Introduction to Randomized Algorithms

Dr. Priyadarshan Dhabe,

Ph.D (IIT Bombay)

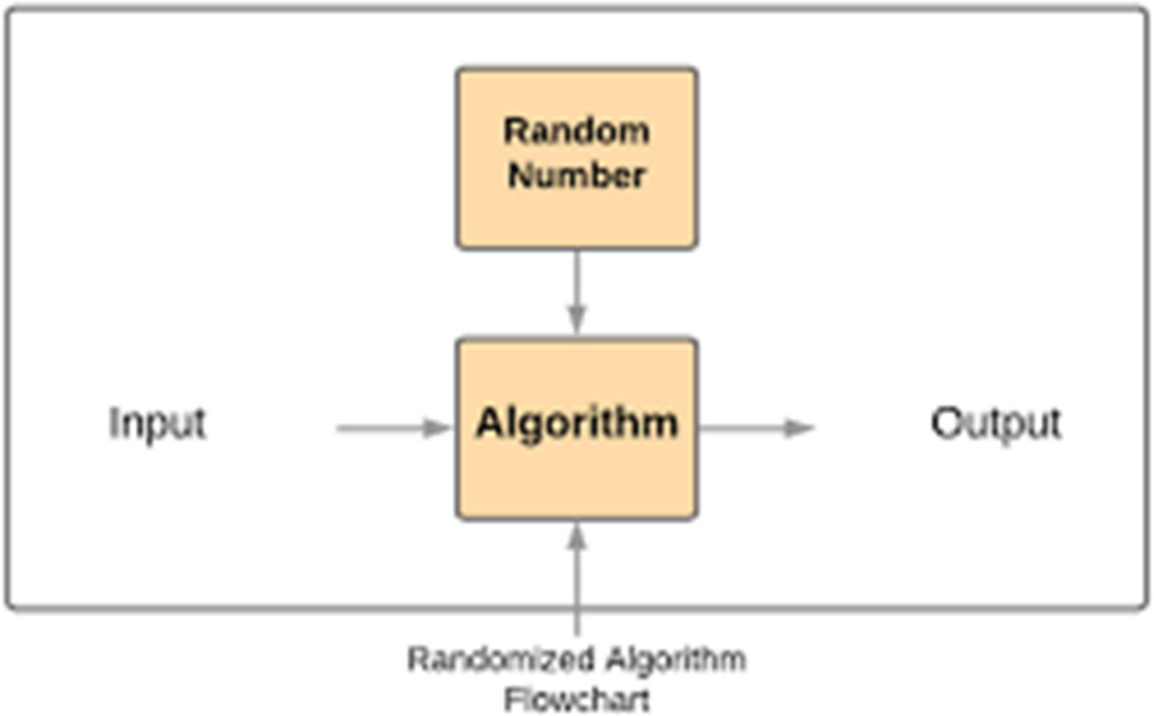
What is Randomness?

- A phenomenon or procedure for generating data is **random** if
  - the outcome is not predictable in advance;
- The probability of any outcome of a random phenomenon must be between 0 and 1.
- The sum of probabilities of all the possible outcomes associated with a particular random phenomenon must add up to exactly 1.

What are randomized Algorithms?

- An algorithm that employs degree of **randomness** in part of its logic.
- They uses uniformly random values to guide its behavior and hoping for **good performance** in the average case using them.
- An algorithm that uses random number to decide what to do next?, anywhere in its logic, is called **randomized algorithm** e.g. randomized quick sort.
- In randomized quick sort pivot can be selected using random number between 1 and n, hoping for better partitioning (balanced one)

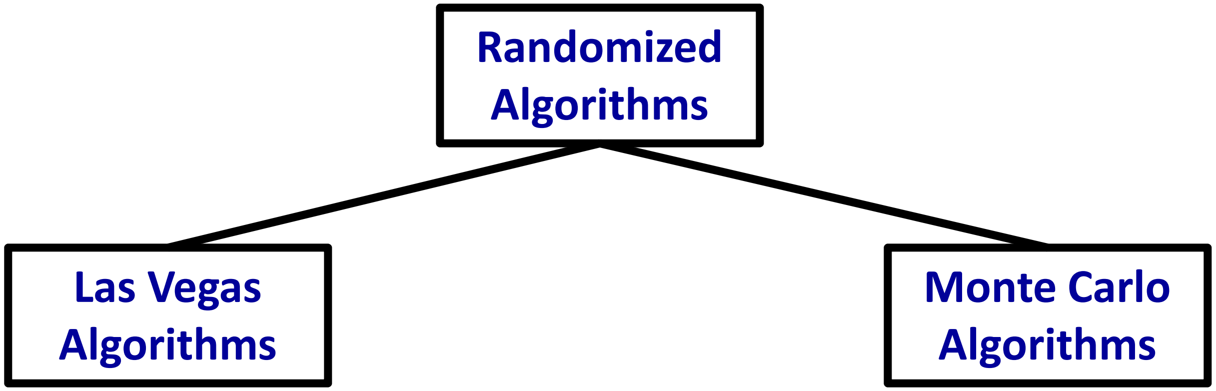
What are randomized Algorithms?



Output is decided by not only input but also using random number, also called as probabilistic algorithms

Deterministic algorithms don't use random numbers

Types of randomized Algorithms



Las Vegas is an internationally renowned major resort city, known primarily for its GAMBLING, shopping, fine dining and entertainment, in Nevada state of USA- wiki
Laszlo Babai ,1979, is associated with Las Vegas and name the algorithm due to randomness in gambling specifically, independent coin flipping experiment.

Monte Carlo is a famous casino in the costal area of France called Monaco. It is a GAMBLING and entertainment complex-wiki
Invented by in 1947 by Nicholas Metropolis and named it due to randomness in gambling in casino.

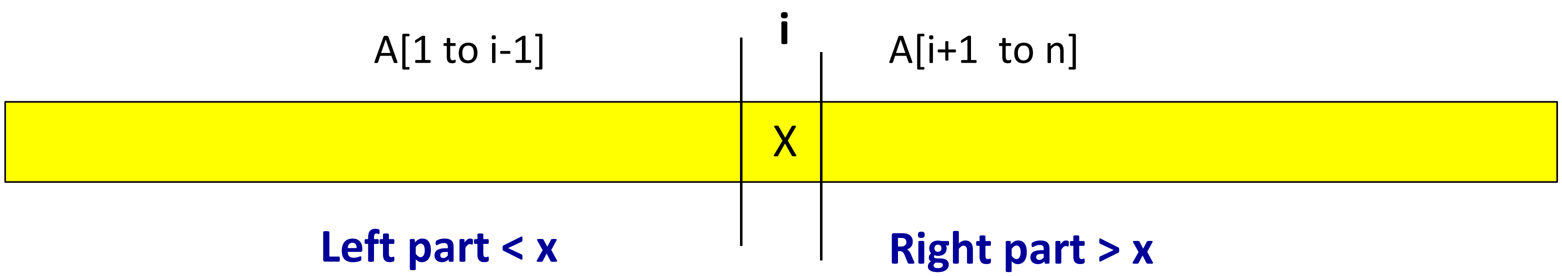
Types of randomized Algorithms

Las Vegas algorithm	Monte Carlo Algorithm
It gives correct results or informs about the failure to find solution.	Result may be incorrect with small probability
Expected runtime may differ depending on input but it is always finite.	Running time is certain or fixed

	Running Time	Correctness
Las Vegas Algorithm	probabilistic	certain
Monte Carlo Algorithm	certain	probabilistic

```
// Las Vegas algorithm
repeat:
   k = RandInt(n)
   if A[k] == 1,
      return k;
```

```
// Monte Carlo  algorithm
Repeat 300 times:
   k = RandInt(n)
   If A[k] == 1,
      return k;
Return "failed"
```



Randomized QuickSort

Bound to produce correct result and execution time varies per run but finite

```
INPUT:
    # A is an array of n elements
def randomized_quicksort(A):
    if n == 1:
        return A  # A is sorted.
    else:
        i = random.randrange(1, n)  # Will take a random number in the range 1~n
        X = A[i]  # The pivot element
    """Partition A into elements < x, x, and >x  # as shown in the figure above.
    Execute Quicksort on A[1 to i-1] and A[i+1 to n].
    Combine the responses in order to obtain a sorted array."""
```

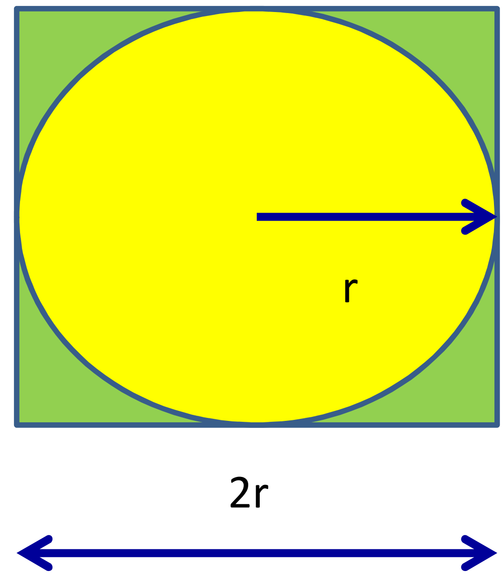
Randomized quick sort

- The runtime of QuickSort depends heavily on how well the **pivot** is selected?
- If a value of pivot is either **too big** or **small**, the partition will be **unbalanced**, resulting in a poor runtime efficiency i.e **O(n^2)**.
- However, if the value of pivot is **near the middle** of the array, then the split will be reasonably **well balanced**, yielding a faster runtime.
- Since the pivot is randomly picked, the running time will be **good most of the time** and bad occasionally, thus behavior can be **O(n log n).**

Estimating value of Pi using Monte Carlo simulation

- Assume we have a square of size 2r.

$$A_{square} = (2r * 2r) = 4r^2$$



- Draw a circle inside the square with radius r

$$A_{circle} = \pi * r^2$$

- The ratio of area of circle to the area of square is then

$$\frac{A_{circle}}{A_{square}} = \frac{\pi * r^2}{4 * r^2} = \frac{\pi}{4}$$

Estimating value of Pi using Monte Carlo simulation

$$\frac{A_{circle}}{A_{square}} = \frac{\pi}{4}$$

Multiplying both sides by 4 yields

$$4 \times \frac{A_{circle}}{A_{square}} = 4 \times \frac{\pi}{4}$$

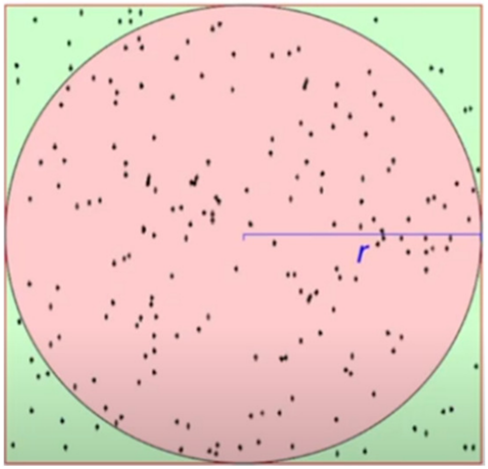
$$4 \times \frac{A_{circle}}{A_{square}} = \pi$$

$$\therefore \pi = 4 \times \frac{A_{circle}}{A_{square}}$$

Estimating value of Pi using Monte Carlo simulation

- Draw many points randomly selected within the square, then, the ratio can be approximated by

$$\frac{A_{circle}}{A_{square}} \approx \frac{\text{Number of points within the circle}}{\text{Number of points within the square}}$$



$$\pi = 4 \times \frac{A_{circle}}{A_{square}}$$

$$\pi \approx 4 * \left(\frac{\text{Number of points within the circle}}{\text{Number of points within the square}} \right)$$

- As we draw **more and more** points randomly selected within the square, the approximation will be **better and better**

Estimating value of Pi using Monte Carlo simulation

- Following are some of the estimated values of pi for the randomly selected points in the square.

$$\pi = 4 * \frac{7}{10} = 2.80$$

$$\pi = 4 * \frac{24}{31} = 3.0967741935$$

$$\pi = 4 * \frac{48}{59} = 3.2542372881$$

$$\pi = 4 * \frac{112}{138} = 3.2463768116$$

$$\pi = 4 * \frac{490}{616} = 3.1818181818$$

More the random points better is the approximation

https://www.youtube.com/watch?v=qcWg59jvqeA