Deterministic Algorithms

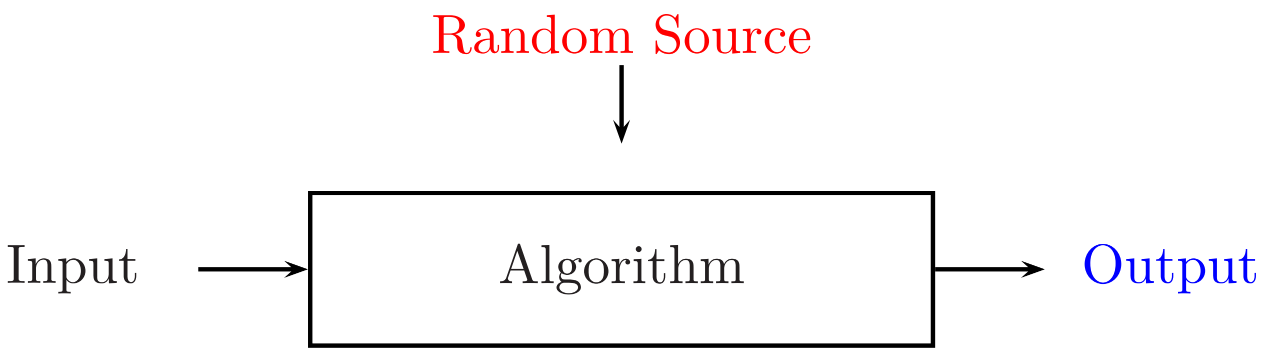
$$\text{Input} \longrightarrow \boxed{\text{Algorithm}} \longrightarrow \text{Output}$$

To prove that the algorithm always solves the problem as quickly as possible (in a polynomial number of steps)

Probabilistic Analysis of Deterministic Algorithms

- The input is assumed to be from a probability distribution.
- The expected performace of the algorithm is nice
- With high probability the algorithm performs well.

Randomized Algorithms



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

The running time of a randomized algorithm may vary from run to run.

Given a problem, we wish to design algorithms + analysis to show that the behaviour of the algorithm is likely to be good, on every input of the problem

Probabilistic Algorithms: Monte Carlo and Las Vegas

- A Monte Carlo algorithm runs for a ~~fixed~~ bounded number of steps and produces an answer that is correct with probability $\geq 1/2$. However, we may make the probability of error as small as we wish amplification.

- A Las Vegas algorithm allways produces the correct answer; its running time is a random variable whose expectation is bounded (by a polynomial).

- A Las Vegas algorithm can be made Monte Carlo by having it terminate with an arbitrary wrong answer, if it exceeds a time bound $t(n)$. Since Las vegas is unlikely to exceed its time bound, the resulting Monte Carlo is unlikely to give the wrong answer.
- There is not known universal method for making a Monte Carlo algorithm into a Las Vegas one.

Random Selection

The first randomization technique we will see is Random Selection:

The intuition behind this idea is that a single randomly selected individual is probably a typical representative of the entire population.

Therefore, random selection provides a good way to avoid selecting rare bad elements.

Verifying Polynomial Identities

Given polynomials $P(x)$ and $Q(x)$ we wish to verify if $P(x) \equiv Q(x)$

We only require that the evaluation of a polynomial can be done in polynomial time!

Recall: $P(x) \equiv Q(x)$ iff at every point x, $P(x) = Q(x)$.

Example Check if $(x+1)(x-2)(x+3)(x-4)(x+5)(x-6) \equiv x^6 - 7x^3 + 25$

Determinitic solution:

If P and Q are written explicitly, there is a linear algorithm (comparing the coefficients):

Transform $P(x)$ to canonical form $H(x) \equiv \sum_{i=0}^{d} c_i x^i$ where d is the degree of $P(x)$ or of $Q(x)$.

then $P(x) \equiv Q(x)$ iff coefficients of all monomial are equal.

But for example, Q can be given:

$$Q(\mathsf{x}) = \prod_{i<j|i,j\neq 1} (\mathsf{x}_i - \mathsf{x}_j) - \prod_{i<j|i,j\neq 2} (\mathsf{x}_i - \mathsf{x}_j) + \prod_{i<j|i,j\neq 3} (\mathsf{x}_i - \mathsf{x}_j) - \ldots$$

$$\pm \prod_{i<j|i,j\neq n} (\mathsf{x}_i - \mathsf{x}_j)$$

Randomized solution

Given $P(x)$ and $Q(x)$ of degree d, check polynomial identity in $S = [0, \cdots, 100d]$:

Schwartz-Zippel's Algorithm

Choose a random integer $r \in S = [0, \cdots, 100d]$
Compute $P(r)$ and $Q(r)$
if $P(r) = Q(r)$ **then return** TRUE
  else return FALSE

For example, if $r = 2$ then $P(2) = 420$ and $Q(2) = 33$, therefore as $P(2) \neq Q(x)$ the algorithm will return FALSE.

Notice that if $P(x) \equiv Q(x)$ any value of r will yield equality

For example $(x+1)(x-1) \equiv x^2 - 1$.

In the case $P(x) \not\equiv Q(X)$: **A bad choice of r may lead to a wrong answer**

For ex. to check if $x^2 + 7x + 1 = (x+2)^2$

If $r = 2$, $19 \neq 16$ so the two polynomials are different,

but $r = 1$, $9 = 9$!!

Theorem: Let $F(x) = P(x) - Q(x)$ have degree $\leq d$. If $F \not\equiv 0$ then

$$\mathbf{Pr}\left[F(r) = 0\right] \leq \frac{d}{|S|} = 1\%$$

A simple event: a choice of r; A sample space: all $[100d]$

The probability of a simple event: $\mathbf{Pr}\left[r\right] = \frac{1}{100d}$.

By the Fundamental Theorem of Algebra, a polynomial of degree d have at most d roots.

Therefore: a bad event is choosing a root of $F(X)$. There as there are at most d roots, then

$$\mathbf{Pr}\left[\text{bad event}\right] \leq \frac{d}{100d}.$$

- If the identity is correct, the algorithm always output the correct answer,
- If the identity is NOT correct, the algorithm outputs the wrong answer only if r is a root of $F(x) = P(x) - Q(x) = 0$

Amplification

We can decrease the error probability at the expense of increasing the run-time of the algorithm:

Run the algorithm k times
if each time is TRUE **output** TRUE
    else FALSE

The probability of a wrong answer in one run of the algorithm is 1/100. Runs of the algorithms are independent. It is enough to get a correct (negative) answer in one of the runs.

The probablility of error in k runs is $(\frac{1}{100})^k$.