csci-5454-hw4

Jake Krol

September 2024

1 Question 1

Algorithm 1 FFT(a, n)

Details: a is an array with size n

if $n == 1$ **then**
    return a
end if
$a_{even} = ARRAY(n/2)$ ▷ Initialize array of length n/2
$a_{odd} = ARRAY(n/2)$
$a_{even} = a[0 : n : 2]$ ▷ Indexing syntax is start:stop:step
$a_{odd} = a[1 : n - 1 : 2]$
$A_{even} = FFT(a_{even}, \frac{n}{2})$
$A_{odd} = FFT(a_{odd}, \frac{n}{2})$
$A = ARRAY(n)$
$\omega_n = e^{\frac{i2\pi}{n}}$
$\omega_k = 1$
for $k = 0$ **upto** $\frac{n}{2} - 1$ **do**
    $A[k] = A_{even}[k] + \omega_k * A_{odd}$
    $A[k + \frac{n}{2}] = A_{even}[k] - \omega_k * A_{odd}$
    $\omega_k = \omega_n * \omega_k$
end for
return A

Algorithm 2 POLY-MULT(a,b,n,m)

Details: a and b are input coefficient arrays from polynomials with n and m degrees.

$size = n + m + 1$ ▷ max size of result is $n + m$ plus the 0th coefficient
$a = \mathit{PAD\text{-}ZEROS}(a, size)$ ▷ 0s will be append at indices n to $size - 1$
$b = \mathit{PAD\text{-}ZEROS}(b, size)$
$A = FFT(a, size)$
$B = FFT(b, size)$
$C = A \circ B$ ▷ Point-wise multiplication of DFTs
for $i \in C$ **do**
  $C[i] = \mathit{COMPLEX\text{-}CONJUGATE}(C[i])$ ▷ Flip sign of imag component
end for
$C = FFT(C, size)$
return C

Algorithm 3 SUM-EXISTS(A,B,C)

```
n = MAX(A)                      ▷ maximum element in A
m = MAX(B)
a = ARRAY(n)                    ▷ Initialize array of length n
b = ARRAY(m)
a = FILL(a, 0)                  ▷ fill array with zeros
a = FILL(a, 0)
for i ∈ A do
    a[i] = 1
end for
for i ∈ A do
    b[i] = 1
end for
c = POLY-MULT(a, b, n, m)
for i ∈ C do
    if c[i] == 1 then
        return TRUE
    end if
end for
return FALSE
```

As hinted, testing whether any sum of elements between two sets A, B exists in a third set C (SUM-EXISTS(A,B,C)) can be done by multiplying one-hot coefficient polynomial representations (POLY-MULT(a,b,n,m)) of the A and B sets, then testing if the product polynomial if the index of any non-zero coefficients are in C. The runtime is $O(n \log n))$ since FFT (FFT(a,n)) is $O(n \log n)$, point-wise multiplication is $O(n)$, and inverse FFT of the point-wise product is $O(n \log n)$. The total cost, $2(n \log n) + n$, is asymptotic to $O(n \log n)$.

2 Question 2

The expected value of a discrete random variable, X, is the weighted sum of events and their probabilities. For a geometric random variable this is

$$E[X] = \sum_{k=1}^{\infty} k(1-p)^k p = \mathbf{p} \sum_{\mathbf{k=1}}^{\infty} \mathbf{k}(\mathbf{1}-\mathbf{p})^{\mathbf{k}}$$

Using the hint, we consider the function $f(p) = \sum_{k=1}^{\infty}(1-p)^k$. This is a geometric series with $0 < k < 1$; therefore, there exists an analytical solution.

$$\sum_{\mathbf{k=1}}^{\infty}(\mathbf{1}-\mathbf{p})^{\mathbf{k}} = \frac{1}{1-(1-p)} = \frac{\mathbf{1}}{\mathbf{p}} - \mathbf{1}$$

Taking the derivative of each side shows the left-hand side corresponds to part of our expected value function. While, the derivative of the right-hand side, offers a simple expression for substitution.

The chain rule can be applied to the left-hand side. Let $u(v) = v^k$, $v(p) = 1 - p$.

$$\frac{du}{dv}(v^k) = k(1-p)^{k-1}$$
$$\frac{dv}{dp}(1-p) = -1$$

The resulting derivative for $u'(v(p)) * v'(p)$ is

$$u'(v(p)) * v'(p) = \mathbf{-k(1-p)^{k-1}}$$

This corresponds to negative form of the summation term in the expected value equation.

The derivative of $\frac{1}{p} - 1$ is simply $-\mathbf{\frac{1}{p^2}}$ by the power rule.

The left-hand derivative can be substituted into the equation expected value:

$$-E[X] = p\sum_{k=1}^{\infty} -k(1-p)^k$$
$$-E[X] = p(-\frac{1}{p^2})$$
$$\mathbf{E[X] = \frac{1}{p}}$$

3 Question 3

3.1 Question 3a

$p :=$ probability of heads, $q :=$ probability of tails $(1-p)$, $H :=$ event of heads, $T :=$ event of tails.

The algorithm recurses until the two TOSS-COIN() results are in disagreement (i.e., heads and tails or tails and heads). It can be shown that the return value is a uniform sample of heads and tails by considering independence of TOSS-COIN outcomes and conditional probabilities.

Algorithm 4 TOSS-COIN(p)

Details: Return HEADS with p probability and TAILS with $1 - p$ probability

return $RANDOM(HEADS, TAILS, p)$

Algorithm 5 FAIR-COIN(p)

```
x = TOSS-COIN(p)
y = TOSS-COIN(p)
if x ≠ y then
    return y
else
    return FAIR-COIN(p)
end if
```

$$Pr[x = H \wedge y = T] = pq$$
$$Pr[x = T \wedge y = H] = qp$$
$$Pr[x \neq y] = pq + qp = 2pq$$

Therefore, given the events are independent, it can be shown that the **probability of two coin tosses with disagreeing outcomes occurs with a probability of $\frac{1}{2}$**.

$$Pr[x = H \wedge y = T | x \neq y] = Pr[x = T \wedge y = H | x \neq y] = \frac{pq}{2pq} = \frac{qp}{2pq} = \frac{1}{2}$$

3.2 Question 3b

Calculating the conditional expectation of disagreeing and agreeing coin tosses can show the expected runtime is $\frac{1}{pq}$.

Consider the two mutually exclusive events where the results of the two coins either agree ($x = y$ with probability $1 - 2pq$) or disagree ($x \neq y$ with probability $2pq$). If T is a random variable representing the number of coin flips until return, then T can be represented by two events $x = y$ and $x \neq y$. These two events encompass the entire sample space for our algorithm, and the expectation of T is the weighted sum of conditional expectations for these two events.

$$E[T] = E[T|x = y] \cdot Pr[x = y] + E[T|x \neq y] \cdot Pr[x \neq y]$$

If $x \neq y$ occurs, then the algorithm returns: $E[T|x \neq y] = 2$.

If the $x = y$ occurs, then the algorithm continues tracking the cost of the initial 2 tosses: $E[T|x = y] = 2 + E[T]$

Plugging in the conditional expectation for disagreeing coin tosses and the probabilities for each event shows that the expected number of coin tosses until FAIR-COIN returns is $E[T] = \frac{1}{pq}$.

$$
\begin{aligned}
E[T] &= (2 + E[T]) \cdot (1 - 2pq) + 2 \cdot 2pq \\
E[T] &= (2 + E[T])(1 - 2pq) + 4pq \\
E[T] &= 2 - 4pq + E[T] - 2pqE[T] + 4pq \\
E[T] &= 2 + E[T] - 2pqE[T] \\
0 &= 2 - 2pqE[T] \\
\frac{\mathbf{1}}{\mathbf{pq}} &= \mathbf{E[T]}
\end{aligned}
$$

4 Question 4

4.1 Question 4a

The probability of the router receiving a packet from the ith server is $Pr[i] = \frac{1}{n}$ since the distribution is uniform. The complementary case where the packet does not come from the ith server is $Pr[\neg i] = \frac{n-1}{n}$. Assuming independence of events, the probability of not receiving a packet from the ith server $m = 10n\ln(n)$ times is $Pr[\neg i]^m = (\frac{n-1}{n})^m$.

The hint of $1+x \leq e^x$ allows a demonstration that $Pr[\neg i]^m$ is upper bounded by n^{-10}.

Rewrite $\frac{n-1}{n}$ to $1 - \frac{1}{n}$, substitute $-\frac{1}{n}$ for x in the hint, and then substitute $m = 10n\ln(n)$.

$$
\begin{aligned}
1 - \frac{1}{n} &\leq e^{-\frac{1}{n}} \\
(\mathbf{1} - \frac{\mathbf{1}}{\mathbf{n}})^{\mathbf{m}} &\leq e^{(-\frac{1}{n})^m} \\
&\leq e^{-\frac{m}{n}} \\
&\leq e^{-\frac{m}{n}} \\
&\leq e^{-\frac{-10n\ln(n)}{n}} \\
&\leq e^{-10\ln(n)} \\
&\leq e^{(\ln(n))^{-10}} \\
&\leq \mathbf{n^{-10}}
\end{aligned}
$$

4.2 Question 4b

By using the inequality from the event in Q4a and taking the complement of the union of events, the probability of receiving at least one packet from all n servers after m seconds is $\geq \mathbf{1} - \frac{\mathbf{1}}{\mathbf{n^9}}$.

Q4a showed the $Pr[\neg i]^m \leq \frac{1}{n^{10}}$, and the complement of this event is $(1 - Pr[\neg i]^m)$. Therefore, the probability of at least one server not sending a packet after m seconds for n servers is $\cup_i^n Pr[\neg i]^m$; the union is applied to consider the event where a packet is not received from server i or server $i+1$ or $\ldots n$th server after m seconds. The complement of the union represents the event where at least one packet is received from all servers after m seconds:

$$1 - \cup_i^n Pr[\neg i]^m =$$
$$1 - \sum_i^n Pr[\neg i]^m =$$
$$1 - nPr[\neg i]^m$$

Taking the inequality from Q4a, multiplying both sides by $-n$ and adding 1 shows:

$$\mathbf{1 - nPr[\neg i]^m \geq 1 - \frac{1}{n^9}}$$