Design and Analysis of Algorithms
Massachusetts Institute of Technology 6.046J/18.410J
Debayan Gupta, Aleksander Madry, and Bruce Tidor Handout 7

Recitation 7: Randomized Algorithms

1 Recap: Matrix Product Verification

For this problem, we are given $n \times n$ matrices A, B, C and we are tasked with verifying that $A \times B = C$. The naive algorithm simply multiplies $A \times B$ in $O(n^3)$ time. Instead, we can use a randomized approach, wherein we pick a random bit vector $r[1...n]$ and simply check if $A(Br) = Cr$. If it is, we claim that $AB = C$; else we say they are not equal. Since we can multiply a matrix by a vector in $O(n^2)$ time, we can perform the verification in $O(n^2)$.

This is an example of a *Monte Carlo* randomized algorithm since it runs fast, specifically in polynomial time, and produces probably correct output. Specifically, when $AB = C$, it always provides the right answer; however, when $AB \neq C$, we claim that we provide the correct answer with probability at least $\frac{1}{2}$.

The proof of this result comes from considering the matrix $D = AB - C$. If $AB \neq C$, then D must have a nonzero entry, say $d_{i,j}$. Consider the bit vector v which has zeros in all its coordinates except with v_j = 1. Notice then that $(Dv)_i = d_{i,j} \neq 0$.

Now, consider a vector r that forces us to provide the incorrect answer, namely $Dr = 0$ even though $AB \neq C$. The vector $r' = r + v$ where $r'_i = (r_i + v_i) \mod 2$ is a distinct bit vector from r. We also have that $Dr' = D(r + v) = Dr + Dv = 0 + Dv \neq 0$. Therefore, for every bit vector we produce that leads to an incorrect answer, there is another equally likely bit vector with the correct answer, as $\mod 2$ is an invertible operation. This shows that with probability at least $\frac{1}{2}$ that the randomized approach gives the correct answer even when $AB \neq C$.

2 Verifying Polynomial multiplication

Consider the problem of verifying the multiplication of two polynomials. Given two polynomials $a(x), b(x)$ of degree n we saw in the class how to compute $a(x) \cdot b(x)$ using FFT in time $O(n \log n)$. In the verification problem one is also given a polynomial $c(x)$ of degree at most $2n$ and we want to check whether $c(x) \equiv a(x) \cdot b(x)$. We will show that there is a randomized algorithm that always runs in time $\Theta(n)$ and has error probability $1/50$. No known deterministic algorithms can match this run time.

Algorithm 1 Verify polynomial multiplication algorithm:
VERIFY-MULT(a, b, c)

1: Choose x_0 uniformly at random from $S = \{0, 1, \ldots, 100n - 1\}$
2: **return** $a(x_0) \cdot b(x_0) == c(x_0)$ (boolean value)

Running time: The algorithm runs in time $O(n)$ since it just needs to evaluate three degree $O(n)$ polynomials at one point.
Correctness:

- If indeed $c(x) \equiv a(x) \cdot b(x)$, then the algorithm will output TRUE with probability 1.
- If $c(x) \not\equiv a(x) \cdot b(x)$, the algorithm will erroneously output TRUE if and only if x_0 is a root of the polynomial $p(x) = c(x) - a(x) \cdot b(x)$. Since $p(x)$ has degree at most $2n$, it has at most $2n$ distinct roots. So, the probability that x_0 is one of those roots is:

$$\Pr[p(x_0) = 0] \leq \frac{\sharp roots}{|S|} = \frac{2n}{100n} = 1/50$$

  where the equality holds iff *all* roots of $p(x)$ are in S.

So, the algorithm can give false positives with probability at most $1/50$ and no false negatives.

3 Probability Theory

Linearity of Expectation

Given that $X_1, ..., X_n$ are arbitrary discrete random variables and $X = \sum_{i=1}^{n} X_i$, we have

$$\mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \mathbb{E}[X_i]$$

Essentially, it states that the expected value of the sum of random variables is equivalent to the sum of the expected values. The most important takeaway from the result is that it holds even if the random variables are not independent. This will be used frequently when we have to find the expected value of a sum of random variables when we are not sure if they are independent.

Union Bound

Let $A_1, ..A_n$ be a set of events. Then we have

$$\Pr[\bigcup_{i=1}^{n} A_i] \leq \sum_{i=1}^{n} \Pr(A_i)$$

The inequality is an equality when all A_i are disjoint. This technique is commonly used when we want to provide a bound on whether a property holds for one of many random variables. In this case, we can upperbound it by the sum of the probabilities that the property holds for all of the random variables.

Markov's Inequality

Let Y be an arbitrary discrete random variable that takes non-negative values in the subset Ω of $\mathbb{N}$. Let also $a \geq 0$ then

$$\Pr[Y \geq a] \leq \frac{\mathbb{E}[Y]}{a}$$

The very nice feature of the inequality is that it requires only the computation of the expectation.

Proof.

$$\mathbb{E}[Y] = \sum_{y \in \Omega} y \Pr[Y = y] = \sum_{y \in \Omega, y \leq a} y \Pr[Y = y] + \sum_{y \in \Omega, y > a} y \Pr[Y = y]$$

$$\geq \sum_{y \in \Omega, y > a} y \Pr[Y = y] \geq \sum_{y \in \Omega, y > a} a \Pr[Y = y] = a \Pr[Y > a] \qquad \square$$

The inequality is an equality in the case where Y is always equal to a. Markov's inequality is important because it ties the probability of a random variable being greater than some threshold to the expected value of the random variables. Therefore, it's commonly used in tail inequalities, which bound the probability of a random variable being some distance from its mean. It can also be extended to prove much more complex results.

4 Random Walks Recap

A *random walk* on a graph $G = (V, E)$ of length t is defined to be a random process in which we start at some node s and repeat the process t times of choosing randomly among the neighbors v' of the current vertex v we are at and moving to it. If the graph graph is weighted, we transition to a neighbor v' from v with probability proportional to the weight of the edge.

In this class, we focus on random walks as distributions, namely the the probability distribution across the vertices of the walk for a fixed starting point s and known time t. We can mathematically represent the distribution using the expression p_v^t, which is the probability that the walk visits vertex v after t steps. We then have the following recurrence.

$$p_v^0 = \begin{cases} 1 & \text{if } v = s \\ 0 & \text{otherwise} \end{cases}$$

$$p_v^{t+1} = \sum_{e \in E, e=(u,v)} \frac{1}{d(u)} p_u^t$$

where $d(u)$ is the degree of u in the G. With these definitions, we can formulate random walks as a series of matrix products.

Consider the adjacency matrix A for which $A_{u,v} = 1$ if $(v, u) \in E$ and 0 and the degree matrix D, which is diagonal and has $D_{u,u} = d(u)$. We can define the *walk* matrix as $W = AD^{-1}$ or

$$W_{u,v} = \begin{cases} \frac{1}{d(v)} & \text{if } (v, u) \in E \\ 0 & \text{otherwise} \end{cases}$$

Notice then that $p^{t+1} = Wp^t = W^t p^0$.

We also define a *stationary distribution* as a distribution π such that $W\pi = \pi$. This distribution π represents a steady state, one that does not change after a random walk time step. Some graphs converge to a stationary distribution regardless of the starting state. A directed graph converges to a *unique* stationary distribution if it

- is strongly-connected
- is aperiodic

A graph is said to be strongly connected if every vertex is reachable from every other vertex. Also, a graph is said to be aperiodic if it is not the case that every cycle has length that is a multiple of some number greater than 1. Furthermore, if we can show that the graph has a self-loop (cycles of length 1), then we can show that it is is aperiodic.

5 Stationary Distributions of Top-In Shuffling

Our goal of card shuffling is to compute a uniformly random permutation of a deck of cards. We will identify a deck with the ordering of cards within it. Here is one algorithm for shuffling a deck of k cards using n operations:

SHUFFLE-DECK$(deck, n)$

for $i = 1$ to n
  TOP-IN$(deck)$
return $deck$

TOP-IN$(deck)$

Take the top card of $deck$ and insert it at one of the k positions in the deck uniformly at random.

We'd like to show that SHUFFLE-DECK is a good way to shuffle a deck of cards, i.e. that as $n \to \infty$ the probability distribution associated with our deck is uniform over all possible permutations of the cards.

We can do this by considering this problem as a random walk along a graph. Consider the weighted directed graph $G = (V, E, p : E \to \mathcal{R})$ with

- $V =$ the set of all possible decks (so $|V| = k!$)
- $p(u, v) = \Pr[\text{TOP-IN}(u) = v]$
- $E = \{(u, v) \mid p(u, v) > 0\} = \{(u, v) \mid v \text{ is a possible output of TOP-IN}(u)\}$ [1]

Now, it is evident that SHUFFLE-DECK is a random walk on G of length n. Thus if we can show that the directed graph G is strongly-connected and aperiodic, we can show that it converges to a unique stationary distribution.

A graph is said to be strongly connected if every vertex is reachable from every other vertex. Also, a graph is said to be aperiodic if it is not the case that every cycle has length that is a multiple of some number greater than 1. Furthermore, if we can show that G has a self-loop (cycles of length 1), then we can show that G is aperiodic.

The first item might seem obvious, but we should still check. Give the cards in the deck some arbitrary numbering and consider any two orderings $A = [a_1, a_2, ..., a_k]$ and $B = [b_1, b_2, ..., b_k]$ of the cards deck, where the i^{th} card in ordering A is the card with number a_i. We could go from A to B using applications of TOP-IN as follows:

A-TO-B(A, B)

for $i = 1$ to k

Take cards off the top and put them at the i^{th} to last position in the deck...
until we uncover the card with number b_{k-i+1}.
Put card b_{k-i+1} in its proper position.

We are choosing where to insert the top card because TOP-IN could insert the card anywhere in the deck.

The second item is in fact obvious, because for any $u \in V$, TOP-IN(u) could return the same deck u if the top card is re-inserted on the top of the deck, and so this is a self-loop at u by definition, making G aperiodic.

Thus G has a stationary distribution. Now all that remains to be shown is that this distribution is uniform. Because the theorem tells us that the stationary distribution is unique, it will suffice to show that the uniform distribution is stationary.

[1] Note that the existence of an edge (u, v) does not imply that $(v, u) \in E$ because TOP-IN could (for example) put the top card somewhere in the middle of the deck from whence it could not possibly be extracted with a single application of TOP-IN.

Suppose our probability distribution over the nodes of G at time t is the uniform distribution

$$x_t(u) = \Pr[\text{we are at vertex } u \text{ at time } t] = \frac{1}{k!} \quad \text{for all } u \in V.$$

Now consider any arbitrary node $u \in V$. Let S be the set of vertices v for which there is an edge in E from v to u. By definition of the random walk the probability of being on node u at time $t+1$ is

$$\sum_{v \in S} x_t(v) p(v, u).$$

We can consider the operation inverse to TOP-IN as taking a card from anywhere in the deck and placing it on top. Evidently there are k possible outcomes. So $|S| = k$. Furthermore, for all $v \in S$, $p(v, u) = \frac{1}{k}$ because TOP-IN chooses a location uniformly at random. Thus

$$\begin{aligned} x_{t+1}(u) &= \sum_{v \in S} x_t(v) p(v, u) \\ &= \sum_{v \in S} \frac{1}{k!} \frac{1}{k} \\ &= \frac{1}{k!} \end{aligned}$$

and we conclude that the uniform distribution is the unique stationary distribution of G.