. So in total we get

$$T_{n,m} = O(mn) + O(n(n-m)) + O(m^2) + O(m*(n-m+1)) + O((n-m)*(n-m+1))$$

Or,

$$T_{n,m} = O(2n^2 + m^2 - mn + n - 2m) \approx O(n^2) \text{ since m < n}$$

Error Probability

Error analysis is similar to previous problem, the only thing that changes is that now $M[i][j]_{11}$ would be bounded by F_{n^2}, since we are flattening 2D square matrix.
Note that $F_n < 2^n$ where F_n is n^{th} finbonacci number, also the terms of $M[i][j]$ are bounded by F_{n^2}.
we know:

$$M(Z)_{11} < 2^{n^2}$$

By above lemma we know that:

$$P(\delta) \leq \frac{\log(2^{n^2})}{\pi(t)}$$

$$P(\delta) \leq \frac{n^2}{\pi(t)}$$

Now,

$$\pi(t) \approx \frac{t}{log(t)}$$

Thus,

$$P(\delta) \leq \frac{n^2 * log(t)}{t}$$

Take $t > 10 * n^6 log(n)$. Then,

$$P(\delta) \leq n^{-4}$$

Thus for $t > 10 * n^6 log(n)$, probability that fingerprint will match, provided actual matrices don't match, would be less that n^{-4}.

Question 2

(1) Let a_k denote the element of S inserted during k^{th} iteration in the above algorithm. Let ε_k be the event that a_k is inserted in table T_2. Calculate $P(\varepsilon_k)$.
(2) Let X_k denote the random variable defined as follows. If a_k is stored in T_1, then $X_k = 0$; otherwise, it is the number of elements among $\{a_1, ..., a_{k-1}\}$ that collide with it in T_2. Provide a suitable upper bound on $E[X_k|E_k]$.
(3) Use (2) to show that we can design a perfect hashing (at most one element in any location of T_1 and T_2) using n which is asymptotically much smaller than s^2.

(1)

Probability of event ε_k would be the same as the probability of the event that a_k collides with at least one of $a_1, a_2, ..., a_{k-1}$ under hash function h_1. Call this event to be E.
Call the event that a_k collides with a_i to be E_i where $1 \leq i \leq (k-1)$ Now,

$$P(E) = P(\bigcup_{i=1}^{k-1} E_i)$$

Since h_1 is selected randomly uniformly from universal hash family H we get for any $a, b \, \epsilon \, U$,

$$P(h_1(a) = h_1(b)) \leq \frac{c}{n}$$

Thus,

$$P(E_i) \leq \frac{c}{n}$$

By union theorem,

$$P(E) \leq P(\sum_{i=1}^{k-1} E_i)$$

$$P(E) \leq \frac{(k-1)c}{n}$$

also we have $P(\varepsilon_k) = P(E)$. Hence we get,

$$P(\varepsilon_k) \leq \frac{(k-1)c}{n}$$

(2)

Given that X_k denote the random variable defined as follows:

- If a_k is stored in T_1, then $X_k = 0$
- else, it is the number of elements among $\{a_1, \ldots a_{k-1}\}$, that collide with it in T_2.

Let us assume that x elements are present in T_2 just before inserting a_k. Lets calculate the expected number of collision that a_k will have in T_2. Let Y_i ($i \in (1, x)$) be a random variable defined as follows:

$$Y_i = \begin{cases} 1 & \text{if element } z_i \text{ collides with } x_k \text{ in } T_2 \\ 0 & \text{otherwise} \end{cases}$$

let $Y = \sum_{i=1}^{x} Y_i$, then we need to calculate $E[Y]$. By the defination of perfect hashing function we know that:

$$P(h_2(a) = h_2(b)) \leq \frac{c}{n}$$

, so now by applying linearity of expectation we get that $E[Y] \leq \frac{xc}{n}$.

So we can say that $E[X_k|\xi_k] \leq \frac{E[x]c}{n}$. Now problem reduces to calculate $E[x]$, which can be done using previous result. Let Z_i ($i \in (1, k-1)$) be a random variable defined as follows:

$$Z_i = \begin{cases} 1 & \text{if element } x_i \text{ moves to } T_2 \\ 0 & \text{otherwise} \end{cases}$$

Z_1 is random variable corresponding to first element, and similarly for other values. So by linerity of expectation, we get $E[X] \leq \sum_{i=1}^{k-1} \frac{(i-1)c}{n}$ (using previous part of problem).

$$E[X_k|\xi_k] \leq \frac{E[x]c}{n}$$

$$E[X_k|\xi_k] \leq \frac{c}{n} \sum_{i=1}^{k-1} \frac{(i-1)c}{n}$$

$$E[X_k|\xi_k] \leq \frac{c^2(k-1)(k-2)}{2n^2}$$

(3)

Question 3

Consider a collection $X_1, ..., X_n$ of n independent geometrically distributed random variables with expected value 2. Let $X = \sum_{i=1}^{N} X_i$ and $\delta > 0$
(1) Derive a bound on $P(X \geq (1+\delta)(2n))$ by appyling the Chernoff bound to a sequence of $(1+\delta)(2n)$ fair coin tosses.
(2) Directly derive a Chernoff like bound on $P(X \geq (1+\delta)(2n))$ from scratch.
(3) Which bound is better?

Let Y be a geometrically distributed random variable then $P(Y = k) = (1-p)^{k-1}p$ and $E[Y] = \frac{1}{p}$.

(1)

Since $E[X_i] = 2$ we find that each X_i is geometrically distributed with parameter $p = \frac{1}{2}$.
As $p = \frac{1}{2}$ we can think of each X_i as the number of coin tosses required to get the first head.
So their sum $X = \sum_{i=1}^{n} X_i$ can be thought of as the number of coin tosses required to get n heads.
We have to find the probability that no less than $(1+\delta)(2n)$ coin tosses are required to get n heads, this is same as the probability that less than n heads appear in $(1+\delta)(2n)$ coin tosses which in turn is same as the probability that no less than $n(1+2\delta)$ tails appear in $(1+\delta)(2n)$ coin tosses
Let Y_i be the random variable taking value 1 when i^{th} coin toss results in a tail and 0 otherwise. Each Y_i is a Bernoulli random variable with probability equal to $\frac{1}{2}$
Consider $Y = \sum_{i=1}^{(1+\delta)(2n)} Y_i$. By linearity of expectation, $E[Y] = (1+\delta)n$
Now,

$$P(X \geq (1+\delta)(2n)) = P(Y \geq n(1+2\delta))$$

$$P(Y \geq n(1+2\delta)) = P\left(Y \geq \frac{(1+\delta)n(1+2\delta)}{1+\delta}\right)$$

$$P(Y \geq n(1+2\delta)) = P\left(Y \geq (1+\delta)n\left(1+\frac{\delta}{1+\delta}\right)\right)$$

$$P(Y \geq n(1+2\delta)) = P\left(Y \geq E[Y]\left(1+\frac{\delta}{1+\delta}\right)\right)$$

Applying chernoff bound on Y,

$$P(Y \geq n(1+2\delta)) \leq \frac{e^{n\delta}(1+\delta)^{n(1+2\delta)}}{(1+2\delta)^{n(1+2\delta)}}$$

$$P(Y \geq n(1+2\delta)) \leq \left(\frac{e^{\delta}(1+\delta)^{(1+2\delta)}}{(1+2\delta)^{(1+2\delta)}}\right)^n$$

Thus we get,

$$P(X \geq (1+\delta)(2n)) \leq \left(\frac{e^{\delta}(1+\delta)^{(1+2\delta)}}{(1+2\delta)^{(1+2\delta)}}\right)^n$$

(2)

$$P(X \geq (1+\delta)(2n)) = P(e^{tX} \geq e^{t(1+\delta)(2n)})$$

By Markov's inequality,

$$P(e^{tX} \geq e^{t(1+\delta)(2n)}) \leq \frac{E[e^{tX}]}{e^{t(1+\delta)(2n)}}$$

Now,

$$E[e^{tX}] = E[e^{(tX_1+...+tX_n)}]$$

$$E[e^{tX}] = E[e^{tX_1}e^{tX_2}...e^{tX_n}]$$

$$E[e^{tX}] = E[\Pi_{i=1}^{n} e^{tX_i}]$$

We know that X_is are independent so e^{tX_i}s are also independent amd hence we get,

$$E[e^{tX}] = \Pi_{i=1}^{n} E[e^{tX_i}]$$

and

$$E[e^{tX_i}] = \sum_{j=1}^{\infty} e^{tj} P(X_i = j)$$

$$E[e^{tX_i}] = \sum_{j=1}^{\infty} e^{tj} (1-p)^{j-1} p$$

$$E[e^{tX_i}] = e^t p \sum_{j=1}^{\infty} \left(e^t(1-p)\right)^{j-1}$$

We had $p = \frac{1}{2}$. Take $t < \ln 2$ so that $e^t(1-p) < 1$.
By using the formula of summation of a GP,

$$E[e^{tX_i}] = e^t p \frac{1}{1 - e^t(1-p)}$$

$$E[e^{tX}] = E[\Pi_{i=1}^{n} e^{tX_i}]$$

$$E[e^{tX}] = \left(\frac{e^t p}{1 - e^t(1-p)}\right)^n$$

Thus we get,

$$P(e^{tX} \geq e^{t(1+\delta)(2n)}) \leq \frac{1}{e^{t(1+\delta)(2n)}} \left(\frac{e^t p}{1 - e^t(1-p)}\right)^n$$

$$P(e^{tX} \geq e^{t(1+\delta)(2n)}) \leq \left(\frac{e^t p}{(1 - e^t(1-p))(e^{2t(1+\delta)})}\right)^n$$

This is true for every value of $t < \ln 2$ and hence differentiating it to find the minimum value -

$$\frac{d}{dt}\left(\frac{e^t p}{(1 - e^t(1-p))(e^{2t(1+\delta)})}\right)^n = 0$$

$$\frac{d}{dt}\frac{e^t p}{(1 - e^t(1-p))(e^{2t(1+\delta)})} = 0$$

$$-(1 - e^t(1-p))(1+2\delta)(e^{-t(1+2\delta)}) + (e^{-t(1+2\delta)})(e^t(1-p)) = 0$$

$$e^t(1-p) = (1+2\delta)(1 - e^t(1-p))$$

$$e^t(1-p) = \frac{1+2\delta}{2+2\delta}$$

Put $p = \frac{1}{2}$,

$$e^t = \frac{1+2\delta}{1+\delta}$$

So we get after substituting this value of e^t,

$$P(e^{tX} \geq e^{t(1+\delta)(2n)}) \leq \left(\frac{(1+\delta)^{2+2\delta}}{(1+2\delta)^{1+2\delta}}\right)^n$$

So,

$$P(X \geq (1+\delta)(2n)) \leq \left(\frac{(1+\delta)^{2+2\delta}}{(1+2\delta)^{1+2\delta}}\right)^n$$

(3)

Let the first and second bounds are $B1$ and $B2$ respectively,

$$\frac{B1}{B2} = \frac{e^\delta}{1+\delta}$$

$$\frac{B1}{B2} \geq 1$$

Thus, the second bound is better.