Learn DAA : From B K Sharma

TCS-503: Design and Analysis of Algorithms

Unit V: Selected Topics: Randomized Algorithms

Unit V

- Selected Topics:
  - NP Completeness
  - Approximation Algorithms
  - Randomized Algorithms
  - String Matching

Introduction

Till now, we know that the running time of algorithm is fixed for a given input, i.e., depends on the size of input n, T(n).

Deterministic Algorithm

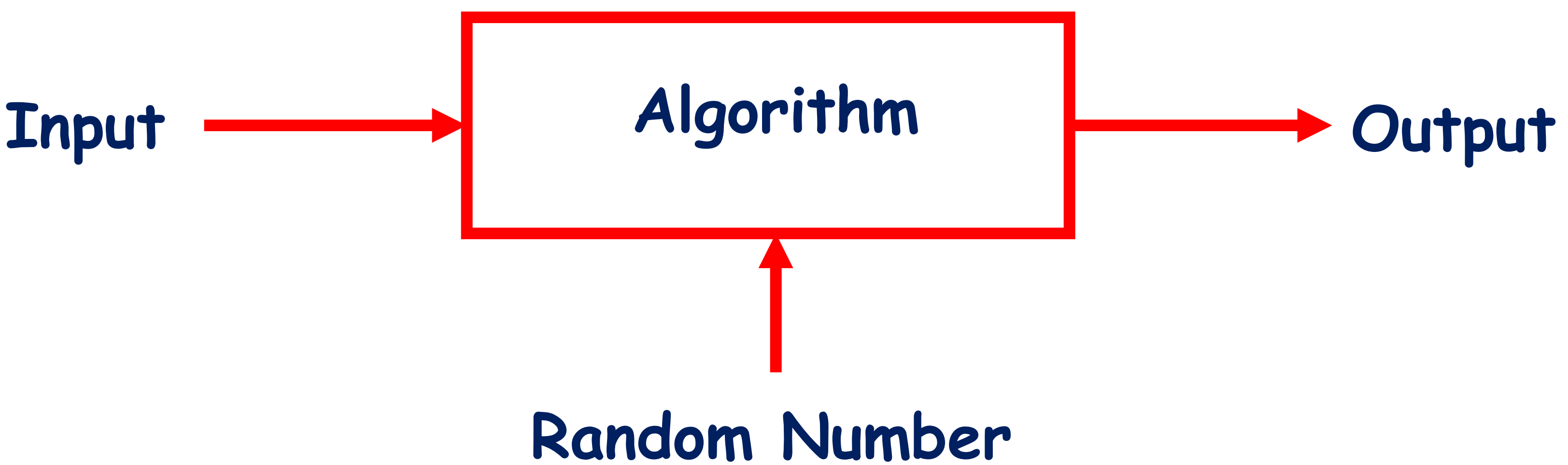
Identical behavior for different runs for a given input.

INPUT → ALGORITHM → OUTPUT

Randomized Algorithm

Behavior is generally different for different runs for a given input.

In Randomized Algorithm, the behavior can vary even for fixed input.



Randomized Algorithm

Design algorithm + analysis to show that this behavior is likely to be good on every input.

The likelihood is over the random numbers only.

Randomized Algorithm

Not to be confused with the Probabilistic Analysis of Algorithms.

Probabilistic Analysis of Algorithms

RANDOM INPUT → ALGORITHM → OUTPUT DISTRIBUTION

Here, Input is assumed to be from a probability distribution.

Show that the algorithm works for most inputs

What is Randomized Algorithm?

An algorithm whose behavior is determined not only by its input but also by the values produced by a random-number generator is a randomized algorithm.

In addition to input, algorithm takes a source of random numbers and makes random choices during execution.

Behavior can vary even on a fixed input.

Why Randomized Algorithm?

Avoid worst-case behavior:

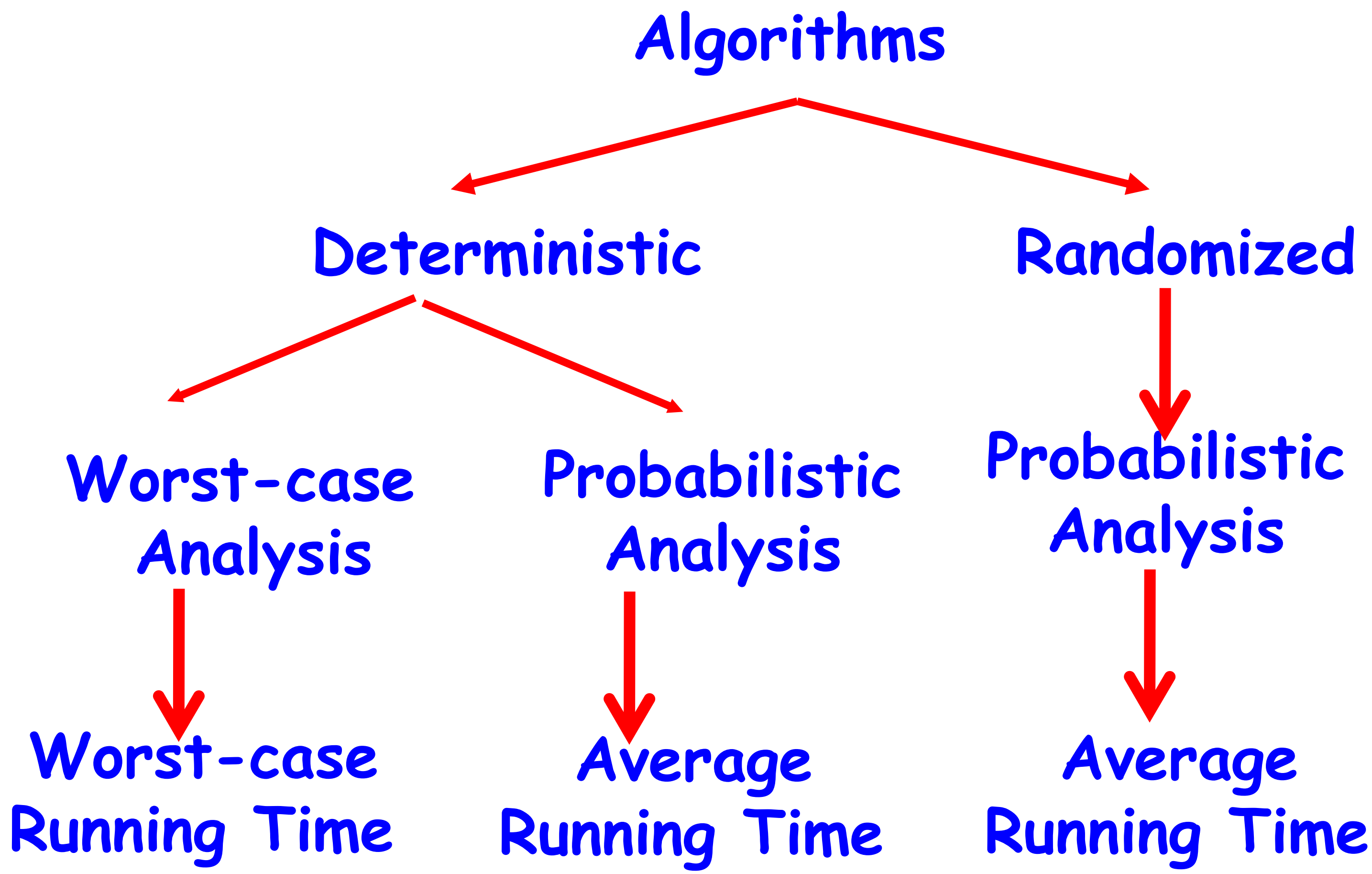
The hope is that the worst case occurs rarely, as no input elicits (draws out) the worst-case behavior.

Randomness can (probabilistically) guarantee average case behavior.

Efficient approximate solutions to intractable problems.

For many problems Randomized algorithms are often the simplest, fastest or both than deterministic algorithms.

They are not suitable only when guarantees on the running time are needed.


Algorithms
Deterministic
Randomized
Worst-case Analysis
Probabilistic Analysis
Probabilistic Analysis
Worst-case Running Time
Average Running Time
Average Running Time

Randomized Quick Sort

For any given input, the behavior of Randomized Quick Sort is determined not only by the input but also by the *random choices of the pivot.*

The pivot element is equally likely to be any of input elements.

Randomized Quick Sort

Instead of always using *A[r] as the* pivot, we will use a randomly chosen element from the array *A[p . . r].*

i	*p j*							*r*
	2	8	7	1	3	5	6	4
	1	2	3	4	5	6	7	8

We do so by exchanging element A[r] with an element chosen at random from A[p . . r].

Randomized Quick Sort

```
QUICKSORT(A, p, r)
  if p < r
     then q ← PARTITION(A, p, r)
             QUICKSORT(A, p, q − 1)
             QUICKSORT(A, q + 1, r)
```

```
RANDOMIZED-QUICKSORT(A, p, r)
 if p < r then
   q ← RANDOMIZED-PARTITION(A, p, r)
        RANDOMIZED-QUICKSORT(A, p, q − 1)
        RANDOMIZED-QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
    x ← A[r]
    i ← p − 1
    for j ← p to r − 1
         do if A[ j ] ≤ x
              then i ←i + 1
              exchange A[i ] ↔ A[ j ]
    exchange A[i + 1] ↔ A[r]
    return i + 1
```

```
RANDOMIZED-PARTITION(A, p, r)
      i ← RANDOM(p, r)
      exchange A[r] ↔ A[i ]
      return PARTITION(A, p, r)
```

Randomized Quick Sort

Call RANDOMIZED-QUICKSORT(A, 1, 8)

i	*p j*							*r*
	2	8	7	1	3	5	6	4
	1	2	3	4	5	6	7	8

i	*p j*							*r*
	2	8	4	1	3	5	6	7
	1	2	3	4	5	6	7	8

```
RANDOMIZED-QUICKSORT(A, p, r)
 if p < r then
    q ← RANDOMIZED-PARTITION(A, p, r)
    RANDOMIZED-QUICKSORT(A, p, q − 1)
    RANDOMIZED-QUICKSORT(A, q + 1, r)
```

```
RANDOMIZED-PARTITION(A, p, r)
      i ← RANDOM(p, r),   let i=3
      exchange A[r] ↔ A[i ]
      return PARTITION(A, p, r)
```

```
PARTITION(A, p, r)
  x ← A[r]
  i ← p − 1
  for j ← p to r − 1
     do if A[ j ] ≤ x
          then i ←i + 1
          exchange A[i ] ↔ A[ j ]
   exchange A[i + 1] ↔ A[r]
   return i + 1
```

Quick-Sort Vs. Randomized Quick Sort

Quick Sort

Best Case: $\Theta(n \lg n)$

Worst case : $\Theta(n^2)$

Avg. case : $\Theta(n \lg n)$ assuming all permutations equally likely

Randomized Quick Sort

Best Case: $\Theta(n)$

Worst case : $\Theta(n^2)$ Avoid

Avg. case : $\Theta(n \lg n)$