scribe : Sarah PEREZ
sarah.perez@mail.mcgill.ca

Professor Joel FRIEDMAN
Professor Will EVANS

An introduction to random variables in algorithms : Finding the expected running time of the Proposal Algorithm.

CPSC 500 - ALGORITHMS AND COMPLEXITY
SCRIBE ASSIGNMENT
September $25^{th}2013$

Overview

The Gale-Shapley algorithm [1], also known as the Proposal Algorithm, provides a stable solution to the Stable Marriage matching problem. The analysis of this matching algorithm's running time gives insight into the roles of random inputs on the variability of algorithms. The expected average running time of the Proposal Algorithm is determined using the analysis of two games with a randomized input : Clock Solitaire and the Coupon's Collector game.

1 The Clock Solitaire Game

In the Clock Solitaire game, the player is invited to shuffle and layout a normal deck of 52 cards faced down as in Figure 1.

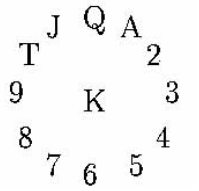

FIGURE 1 – Each pile of 4 is labeled by the rank of a card given its position around the clock.

Game rules and procedure :

1. Draw a card from the King pile. Call this card X.
2. Draw a card from the pile labeled by rank(card X).
3. Repeat step 2 by drawing from the pile labeled by the rank(new card). If the pile is empty, break.
4. If all 52 cards have been turned over, the game is won !

What is the probability of winning ? The outcome of the game depends on a random input : the order in which the cards are placed on the clock determines if the player wins or loses. To calculate the probability of winning, we begin by analyzing the game's procedure and observe the following [1] :

Fact A : The game ends with an attempt to draw of the K pile.

Indeed, the King pile is the only pile which can become empty and from which we may still attempt to draw a card. Our next observation is that the distribution of the first card we pick up is uniform : the probability of picking up any card is equally likely on the first turn. Furthermore, the probability of picking a particular card on the second turn is uniform across all cards except the first card drawn, which cannot be drawn again. This fact leads us to our second observation :

Fact B : At each draw, any unseen card is equally likely.

Keeping these two facts in mind, let us consider the following sequence :

3 7 K J 10 Q ... ... A

1. One way of approximating the probability of winning is to count all the possible arrangements that lead to a win and divide that number by the total number of possible arrangements. Since we would have to consider each of 52! different arrangements (ways to deal the cards around the clock), this is quite difficult.

We first notice that this sequence cannot be a sequence of cards revealed during an execution of the clock solitaire game because, by *Fact A*, the last card in such a sequence must be a King. Only sequences that contain all four Kings and end with a King are valid sequences of the game. To be a winning valid sequence, the sequence must contain all 52 cards. What is the probability that such a sequence is produced ? We can imagine writing down all 52 ! orderings of cards to calculate this probability. Exactly that part of the ordering up to when the fourth King occurs is the sequence that is revealed by the algorithm. What is the probability that the arrangement (or ordering) results in a win ? It is the probability that the ordering ends with a King. Since each ordering is equally likely by *Fact B*, this probability is $p = \frac{1}{13}$.

2 The Proposal Algorithm's average running time

2.1 Repeating a Clock Solitaire like analysis

In the Stable Marriage problem, we are trying to match men and women given their ordered match preferences. The Proposal Algorithm fortunately solves the matching problem and guarantees a stable solution. Last class, we saw that the running time of the algorithm ranges from [n to n^2] depending on the order in which we ask the men to make proposals. The running time of this algorithm depends on a random input just as the Clock Solitaire game's outcome depends on the shuffled cards' order. To determine the average running time of the Proposal Algorithm, we mimic our analysis of the Clock Solitaire game :

When does the algorithm terminate ?
Fact A : Proposals stop once every woman has been proposed too.

What does the order of the proposals depend on ?
Fact B : An unpaired man chooses the next women to propose to
by choosing at random from those he has not yet proposed to.

Sadly, *Fact B* is difficult to investigate ! The series of proposals follows a complicated distribution given that a man will not propose to a woman who has rejected him. In order to simplify our investigation we consider a variant of the original Proposal Algorithm.

2.2 The Amnesiac Proposal Algorithm

In the amnesiac variant, men are forgetful and may propose again to a woman who has rejected them. An unpaired man then chooses the next women to propose to uniformly from all women. This modification simplifies the calculation of the number of total proposals and thus the calculation of running time of the algorithm. But does the variant also guarantee a stable solution ? Indeed it does since any repeated proposal will still be rejected.

We also note that this variant will have a number of proposals that stochastically dominates the number of proposals made by the original algorithm.

Reminder :
Where X and Y are two random variables we wish to compare,
if X stochastically dominates Y then : $P(X > x) \geq P(Y > x)$

The amnesiac version will therefore give us an upper bound on the average running time of the original algorithm which is given by the number of iterations it takes to see each women be proposed to, according to *Fact A*. The number of iterations needed to see each woman being proposed to is equivalent to measuring the number of trials needed to sample from a uniform finite distribution, with replacement. The later was first formulated as the Coupon Collector's Problem, which we see next.

3 The Coupon Collector's Problem

The Coupon Collector is playing the game of collecting all possible n coupons found in cereal boxes. Each box is equally likely to contain any of the n coupons needed for a complete collection of n coupons. This problem has several applications including that of testing the "randomness" in the decimal expansion of π and e [3]. We formulate the problem mathematically as such :

Let Y be the number of trials needed to collect at least one of each type of coupon.
Let X_i be the number of trials performed when we have exactly i coupons, where :
$Y = X_o + X_1 + \cdots + X_{n-1}$ and $P[X_i = j] = (1-p)^{(j-1)} * p$

X_i has a geometric distribution with probability $p = \frac{n-i}{n}$. What we are interested in however is the expected value of Y or the expected value of the sum of all X_i :

$$E[X_i] = \sum_j P[X_i = j] * j \quad (1)$$
$$= \sum_j (1-p)^{(j-1)} * pj \quad (2)$$
$$= \frac{1}{p} = \frac{n}{n-i} \quad (3)$$
$$(4)$$

where in the last derivation in Equation 3 was calculated using generating functions, something we won't cover here. Finally we obtain a Harmonic series that of course doesn't converge ! However we can provide an estimate to the series :

$$E[Y] = \sum_i E[X_i] \quad (5)$$
$$= \sum_{i=0}^{n-1} \frac{n}{n-i} \quad (6)$$
$$= n \sum_{i=1}^{n} \frac{1}{i} \quad (7)$$
$$\approx n \ln(n) + O(n) \quad (8)$$

We can conclude with certainty that the average number of trials needed to collect all coupons is $\log n$ which is also an upper bound on the average running time of the Proposal Algorithm.

4 Random Variable measurements : Markov's Inequality and variances

We continue our investigation of random variables and try to further characterize Y. One simple measurement is that of the mean deviation of Y to its expected value. This measurement is called the variance. The expected variance is defined as follows : $Var(Y) = E[(Y - E[Y])^2]$.

Whereas the variance tells us about of the random variable Y, we would also like to know how likely it is that Y is a certain distance away from its expected value. We use Markov's Inequality which gives a rough estimate of this distance :

> For any Z, a non-negative random integer, the following holds :
> $P[Z \geq a] \leq \frac{1}{a} * E[Z]$ where $a > 0$
> **Proof :** $P[Z \geq a] = \sum_{j \geq a} P[Z = j] \leq \sum_{j=0}^{\infty} \frac{1}{a} * j * P[Z = j]$

Markov's inequality therefore quantifies an upper bound on the likelihood that a random variable is far from its expected value by a factor of a. This inequality is used to prove Chebyshev's inequality, which provides an even better upper bound on the deviation from the mean given the random variable's variance, as we shall see next class.

5 Conclusion

Expected values and variances are only the tip of the iceberg in understanding the distribution of random variables such as those describing the outcomes of the Clock Solitaire game, the Amnesiac Proposal Algorithm, and the Coupon Collector's problem. We have analyzed all three scenarios and drawn insight from them to finally find an upper bound on the running time of the Proposal Algorithm, that of $n \log(n)$. Such derivations are also important for studying the variants of the Proposal Algorithm such as the Roommate problem and the Hospitals/Residents problem with couples [2]. Further investigations of these random variables will also touch on the notion of sampling, an important topic in the field of randomized algorithms.

References

[1] D. Gale and L. S. Shapley : College Admissions and the Stability of Marriage, *American Mathematical Monthly* 69, 9-14, 1962.

[2] D. Gusfield and R. W. Irving : The Stable Marriage Problem : Structure and Algorithms, MIT Press,p. 54, 1989.

[3] Brian Dawkins : Sibhan's Problem : The Coupon Collector Revisited, *American Statistical Association* 45, No.1, 1991.