CIS 677

Homework 3

Daniel Sparber

Collaborators: Julian Schnitzler & Seyoung Kim

Problem 1

Let S denote the stream. Let a_i be the number of occurrences of (A, i) in the stream S. Let b_i be the number of occurrences of (B, i).

Consider the following polynomials:

$Q_A(x) = \sum_{i=1}^{n} a_i x^i$

$Q_B(x) = \sum_{i=1}^{n} b_i x^i$

$Q_C(x) = Q_A(x) - Q_B(x) = \sum_{i=1}^{n} \underbrace{(a_i - b_i)}_{c_i} x^i = \sum_{i=1}^{n} c_i x^i$

By definition of a_i, we have $Q_A(x) = \sum_{(A,i)\in S} 1 \cdot x^i$. Similarly, $Q_B(x) = \sum_{(B,i)\in S} 1 \cdot x^i$.

Thus, given x we can evaluate the polynomials as we go over the stream S.

<u>Algorithm</u>

Let p be a the smallest prime number greater n^2 and greater $100n$.

First, draw a random number $r \in [2...p-1]$.

As we move along the stream, evaluate $Q_A(r)$ and $Q_B(r)$ over the field $\mathbb{F}_p$.

Calculate $Q_C(r) = Q_A(r) - Q_B(r)$. Return $A = B$ if $Q_C(r) = 0$, otherwise $A \neq B$.

<u>Space complexity</u>

The algorithm only needs to store p, r and the current evaluation of $Q_A(r)$ and $Q_B(r)$ over $\mathbb{F}_p$.

Since, $p \in \mathcal{O}(n^2)$, p can be stored with $\mathcal{O}(\log n^2) = \mathcal{O}(\log n)$ bits. Since $r < p$ and $Q_A(r)$, $Q_B(r)$ are evaluated over $\mathbb{F}_p$, all values can also saved with $\mathcal{O}(\log n)$ bits.

<u>Correctness</u>

Let S be an arbitrary stream. Let A and B be the multi-sets implied by S.

- Case $A = B$

  By definition of the polynomials, we get $Q_A = Q_B$. Therefore, $Q_C = 0$. Thus, the algorithm always yields $A = B$.

- Case $A \neq B$

  From number theory we know, $\forall x \in [2...p-1]$, for any $i, j \in [1..n]$, $i \neq j \Rightarrow x^i \neq x^j$ over $\mathbb{F}_p$, since $p > n$.

  Since $A \neq B$, there exists at least one $c_i \neq 0$. Because $p > 100n$ and $c_i \leq 100n$, c_i is also non zero over $\mathbb{F}_p$. Furthermore, if there are multiple non zero c_i, they do not cancel out, since $x^i \neq x^j$ for $i \neq j$. Therefore, the polynomial Q_C is non zero.

  Because Q_C is non zero, we can apply the Schwartz-Zippel Lemma and get:

  $\Pr[Q_C(r) = 0] = \Pr[Q_C(r) = 0 \mid Q_C \neq 0] \leq \frac{d}{|[2...p-1]|} = \frac{n}{n^2} = \frac{1}{n}$.

  Thus, the algorithm returns the correct answer with probability at least $1 - 1/n$.

Problem 2

Part (a)

To compute all $\lambda(v)$ we use the following approach:

First, we compute all strongly connected components in $G(V, E)$ with a DFS based algorithm (e.g. Tarjan).

By doing this we get a graph G' of strongly connected components. Each node in G' represents a strongly connected component. G' is acyclic (if G' had a cycle, all components on this cycle would be strongly connected, thus contradicting the construction of G').

For every strongly connected component C, we compute a sorted list L_C of $r(v)$ values ($v \in C$). We store at most t elements in this list. This list can be constructed by inserting the nodes of the component into a linked list L_C. We use linear search to find the correct position. If the linear search visits more than t, we abort the search and do not insert the node into the list.

Since G' is acyclic, we compute the topological order of G' with a modiefied DFS. We traverse the components in reverse topological order (i.e. starting at a sink). For every component we merge the sorted lists L_C of the component with the lists of all direct successors and assign the merged list to L_C. When merging we only keep the first t values.

For every component C and every $v \in C$, we set $\lambda(v)$ to the t-th element of the list L_C.

<u>Runtime</u>

We perform two DFS based algorithms over the entire graph. The runtime for this is bounded by $\mathcal{O}(n + m)$. Furthermore, every node $v \in V$ is inserted exactly once into one L_C. Inserting uses at most t steps (linear search). Thus, inserting all elements is bounded by $\mathcal{O}(n \cdot t)$. We merge at most m lists and merging takes t steps, yielding a bound of $\mathcal{O}(m \cdot t)$. Getting the t-th element for all nodes and saving all $\lambda(v)$ is also bounded by $\mathcal{O}(n \cdot t)$.

The total runtime is bounded by $\mathcal{O}((n + m) \cdot t)$.

<u>Correctness</u>

For any strongly connected component C, $\lambda(v)$ is the same for all $v \in C$, since the same nodes are reachable for every node in C. The reachable nodes for all $v \in C$ are the nodes in C and all nodes, that are in any successor component of C.

Thus, the algorithm yields the correct $\lambda(v)$ $\forall v \in V$ by construction of the L_Cs.

Part (b)

Claim

$\forall v \in V : |S(v)| \geq t \Rightarrow (1-\varepsilon)|S(v)| \leq \frac{n \cdot t}{\lambda(v)} \leq (1+\varepsilon)|S(v)|$ with high probability.

Proof

Let $v \in V$ be arbitrary. Assume $|S(v)| \geq t$.

We show $\Pr\left[\frac{n \cdot t}{\lambda(v)} < (1-\varepsilon)|S(v)|\right] \leq \frac{1}{poly(n)}$ and $\Pr\left[\frac{n \cdot t}{\lambda(v)} > (1+\varepsilon)|S(v)|\right] \leq \frac{1}{poly(n)}$.

Thus, $\Pr\left[(1-\varepsilon)|S(v)| \leq \frac{n \cdot t}{\lambda(v)} \leq (1+\varepsilon)|S(v)|\right] \geq 1 - 2 \cdot \frac{1}{poly(n)} \mathrel{\widehat{=}} 1 - \frac{1}{poly(n)}$.

We define the following random variables:

$\forall u \in V : Y_u = 1$ if $r(u) \leq \frac{n \cdot t}{(1+\varepsilon) \cdot |S(v)|}$, we get $\mathbb{E}[Y_u] = \frac{t}{(1+\varepsilon) \cdot |S(v)|}$

$Y = \sum_{u \in S(v)} Y_u$, we get $\mathbb{E}[Y] = \frac{t}{1+\varepsilon}$

$$\Pr\left[\frac{n \cdot t}{\lambda(v)} > (1+\varepsilon)|S(v)|\right] = \Pr\left[\lambda(v) < \frac{n \cdot t}{(1+\varepsilon)|S(v)|}\right] = \Pr[Y \geq t]$$

Since $\forall u \in V$, $r(u)$ was drawn uniformly at random, we can use Chernoff bounds to estimate the probability. Furthermore, we assume $\varepsilon \leq 0.5$ (if the algorithm gets a larger ε as input, we just use $\varepsilon = 0.5$. This increases the run time only by a constant).

$$\Rightarrow \Pr[Y \geq t] = \Pr[Y \geq (1+\varepsilon)\mathbb{E}[Y]] \leq e^{\frac{-\mathbb{E}[Y] \cdot \varepsilon^2}{4}} = e^{\frac{-t \cdot \varepsilon^2}{(1+\varepsilon) \cdot 4}} \leq e^{\frac{-t \cdot \varepsilon^2}{2 \cdot 4}} =$$

$$= e^{\frac{-36 \cdot \log n \cdot \varepsilon^2}{8\varepsilon^2}} = e^{-4.5 \cdot \log n} = n^{-4.5} = \frac{1}{poly(n)}$$

Analogously, we get the probability for the other case:

$\forall u \in V : Y'_u = 1$ if $r(u) \leq \frac{n \cdot t}{(1-\varepsilon) \cdot |S(v)|}$, we get $\mathbb{E}[Y'_u] = \frac{t}{(1-\varepsilon) \cdot |S(v)|}$

$Y' = \sum_{u \in S(v)} Y_u$, we get $\mathbb{E}[Y'] = \frac{t}{1-\varepsilon}$

$$\Pr\left[\frac{n \cdot t}{\lambda(v)} < (1-\varepsilon)|S(v)|\right] = \Pr\left[\lambda(v) > \frac{n \cdot t}{(1-\varepsilon)|S(v)|}\right] = \Pr[Y' \leq t] = \Pr[Y \leq (1-\varepsilon)\mathbb{E}[Y']] \leq$$

$$\leq e^{\frac{-\mathbb{E}[Y'] \cdot \varepsilon^2}{2}} = e^{\frac{-t \cdot \varepsilon^2}{(1-\varepsilon) \cdot 2}} \leq e^{\frac{-t \cdot \varepsilon^2}{1 \cdot 2}} = e^{\frac{-36 \cdot \log n \cdot \varepsilon^2}{2\varepsilon^2}} = e^{-18 \cdot \log n} = n^{-18} = \frac{1}{poly(n)}$$

Problem 3

Part (a)

Claim

$\forall i \in [1...N-1] : P_{i+1} - P_i = \frac{q}{p}(P_i - P_{i-1})$

Proof

Let $i \in [1...N-1]$ be arbitrary.

To win from state i, the gambler performs one round. With probability p the gambler ends up in $i+1$, with probability q the gambler ends up in $i-1$. The probability of winning from $i-1$ is P_{i-1} and P_{i+1} from $i+1$, thus:

$P_i = p \cdot P_{i+1} + q \cdot P_{i-1}$

Since $p + q = p + (1-p) = 1$:

$p \cdot P_i + q \cdot P_i = p \cdot P_{i+1} + q \cdot P_{i-1}$

Rearranging the equation:

$p \cdot P_{i+1} - p \cdot P_i = q \cdot P_i - q \cdot P_{i-1}$

$p \cdot (P_{i+1} - P_i) = q \cdot (P_i - P_{i-1})$

$P_{i+1} - P_i = \frac{q}{p} \cdot (P_i - P_{i-1})$

Part (b)

Claim

$P_{i+1} - P_i = \left(\frac{q}{p}\right)^i \cdot P_1$

Proof

By induction over i.

Let $i \in [1...N-1]$ be arbitrary.

- Case $i = 1$

  $P_{i+1} - P_i = P_2 - P_1 = \frac{q}{p}(P_1 - P_0) = \frac{q}{p}(P_1 - 0) = \left(\frac{q}{p}\right)^1 \cdot P_1 = \left(\frac{q}{p}\right)^i \cdot P_1$

- Case $i > 1$

  $P_{i+1} - P_i = \frac{q}{p}(P_i - P_{i-1}) = \frac{q}{p} \cdot \left(\frac{q}{p}\right)^{i-1} \cdot P_1 = \left(\frac{q}{p}\right)^i \cdot P_1$

Claim

$P_{i+1} = \left(\sum_{j=0}^{i} \left(\frac{q}{p} \right)^j \right) \cdot P_1$

Proof

Note: Rewriting the formula from the previous claim yields: $P_{i+1} = P_i + \left(\frac{q}{p} \right)^i \cdot P_1$

By induction over i.

Let $i \in [1...N-1]$ be arbitrary.

- Case $i = 1$

  $P_{i+1} = P_1 + \frac{q}{p} \cdot P_1 = \left(\left(\frac{q}{p} \right)^0 + \left(\frac{q}{p} \right)^1 \right) \cdot P_1 = \left(\sum_{j=0}^{1} \left(\frac{q}{p} \right)^j \right) \cdot P_1 = \left(\sum_{j=0}^{i} \left(\frac{q}{p} \right)^j \right) \cdot P_1$

- Case $i > 1$

  $P_{i+1} = P_i + \left(\frac{q}{p} \right)^i \cdot P_1 = \left(\sum_{j=0}^{i-1} \left(\frac{q}{p} \right)^j + \left(\frac{q}{p} \right)^i \right) \cdot P_1 = \left(\sum_{j=0}^{i} \left(\frac{q}{p} \right)^j \right) \cdot P_1$

Part (c)

Claim

$p = q = \frac{1}{2} \Rightarrow P_i = \frac{i}{N}$

Proof

Let $i \in [0...N]$ be arbitrary.

- Case $i = 0$:

  $P_i = P_0 = 0 = \frac{0}{N} = \frac{i}{N}$

- Case $i = 1$:

  $P_N = \left(\sum_{j=0}^{N-1} \left(\frac{q}{p} \right)^j \right) \cdot P_1 = \left(\sum_{j=0}^{N-1} 1 \right) \cdot P_1 = N \cdot P_1$

  $\Rightarrow P_1 = \frac{P_N}{N} = \frac{1}{N} = \frac{i}{N}$

- Otherwise:

  $P_i = \left(\sum_{j=0}^{i-1} \left(\frac{q}{p} \right)^j \right) \cdot P_1 = \left(\sum_{j=0}^{i-1} 1 \right) \cdot P_1 = i \cdot P_1 = \frac{i}{N}$

<u>Claim</u>

$p \neq q \Rightarrow P_i = \frac{1-\left(\frac{q}{p}\right)^i}{1-\left(\frac{q}{p}\right)^N}$

<u>Proof</u>

The term $\sum\limits_{j=0}^{i-1}\left(\frac{q}{p}\right)^j$ is a geometric series. From calculus we know: $\sum\limits_{j=0}^{i-1}\left(\frac{q}{p}\right)^j = \frac{1-\left(\frac{q}{p}\right)^i}{1-\frac{q}{p}}$

Let $i \in [0...N]$ be arbitrary.

- Case $i = 0$: $P_i = P_0 = 0 = \frac{1-1}{N} = \frac{1-\left(\frac{q}{p}\right)^0}{1-\left(\frac{q}{p}\right)^N} = \frac{1-\left(\frac{q}{p}\right)^i}{1-\left(\frac{q}{p}\right)^N}$
- Case $i = 1$:

  $$P_N = \left(\sum_{j=0}^{N-1}\left(\frac{q}{p}\right)^j\right) \cdot P_1 = \left(\frac{1-\left(\frac{q}{p}\right)^N}{1-\frac{q}{p}}\right) \cdot P_1$$

  $$\Rightarrow P_1 = \frac{1-\frac{q}{p}}{1-\left(\frac{q}{p}\right)^N} = \frac{1-\left(\frac{q}{p}\right)^1}{1-\left(\frac{q}{p}\right)^N} = \frac{1-\left(\frac{q}{p}\right)^i}{1-\left(\frac{q}{p}\right)^N}$$
- Otherwise

  $$P_i = \left(\sum_{j=0}^{i-1}\left(\frac{q}{p}\right)^j\right) \cdot P_1 = \left(\frac{1-\left(\frac{q}{p}\right)^i}{1-\frac{q}{p}}\right) \cdot P_1 = \frac{1-\left(\frac{q}{p}\right)^i}{1-\left(\frac{q}{p}\right)^N}$$

Problem 4

Let $G(V, E)$ be the original graph. To analyse this problem, we construct a new graph $G'(V', E')$.

$V' = \{(u, v) \ \mid u, v \in V\}$

$E' = \{((u, v), (x, y)) \mid (u, x), (v, y) \in E\}$

The node (u, v) symbolises the lion being in state u and the deer being in state v in the original graph.

Let $n = |V|, m = |E|$, then $|V'| = n^2, |E'| = m^2$ follows directly from the definition of the new graph.

Since G is not bipartite, G has at least one odd cycle C. Let the nodes of C be denoted by $c_1, c_2, \dots c_k$ where k is the length of C. From the definition of G' follows that the nodes $(c_1, c_1), (c_2, c_2), \dots, (c_k, c_k)$ are connected to each other on a cycle in G' and thus form the odd cycle C'. Since C' exists, G' is not bipartite.

Claim

$\forall (u, v), (x, y) \in V'$, there exists a path from (u, v) to (x, y) with length $\mathcal{O}(n)$ in G'.

Proof

Let $(u, v), (x, y) \in V$ be arbitrary.

Since G is connected, there exists a path P_1 from u to x and P_2 from v to y. Let l_1 be the length of P_1 and l_2 be the length of P_2. Since G has only n nodes, $l_1, l_2 \in \mathcal{O}(n)$.

- Case 1: $l_1 \equiv l_2 \pmod 2$

  Assume without loss of generality $l_1 \leq l_2$.

  Let $u = p_1 \to p_2 \to \dots \to p_{l_1} = x$ be the representation of P_1 and $v = q_1 \to q_2 \to \dots \to q_{l_2} = y$ be the representation of P_2.

  Consider the following path: $(u, v) = (p_1, q_1) \to (p_2, q_2) \to \dots \to (p_{l_1}, q_{l_1}) = (x, q_{l_1})$. This path exists by definition of G'.

  To get to (x, y), we extend the path in the following way: The lion moves between $p_{l_1 - 1}$ and p_{l_1} until the deer arrives at at q_{l_2}. I.e. the first part of the tuple transitions between $p_{l_1 - 1}$ and p_{l_1}, the second part continues from $q_{l_1 + 1}$ to q_{l_2}. Since $l_1 \equiv l_2 \pmod 2$, the last node of the extension is $(p_{l_1}, q_{l_2}) = (x, y)$.

  Concatenating those paths results in a path from (u, v) to (x, y) with length $l_2 \in \mathcal{O}(n)$.

- Case 2: $l_1 \not\equiv l_2 \pmod 2$

  Let P_c be a path from u to c_1, i.e. a path to the odd cycle. This path exists since G is connected. The length of P_c is at most n. Let $\bar{P}_c$ denote the same path but backwards.

  Let $\bar{P}_1$ be a new walk on G:

$\bar{P}_1 = P_c + C + \bar{P}_c + P_1$

Let $\bar{l}_1$ be the length of $\bar{P}_1$.

Since the length of C is odd and the length of $P_c + \bar{P}_c =$ length of $2P_c$ is even, the following holds: $\bar{l}_1 \not\equiv l_1 \pmod 2$, hence: $\bar{l}_1 \equiv l_2 \pmod 2$

Thus, the problem was reduced to case 1 and therefore a path from (u, v) to (x, y) with length $\mathcal{O}(n)$ exists.

Since $\forall (u, v), (x, y) \in V'$, there exists a path from (u, v) to (x, y) with length $\mathcal{O}(n)$ in G', the diameter D of G' is in $\mathcal{O}(n)$ by definition.

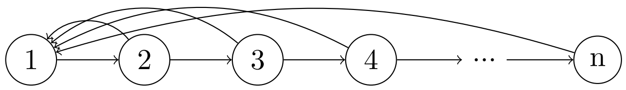
By invoking the corollary from the lecture, we know that the the maximum hitting time for any pair of vertices is bounded by $\mathcal{O}(|E'|D) = \mathcal{O}(m^2 n)$.

Let u, v be the the start vertices of the lion and the deer. The lion eats the dear if they arrive at a state (w, w) for an arbitrary $w \in V$.

Since the bound $\mathcal{O}(m^2 n)$ holds for any pair of vertices, it also holds for (u, v) and (w, w). Thus, the expected number of time steps for which the deer remains alive is bounded by $\mathcal{O}(m^2 n)$.

Problem 5

Consider the following directed graph for $n \geq 3$:



This graph has a the directed edges $(i, i+1)$ and $(i+1, 1)$ for $1 \leq i < n$.

Thus, the graph has the a transition matrix $P \in [0,1]^{n \times n}$ with the following entries:

$p_{1,2} = 1$, $p_{n,1} = 1$

For $2 \leq i < n$: $p_{i,i+1} = 0.5$ and $p_{i,1} = 0.5$

For all other elements: $p_{i,j} = 0$.

Since the graph is irreducible (strongly connected) and aperiodic (there exits a path of length $n+2$ and one of length $n+3$ for every node and $\gcd(n+2, n+3) = 1$). Therefore, a unique stationary solution π exists . By definition of a stationary solution, the following equation holds:

$\pi = \pi P$

Furthermore, by matrix multiplication we know for $j \in [1..n]$:

$\pi_j = \sum_{i=1}^{n} \pi_i \cdot p_{i,j}$

This yields the following linear equations for the given transition matrix P:

I: $\pi_1 = \pi_n + 0.5 \cdot \sum_{i=2}^{n-1} \pi_i$

II: $\pi_2 = \pi_1$

III: For $2 \leq i < n$: $\pi_{i+1} = 0.5 \cdot \pi_i \Leftrightarrow 2 \cdot \pi_{i+1} = \pi_i$

Claim: For $n \geq 3$: $2^{n-2} \cdot \pi_n = \pi_2$

Proof: By induction on n, using equation III

Let $n \geq 3$ be arbitrary.

Case $n = 3$: $2^{3-2} \cdot \pi_3 = 2 \cdot \pi_3 = \pi_2$

Case $n > 3$: $2^{n-2} \cdot \pi_n = 2^{n-3} \cdot (2 \cdot \pi_n) = 2^{n-3} \cdot (\pi_{n-1}) = 2^{(n-1)-2} \cdot \pi_{n-1} = \pi_2$

Therefore, for this graph and $u = 2, v = n$ the following holds:

$\frac{\pi_u}{\pi_v} = \frac{\pi_2}{\pi_n} = \frac{2^{n-2} \cdot \pi_n}{\pi_n} = 2^{n-2}$

Thus, $\pi_u / \pi_v = 2^{\Omega(n)}$.