

Randomized Algorithms

- Deterministic and (Las Vegas & Monte Carlo) Randomized Algorithms
- Probability Review
- Probabilistic Analysis of deterministic QUICK-SORT Algorithm
- RANDOMIZED-SELECT and RANDOMIZED-QUICK-SORT
- Max-Cut
- Min-Cut
- MAX-3-SAT and Derandomization
- Closest pair
- Hashing, Bloom filters, Streams, Sampling, Reservoir sampling, Sketch

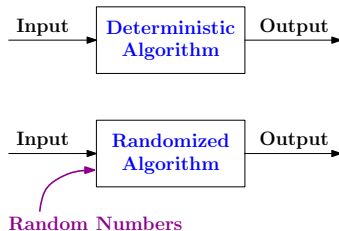
IMDAD ULLAH KHAN

Algorithmic Design Paradigm

- Greedy
- Divide-and-Conquer
- Dynamic Programming
- Network Flows

We have only seen **Deterministic Algorithms** so far

Randomized Algorithms incorporate randomness in their operation



Randomized Algorithm

Randomized Algorithm receives, in addition to the input, a random number stream to make random decisions during execution

- ▷ May give different results on the same input in different runs

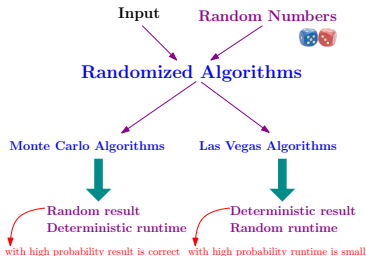
Often aims for properties like:

- Good average-case (expected) behavior
- Getting exact answers with high probability
- Getting answer close to the right answer with high probability
- Runtime is small with high probability

Advantages:

- Simple and elegant. Their output/runtime is good with high probability
- The execution time or space requirement is smaller than that of the best deterministic algorithm

Las Vegas and Monte Carlo Algorithms



■ Monte Carlo Algorithms

- Guaranteed to run in a fixed time
- Output is random
 - ▷ Output is correct with high probability
- e.g. Min cut algorithm

■ Las Vegas Algorithms

- Guaranteed to output the correct answer
- Running time is random
 - ▷ Runtime is small with high probability
- e.g: randomized quicksort, Closest pair

Las Vegas and Monte Carlo Algorithms

Input: An array A with $n/4$ 1's and $3n/4$ 0's

Output: An index k such that $A[k] = 1$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

Algorithm Deterministic

```
k ← 0
for i = 1 → n do
  if A[i] = 1 then
    k ← i
return k
```

Quality: correct
worst case runtime: $\frac{3n}{4}$



Algorithm Monte Carlo

```
k ← RANDOM(1 ⋯ n)
return k
```

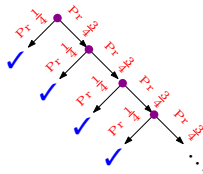
Quality: correct w.p $\frac{1}{4}$
worst case runtime: 1



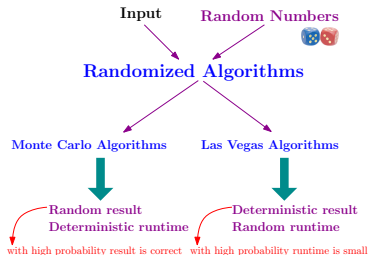
Algorithm Las Vegas

```
k ← 1
while A[k] ≠ 1 do
  k ← RANDOM(1 ⋯ n)
return k
```

Quality: correct
expected runtime: 4



Las Vegas and Monte Carlo Algorithms



■ Monte Carlo Algorithms

- Guaranteed to run in a fixed time
- Output is random
 - ▷ Output is correct with high probability
- e.g. Min cut algorithm

■ Las Vegas Algorithms

- Guaranteed to output the correct answer
- Running time is random
 - ▷ Runtime is small with high probability
- e.g: randomized quicksort, Closest pair

Can always convert a Las-Vegas algorithm into a Monte Carlo algorithm

- Stop the algorithm after a certain point
- but no method is known for the other way - needs efficient verification