

# CS648 Assignment 2

SWASTIK SHARMA (14741)

ANUBHAV SHRIVASTAVA (14114)

## 1. 2-Dimensional Pattern Matching

1.

**Notations :** Let  $T$  be the text bit-string and  $P$  the pattern bit-string. Size of  $T$  is  $n$  and  $P$  is  $m$ .

$M(P)$  is mapping of the pattern  $P$  to a  $2 \times 2$  matrix as given in the problem statement.

$M(P)^{-1}$  is the inverse matrix to  $M(P)$ . Let  $l$  and  $r$  be two indices in  $T$  s.t.  $1 \leq l \leq r \leq n$ .

$T[l, r]$  is the substring starting at index  $l$  and ending at  $r$ .

Consequently,  $M(T[l, r])$  is mapping of  $T[l, r]$  to  $2 \times 2$  matrix as given in the problem statement.

$$M(\epsilon) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad M(0) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad M(1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad M(0)^{-1} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \quad M(1)^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

**Algorithm :** Basically, we will compare  $M(P)$  with  $M(T[i, i + m - 1]) \forall i$  in  $[1, n - m + 1]$ .

We will calculate  $M(P)$  by right multiplying corresponding  $M(P[i])$  for  $i$  from 1 to  $m$  to an identity matrix. Similarly, we will calculate  $M(T[1, m])$ . After that to get,  $M(T[i, i + m - 1])$  for  $i > 1$ , we will take  $M(T[i - 1, i + m - 2])$ , right multiply  $M(T[i + m, i + m])$  and left multiply  $M(T[i - 1, i - 1])^{-1}$  to it.

The following pseudo code formalize the above algorithm.

```
pattern ← M(P);
current ← M(T[1, m]);
for i = 1 to n - m + 1
    if pattern == current
        print "Pattern present at index i";
    if i < n - m + 1
        current = multiply(current, M(T[i + m, i + m]));
        current = multiply(M(T[i, i])^{-1}, current);
```

**Order Analysis :**

Calculating  $M(P)$  takes  $O(m)$  time

Calculating  $M(T[1, m])$  takes  $O(m)$  time

Loop Runs  $n - m + 1$  times and at each iteration we do a  $O(1)$  computation. Hence loop takes  $O(n - m + 1)$  time.

Overall Complexity =  $O(n + m)$  time.

2.

**Notation :**

Let  $M$  be a matrix then  $M \bmod p$  denotes a matrix in which all entries of  $M$  are replaced by their modulo with  $p$ .

Hence, We perform the above algorithm but at each step, while multiplying two matrices, we take modulo with  $p$ .

Let  $M(A)$  and  $M(B)$  be two matrices which are mapping of string  $A$  and  $B$  of size  $m$ .

$\pi(t)$  = number of primes less than equal to  $t$ .

Now, if  $M(A) == M(B)$  then  $M(A) \bmod p == M(B) \bmod p$  then  $A == B$  which is true.

But if  $A != B$  and then  $M(A) != M(B)$  then it might be possible that  $M(A) \bmod p == M(B) \bmod p$ . which will imply that  $A == B$  which is not true.

Hence, this algorithm will fail iff  $M(A) != M(B)$  and  $M(A) \bmod p == M(B) \bmod p$ .

Let  $D = M(A) - M(B)$

$M(A) \bmod p == M(B) \bmod p$

$\Rightarrow M(A) - M(B) \bmod p == 0$  ( where 0 is  $2 \times 2$  null matrix)

$\Rightarrow D \bmod p == 0$

Hence, every entry of  $D$  is multiple of  $p$ , or conversely  $p$  is factor of every entry of  $D$ .

Now, take any entry of  $D$  (say  $x$ ). Now,  $x$  is bounded by  $m^{th}$  Fibonacci number which is bounded by  $2^m$ ;

Now, any number  $N$  has at most  $\log N$  factors hence  $x$  has at most  $m$  factors or at most  $n$  factors ( $m \leq n$ ).

Let say we choose  $p$  uniformly randomly from range  $[2, t]$  then probability that a factor of  $x$  was chosen was  $\frac{n}{\pi(t)}$

There are four elements of  $D$  and event that  $p$  is a factor of one of the element is independent of the event that  $p$  is a factor of other element of  $D$ .

Hence probability that  $p$  was a factor of all of the element  $\leq (\frac{n}{\pi(t)})^4$ .

We want this probability to be less than  $\frac{1}{n^4}$ .

Hence, we want  $\frac{n}{\pi(t)}$  to be less than  $\frac{1}{n}$ .

Putting  $t = n^2 \log n$  and  $\pi(t) = \frac{t}{\log t}$ .  
 We get this probability to be less  $\frac{1}{n^4}$ .

(As done in class)

3.

**Notation :**

Here we assume that all operation are accompanied by a modulo operation by p which is chosen in the begining.

Let S be 2-D bit-matrix and  $S(i, j)$  denotes the element of  $i^{th}$  row and  $j^{th}$  column.

$PrefixProduct(i, j) = \text{Product of all the } M(S(k, j)) \text{ s.t. } 1 \leq k \leq i.$

i.e.  $PrefixProduct(i, j) = M(S(1, j)) * M(S(2, j)) * \dots * M(S(i, j))$

$PrefixProductInverse(i, j) = \text{Matrix inverse of } PrefixProduct(i, j)$

i.e.  $PrefixProductInverse(i, j) = M(S(i, j))^{-1} * M(S(i-1, j))^{-1} * \dots * M(S(1, j))^{-1}$

Now let us assume there is a matrix X of size  $N \times M$  then we have map this matrix to  $2 \times 2$  matrix  $M(X)$

We define this map as follows:

If X is column matrix, cut this matrix into two submatrix by cutting at any row. Let U be upper column matrix and L be the lower one.

Then  $M(X) = M(U) * M(L)$

If X is not a column matrix, cut this at any column, let L be the submatrix left of this line and R be the one right to it. Then  $M(X) = M(L) * M(R)$

Now we have defined the new mapping M, we can state out algorithm.

**Algorithm :** Let T be the text  $n \times n$  2-D bit-matrix and P be the pattern  $m \times m$  2-D bit-matrix. First we show how to calculate  $PrefixProduct(i, j)$  and  $PrefixProductInverse$  for matrix T in  $O(n^2)$  time.

$PrefixProduct(1, j)$  and  $PrefixProductInverse(1, j)$  will be simply the M of the corresponding element. To calculate them for other rows just use the  $PrefixProduct(i, j)$  and  $PrefixProductInverse$  of just upper row.

Following pseudocode describes out algorithm :

for j = 1 to n

$PrefixProduct(1, j) = M(T[1, j])$

$PrefixProductInverse(1, j) = M(T[1, j])^{-1}$

for i = 2 to n

for j = 1 to n

$PrefixProduct(i, j) = PrefixProduct(i-1, j) * M(T[i, j])$

$PrefixProductInverse(i, j) = M(T[i, j])^{-1} * PrefixProductInverse(i-1, j)$

Also we calculate  $M(P)$  in  $O(m^2)$  time.

Initialize  $M(P) = I_{2 \times 2}$

for i = 1 to m

for j = 1 to m

$M(P) = M(P) * M(T(j, i))$

Now we have to calculate mapping of every submatrix of size m of matrix P. Let  $M(P(i, j))$  is the mapping of submatrix of size m ending at (i,j) in P.  $m \leq i, j \leq n$ . We will calculate this mapping in  $O(n^2)$  time as follows :

First we calculate  $M_c(i, j)$  which is product of all the  $M(P[k, j])$  such that  $i-m+1 \leq k \leq i$ ,  $m \leq i \leq n, 1 \leq j \leq n$

Also we will calculate  $M_c(i, j)^{-1}$  with it. We can calculate these both in  $O(n^2)$  as follows:

If  $i = m$

$M_c(m, j) = PrefixProduct(m, j)$

$M_c(m, j)^{-1} = PrefixProductInverse(m, j)$

else

$M_c(i, j) = PrefixProductInverse(i-m, j) * PrefixProduct(i, j)$

$M_c(m, j)^{-1} = PrefixProductInverse(i, j) * PrefixProduct(i-m, j)$

Now with help of this we can calculate  $M(P(i, j))$  in  $O(n^2)$  as follows :

if  $j = m$

Initialize  $M(P(i, j)) = I_{2 \times 2}$

for k = 1 to m

$M(P(i, j)) = M(P(i, j)) * M_c(i, k)$

else

$M(P(i, j)) = M_c(i, j-m)^{-1} * M(P(i, j-1)) * M_c(i, j)$

For  $n-m+1$  elements we take  $O(m)$  time and for rest of the elements we take  $O(1)$  time. Hence total time =  $O(m * (n-m+1)) + O((n-m+1) * (n-m+1)) = O(n^2)$

Hence we just go at each i,j s.t  $m \leq i, j \leq n$ , compare  $M(T)$  with  $M(P(i, j))$  in  $O(1)$  time which take overall  $O(n^2)$

## 2. How well did you internalize the proof of Chernoff bound ?

1. Probability that number of coin tosses required to get  $n$  heads is greater than  $2n(1 + \delta)$  is same as Probability of getting less than or equal  $n - 1$  heads out  $2n(1 + \delta) - 1$  coin tosses.

For given geometrically distributed random variable, we know that expected value is  $\frac{1}{p}$  which is given 2, hence  $p = \frac{1}{2}$ .

$$X_i = \begin{cases} 1 & \text{if head is obtained at } i_{th} \text{ iteration} \\ 0 & \text{otherwise} \end{cases}$$

$$E[X_i] = \frac{1}{2} * 1 + \frac{1}{2} * 0 = \frac{1}{2}.$$

$$X' = \sum_{i=1}^{2n(1+\delta)-1} X_i$$

$$E[X'] = n(1 + \delta) - \frac{1}{2}$$

Assuming  $n \gg \frac{1}{2}$ , we get  $E[X'] = n(1 + \delta)$

$$P(X' \leq (n - 1))$$

$$= P(X' \leq (n - 1) \frac{E[X']}{(1+\delta)n})$$

$$= P(X' \leq (1 - \frac{n\delta+1}{n(1+\delta)}) * E[X'])$$

$$\leq P(X' \leq (1 - \frac{\delta}{\delta+1}) * E[X']) \text{ (As } \frac{n\delta+1}{n(1+\delta)} > \frac{\delta}{\delta+1} \text{)}$$

$$= P(X' \leq (1 - \delta') * E[X']) \text{ By letting } \delta' = \frac{\delta}{\delta+1}$$

$$\leq (\frac{e^{-\delta'}}{(1-\delta')^{1-\delta'}})^{E[X']}$$

Putting value of  $\delta'$ , we get

$$\leq ((1 + \delta)e^{-\delta})^n$$

2. According to Markov's Inequality

$$P(X \geq a) \leq \frac{E[X]}{a} \text{ if } X(\omega) \geq 0 \forall \omega \in \Omega$$

$$\because X(\omega) \in \mathbb{N}$$

$$P(X \geq (1 + \delta)\mu)$$

$$= P(e^{tx} \geq e^{(1+\delta)t\mu}) \leq \frac{E[e^{tx}]}{e^{(1+\delta)t\mu}}$$

$$E[e^{tX}]$$

$$= E[e^{tX_1+tX_2+tX_3+\dots+tX_n}]$$

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

$$E[e^{tX_i}] = \frac{e^t}{2} + \frac{e^{2t}}{4} + \frac{e^{3t}}{8} \dots = \frac{e^t}{2(1-\frac{e^t}{2})} = \frac{e^t}{2-e^t}$$

$$E[e^{tX}] = \prod_{i=1}^n E[e^{tX_i}] = (\frac{e^t}{2-e^t})^n$$

Now  $\mu = 2n$

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

Now differentiating w.r.t  $t$  to get the *minima* we get

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

3.

**Claim** : Bound obtained from 2nd part is better

**Proof :** Bound obtained in 1st part (say  $b1$ ) =  $(\frac{1+\delta}{e^\delta})^n$

Bound obtained in 2nd part (say  $b2$ ) =  $(\frac{(1+\delta)^{2(1+\delta)}}{(1+2\delta)^{1+2\delta}})^n$

To prove  $\frac{b1}{b2} \geq 1$   $\frac{b1}{b2} = (\frac{(1+\delta)(1+2\delta)^{1+2\delta}}{((1+\delta)^{2(1+\delta)})e^\delta})^n$

Taking  $ln$  on both sides, we get

$$n(ln(1+\delta) - \delta - 2(1+\delta)ln(1+\delta) + (2\delta+1) * ln(1+2\delta)) \geq 0$$

$$(2\delta+1)(ln(\frac{1+2\delta}{1+\delta})) > \delta$$

$$ln(\frac{1+2\delta}{1+\delta}) = ln(1 + \frac{\delta}{1+\delta}) > \frac{\delta}{\delta+1} - \frac{\delta^2}{2(\delta+1)^2}$$

$$LHS > (2\delta+1)(\frac{\delta}{\delta+1} - \frac{\delta^2}{2(\delta+1)^2}) = \frac{\delta(2\delta+1)(\delta+2)}{2(\delta+1)^2} = \frac{\delta(2\delta^2+5\delta+2)}{(2\delta^2+4\delta+2)} > \delta$$

Hence proved.

### 3. Estimating biasness of a coin

Strategy: We toss the coin  $N$  times, Let  $H$  be the random variable denoting the number of heads in  $N$  coin tosses. We report  $\tilde{p}$  to be  $\frac{H}{N}$ .

Now, given  $\epsilon$ ,  $\delta$  and  $a$ , we will estimate the value of  $N$  so that the following probability hold.

$$\Pr[|p - \tilde{p}| > p\epsilon] < \delta$$

$$\implies \Pr[\tilde{p} < p(1-\epsilon)] + \Pr[\tilde{p} > p(1+\epsilon)] < \delta$$

$$\implies \Pr[N\tilde{p} < Np(1-\epsilon)] + \Pr[N\tilde{p} > Np(1+\epsilon)] < \delta$$

$$\implies \Pr[H < E[H](1-\epsilon)] + \Pr[H > E[H](1+\epsilon)] < \delta$$

$$\text{Now } \Pr[H < E[H](1-\epsilon)] + \Pr[H > E[H](1+\epsilon)]$$

$$< e^{-\frac{N\epsilon^2 p}{2}} + e^{-\frac{N\epsilon^2 p}{4}}$$

$$< 2e^{-\frac{N\epsilon^2 p}{4}}$$

$$< 2e^{-\frac{N\epsilon^2 a}{4}}$$

If we have this term to be less than  $\delta$  then our Probability condition will hold.

$$2e^{-\frac{N\epsilon^2 a}{4}} < \delta$$

$$\implies -\frac{N\epsilon^2 a}{4} < ln(\frac{\delta}{2})$$

$$N > 4 \frac{ln(\frac{\delta}{2})}{a\epsilon^2}$$