Examples of Randomized Algorithms

Example 1: Matrix Multiplication Check

We consider the following problem.

> Matrix Multiplication Test (MULT-TEST)
> **Instance**: Matrices A, B, and C of size $n \times n$.
> **Question**: $A \times B \stackrel{?}{=} C$.

Although there is a matrix multiplication algorithm running less than $O(n^3)$ steps, it seems difficult to achieve the above test in $O(n^2)$ steps. On the other hand, randomness gives us the following simple yet efficient algorithm. (Here we simply count the number of arithmetic operations for running time.)

> **program** TestSoSo(A, B, C);
> choose r_1,...,r_n uniformly at random from $\{0, 1\}$,
> and let $r \leftarrow (r_1, ..., r_n)$;
> $u \leftarrow A \cdot (B \cdot r)$; $v \leftarrow C \cdot r$;
> **if** $u = v$ **then output**(yes);
> **else output**(no);

For a given input A, B, and C, it is easy to see that the algorithm outputs 'yes' if $A \times B = C$. On the other hand, we can show that if $A \times B \neq C$, then with probability $\geq 1/2$, u and v differs and the algorithm outputs 'no'.

Lemma 1.1. For any A, B, and C, if $A \times B \neq C$, then the probability that TestSoSo(A, B, C) wrongly outputs 'yes' is at most $1/2$.

In order to improve the precision, let us now consider an algorithm Test_k that executes TestSoSo for k times. More specifically, for given A, B, and C, Test_k executes TestSoSo(A, B, C) for k times, and concludes 'yes' if *all* k answers are 'yes', and 'no' otherwise. Then we can reduce the error probability very much.

Theorem 1.2. The error probability of Test_k is at most 2^{-k}. Thus, for given ϵ, by using Test_k with $k = \lceil \log_2 \epsilon \rceil$, we can solve MULT-TEST with error probability $\leq \epsilon$.

Proof. Again Test_k makes no error for the case $A \times B = C$. Thus, we consider any input A, B, and C such that $A \times B \neq C$.

Then, on this input, Test_k makes error (i.e., gives 'yes' answer) if and only if TestSoSo(A, B, C) gives 'yes' answer for k times. On the other hand, for each execution of TestSoSo(A, B, C), TestSoSo gives 'yes' answer with probability at most $1/2$. Since TestSoSo are executed independently, Test_k makes error with probability at most $(1/2)^k$. □

Example 2: Smallest Enclosing Disk

Next, consider the following problem.

Smallest Enclosing Disk
Instance: A finite set D of points in R^2.
Question: The smallest disk enclosing D (its center and radius).

Gärtner and Welzl formulated[1] some combinatorial sampling lemma that can be used for designing various algorithms in mathemtical programming, computational geometry, etc. Here we explain this by using a simple combinatorial problem — smallest enclosing disk. We give a simple but efficient randomized algorithm.

Let S be any finite set of n elements. Below we often call each element of S *point* or *instance* and call S *space* or *domain*. Let ϕ be any function defined on subsets of S. These S and ϕ are fixed throughout this section.

Let T be any subset of S. For any point p in T, we say that p is *extreme* (or, p is an *extremer* in T) if $\phi(T - \{p\}) \neq \phi(T)$, and we say that p *violates* T (or, p is a *violator* of T) if $\phi(T \cup \{p\}) \neq \phi(T)$.

Lemma 2.1. (Gärtner and Welzl)
For any r, $1 \leq r \leq n$, define a_r and b_r by

$$\begin{aligned} a_r &= \text{expected number of violators of } R, \text{ and} \\ b_r &= \text{expected number of extremers in } R, \end{aligned}$$

when R is a random subsets of S consisting of r points. Then for any r, $1 \leq r < n$, we have

$$a_r(r+1) \;=\; b_{r+1}(n-r).$$

Remark. The lemma holds even if S is a multi set. (A *multi set* is a set having the same element more than once; for example, $\{7, 1, 2, 1, 3, 5, 7\}$.

Proof. For any subset T of S and any point p, we have

$$p \text{ violates } T \iff p \text{ is extreme in } T \cup \{p\}.$$

With this key fact, we have

$$\begin{aligned} \binom{n}{r} a_r &= \sum_{R \subseteq S : |R| = r} \sum_{p \in S - R} [\, p \text{ violates } R \,] \\ &= \sum_{R \subseteq S : |R| = r} \sum_{p \in S - R} [\, p \text{ is extreme in } R \cup \{p\} \,] \\ &= \sum_{Q \subseteq S : |Q| = r+1} \sum_{p \in Q} [\, p \text{ is extreme in } Q \,] \;=\; b_{r+1} \binom{n}{r+1}. \end{aligned}$$

□

[1] B. Gärtner and E. Welzl, Linear programming – Randomization and abstract frameworks, in *Proc. 13th Ann. Symp. on Theoret. Aspects of Comput. Sci.*, Lecture Notes in Computer Science 1046 (1996), 669−687.

This lemma may be useful for estimating certain expected values. For example, consider the Smallest Enclosing Disk problem. Here we define, for any $T \subseteq S$, $\phi(T)$ to be the smallest disk containing T. Then we can show the following fact; that is, $b_r \leq 3$ for any r, $1 \leq r \leq n$. Thus, when choosing a set R of r points from S, the expected number of violators of R is at most $3(n-r)/(r+1)$.

Fact 1. Any $T \subseteq S$ has at most three extremers.

Remark. In most cases, those extremers of T are *three* points on the edge of the disk $\phi(T)$, and the magic number *three* is related to the fact that any three points (not on the same line) determine one circle that go through these three points. But be careful; this is not the proof for the above fact!

So this lemma may be useful for estimating certain expected values, but its importance is to derive the following sampling algorithm. (In the algorithm, we use parameters r and c that will be fixed later.)

```
Sampling Algorithm (w.r.t. S and φ)
  SS ← S;   % SS is a multi set of points.
  while true do
    R ← random subset of SS with r points;
    % I.e., select r points according to their "weights".
    if R has no violator then halt;
    if the "weight" of violators ≤ |SS|/3c then (*)
      SS ← SS ∪ {p ∈ SS : p is a violator of R};
  end-while;
```

In this algorithm, some points are put into SS several times. We intuitively consider the number of duplications of each point as its *weight*. That is, the above sampling algorithm doubles the weight of violators, and points are selected to R according to their weights.

For any subset C of S, we say that C is *critical* (or, a *critical set* of points) if for any $T \subseteq S$ that has a violator, some point in C is a violator of T. Note that at each while-iteration of the algorithm the algorithm always double the weight of some point in C.

Lemma. (Termination Lemma)
Suppose that there exists a critical set C of size c_0 and that $b_r \leq b_0$ for any r, $1 \leq r \leq n$. If we execute Sampling Algorithm with $r = 6c_0b_0$ and $c = c_0$, then it terminates "on average" within $2\lceil 3c_0 \ln n \rceil$ steps. That is, the expected number t of the total while-loop iteration is bounded by

$$t \;\leq\; 2\lceil 3c_0 \ln n \rceil.$$

Proof. We call a while-loop iteration *successful* if the if the if-condition $(*)$ holds. First we bound the number of successful iterations.

For any $k \geq 0$, suppose that we have executed k successful iterations (and still does not halt). Let SS_k denotes SS after the kth successful iterations ($SS_0 = S$), and let n_k is the number of points in SS_k, i.e., the total weights.

Then we have

$$n_k \ \leq \ n_{k-1} + n_{k-1}/3c_0.$$

Hence, we have

$$n_k \ = \ n_{k-1} + n_{k-1}/3c_0. \ \leq \ n_{k-1}\left(1 + \frac{1}{3c_0}\right) \ \leq \ n_0\left(1 + \frac{1}{3c_0}\right)^k \ < \ n_0 \cdot e^{k/3c_0},$$

where n_0 is $|SS_0| = |S| = n$.

On the other hand, the weight of some point in C gets doubled at each successful iteration. Thus, there is some point in C whose weight becomes $2^{k/c_0}$ after k successful iterations. Then n_k must be larger than $2^{k/c_0}$. Thus, we have

$$2^{k/c_0} \ \leq \ n_k \ < \ n \cdot e^{k/3c_0}.$$

This implies that $k < 3c_0 \ln n$. Therefore, the number of successful iterations should be at most $\lceil 3c_0 \ln n \rceil$.

Next we estimate the total number of iterations. It follows from the Sampling Lemma, the average weight of violators (at each iteration) is at most

$$\frac{b_0}{r+1}(n' - r) \ < \ \frac{b_0}{r}n' \ = \ \frac{1}{6c_0}n',$$

where n' is the total weight at this point, i.e., $n' = |SS|$. Thus, by using Markov's inequality, the probability that the weight of violators exceeds $n'/3c_0$ is less than $1/2$. From this, it is easy to show that the expected number of iterations is at most twice as large as the number of successful iterations. Therefore, we have $t \leq 2\lceil 3c_0 \ln n \rceil$. □

Example 3: Polynomial Identity Test

Next, consider the following problem.

Polynomial Identity Test
Instance: A polynomial $P(x_1, \ldots, x_n)$ of degree d
on some finite field F with $q >> d$ elements.
Question: Test whether $P(x_1, \ldots, x_n)$ is the zero polynomial;
that is, $P(x_1, \ldots, x_n) = 0$ for all points in F^n.

Given two polynomials Q_1 and Q_2, we can test whether they are identical or not by testing whether $P := Q_1 - Q_2$ is the zero polynomial. Thus, the above problem is called "Polynomial Identity Test."

Theorem 3.1. Consider any finite subset S of F, and let $s = |S|$. Let us select $r_1, \ldots, r_n$ uniformly at random from S. Then for any nonzero polynomial $P(x_1, \ldots, x_n)$ of degree $\leq d$, we have

$$\Pr_{r_1,\ldots,r_n}[\, P(r_1, \ldots, r_n) = 0 \,] \;\leq\; \frac{d}{s}.$$