Module 10
Randomized Algorithms

S. Lakshmivarahan
School of Computer Science
University of Oklahoma
USA-73019
Varahan@ou.edu

Randomized Algorithms

- A deterministic algorithm gives exact results for all instances of a problem.
- The complexity is that of the worst case analysis
- Randomized algorithms use one or more random moves/choices in an otherwise deterministic method
- These algorithms may behave differently when run on the same input

Randomized Algorithms

- Random moves/choices are made by consulting a random number generator
- Algorithm works well on typical instances
- Under a suitable assumption on the input distribution, we can compute the average case analysis

Randomized Algorithms

- The question is: what constitutes a “good” distribution?
- The performance measure are:
  - Expected or average running time
  - Probability that an algorithm will terminate in a certain number of steps

Classification of Randomized Algorithms

1. Monte Carlo Method:

- Always provide an approximate solution
- The quality of the approximation increases with the number of samples used
- Originally developed to numerically evaluate multiple integrals with a complex integrand
- Used since 1940s

Classification of Randomized Algorithms

2. Las Vegas Method:

- Never returns a wrong answer, but it may take forever to get the right answer
- Introduced by Babai in 1979

Classification of Randomized Algorithms

- We illustrate both the classes by one example for each
- Before that, we need to understand the basic algorithms used to generate uniformly distributed integers in the set {1, 2, … , n} for some large N.

Random Number Generation

- Let $m > 0$ be a large integer
- Let $a > 0$ be an integer that is relatively prime to m. That is, GCD(a,m) = 1
- Let $c > 0$ be a large integer
- Let $x_0 > 0$ be a large integer

Random Number Generation

- Then, the recurrence

$$x_{n+1} = (ax_n + c) \bmod m \rightarrow (1)$$

that generates a sequence x_0, x_1, x_2, … is said to behave like a random sequence of integers in the range [1, m-1]

- Since the recurrence (1) is a deterministic device, its output is often called a pseudo-random sequence.

Random Number Generation

- In a standard 32-bit architecture, since the integer arithmetic is modulo arithmetic with $m=2^{31}-1$, using this value in ①, we will get the maximum possible 2,147,483,647 number of numbers in the range $[1,2^{31}-1]$ in a random order.

Random Numbers in the range [0,1)

- Define

$$y_n = \frac{x_n}{m} \qquad \rightarrow (2)$$

where x_n is generated by (1)

- It can be easily verified that

$$0 \leq y_n < 1 \qquad \rightarrow (3)$$

- The claim is $\{y_n\}$ is uniformly distributed in [0,1)

Homework

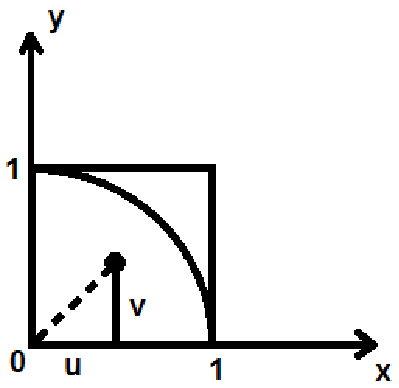
1. Generate 10^6 random number y_n. Compute the histogram of the sequences
2. Compute the mean and variance of the 10^6 you have generated and compare it to those of the standard uniform distribution in [0,1]

Random Number Generators

- Note: For a fixed m, a, and c, by using a different initial seed x_0, we can generate a different set of random numbers
- If we use the same x_0, we will get the same sequence
- By suitable transformation of y_n, we can generate random numbers from quite different families of distributions such as binomial, exponential, Poisson and Gaussian distributions.

Monte Carlo Method

- Consider a quadrant of a circle of radius inscribed in a unit square as shown



- Our goal is to compute the area of this part of the circle

Monte Carlo Method

- The idea goes as follows: generate a pair (u, v) of uniformly distributed random numbers in the range [0,1)
- Then, this pair (u,v) defines a point randomly chosen within the unit square
- If $(u^2 + v^2) \leq 1$ then this point lies inside the circle, otherwise it lies in the annulus region outside the circle but within the square

Monte Carlo Method

- Thus, by picking a set of N pairs (u_i, v_i) for $1 \le i \le N$, we can determine the number M of pairs that fall within the circle
- Then,

$$\lim_{n\to\infty} \frac{M}{N} \quad \to \quad \frac{\pi}{4}$$

which is the area of one quadrant of a unit circle

- This can also be thought of as a Monte Carlo algorithm to estimate the value of π

Monte Carlo Method

- A program:

  $Count = 0$

  for $i = 1$ to N

  Generate (u, v)

  if $((u^2 + v^2) \leq 1)$ then $count = count + 1$

  End for

  Print the ratio $\frac{Count}{N}$

Monte Carlo Method

- Homework:

3. Plot the ratio $\frac{count}{N}$ by changing N through 10^3, 10^4, 10^5, 10^6

- Note: The value of π is a non-terminating fraction and expansion of π for several thousands of digits are published. Check how many digits you are able to compute

Las Vegas Algorithm

- Suppose that there is a small township with a population of n persons (eligible voters)
- An election for the mayor of the town is coming in a month
- The problem is that every one of the eligible voters wants to run for the Mayoral race
- Since no one will vote for anyone else, we cannot conduct the election in the way we are used to

Las Vegas Algorithm

- Randomized algorithm of the Las Vegas type comes to our rescue
- The town hires an external consultant to conduct the election so that sooner or later we will have a new Mayor taking the oath of office

Las Vegas Algorithm

- The consultant brings with him n copies of a computing machine that can generate random integers in the range [1,N] for N>n
- On the election day, they all assemble in a local stadium where a copy of the computing machine is distributed to every one as they come in and take up their assigned seats

Las Vegas Algorithm

- The election is conducted in cycles
- <u>**Step 1**</u>: Everyone generates a uniformly generated random integer in the range [1, n]
- If anyone gets the number 1, they stay in the election and all others cannot continue to the next round
- <u>Note</u>: Since the generator is fair and generates numbers uniformly, everyone has equal chance of $\frac{1}{n}$ to get the number 1. This guarantees there is no bias in the process

Las Vegas Algorithm

- **Step 2**: The consultant counts those who all got the number 1, and lets n_1 be the number of those in this group. Clearly, $1 \le n_1 \le n$
- Note: If $n_1 = 1$, we have a winner. If $n_1 > 1$, then we go to the next round.

Las Vegas Algorithm

- **<u>Step 3</u>**: The consultant asks this subgroup to generate a uniformly distributed random number in the range $[1, n_1]$. Clearly, with probability $\frac{1}{n_1}$ anyone can obtain 1.
- All those who got 1 remain in the race and all the others step aside
- Let n_2 where $1 \leq n_2 \leq n_1$ be the number of folks still staying in the race.

Las Vegas Algorithm

- **<u>Step 4</u>**: This protocol is continued until we get a stage where only one person gets the number 1, at which stage the election ends successfully
- It is clear that it may take a large number of rounds to get a leader, but eventually we will find one
- The question is, what is the average number of rounds needed to elect a leader by this protocol?

Las Vegas Analysis

- Let L(n) be the average number of rounds needed to elect a leader
- Then $L(n) = \begin{Bmatrix} 1 \; with \; probability \; p(n,1) \\ 1 + L(n) \; with \; probability \; p(n,0) \\ 1 + L(j) \; with \; probability \; p(n,j) \end{Bmatrix} \rightarrow ⑤$

where

$$p(n,j) = \binom{n}{j}\left(\frac{1}{n}\right)^{j}(1-n)^{n-j} \quad \rightarrow ⑥$$

$=$ probability that j out of n folks will generate the number 1

Las Vegas Analysis

- Thus,

$$L(n) = 1 * p(n,1) + [1 + L(n)]p(n,0) + \sum_{j=2}^{n} L(j)p(n,j) \rightarrow 7$$

$$= \sum_{j=0}^{n} p(n,j) + L(n)p(n,0) + \sum_{j=2}^{n} L(j)p(n,j)$$

$$= 1 + L(n)p(n,0) + \sum_{j=2}^{n} L(j)p(n,j)$$

and

$$L(n) = \frac{\left[1 + \sum_{j=2}^{n-1} L(j)p(n,j)\right]}{[1 - p(n,0) - p(n,n)]} \rightarrow 8$$

Las Vegas Analysis

- Notice that L(n) depends on L(j) for $2 \leq j \leq n-1$ and it is a complete history recurrence
- It can be verified

$$\left.\begin{aligned} &a)\ L(n) < e = 2.718 \quad \text{for all } n \geq 2 \\ &b)\ \lim_{n\to\infty} L(n) < 2.442 \end{aligned}\right\} ⑨$$

- That is, on average one only needs three rounds, even for very large n

Homework

4. Plot $p(n, j)$ for $0 \le j \le n$ and $n=10, 100$
5. Compute and plot $L(n)$ vs. n for $2 \le n \le 50$

Reference

- Itai and Rodeh (1981). “Symmetry breaking in distributed networks”, Proceedings of the 22nd Annual IEEE Symposium of Foundations of Computer Science, page 150-158