Introduction to algorithmic Fall 2017

IFT2125-6001 TA: Maëlle Zimmermann

Demonstration 12

1

Question: Calculate the expected number of rolls of two dice before getting a pair of 1. Ditto to get two digits that goes to 7.

Solution: This problem amounts to calculating the expectation of a parametric geometrical law p. Indeed the geometric law considers the problem of repeating a test whose probability of success is p, and the geometric random variable denotes the rank of the first succ`es.

Let p be the probability of obtaining the event sought during a throw. The hoped number of throws n is given by:

$$n = 1 \cdot p + 2(1-p)p + 3(1-p)^2 p + \dots$$
$$= \sum_{k=1}^{\infty} k(1-p)^{k-1} p.$$

Now we have for $x \in [0,1)$:

$$f(x) = \sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \qquad \text{(geometric series)}$$
$$f(x) = \sum_{k=1}^{\infty} kx^{k-1} = \frac{1}{(1-x)^2}.$$

Like $n = p\sum_{k=1}^{\infty} k(1-p)^{k-1}$, we get

$$n = \frac{p}{(1-(1-p))^2} = \frac{1}{p}.$$

Since the probability of getting a pair of 1 is given by p = 1/36, the number hoped for throws is 36. Similarly, the probability of getting a pair of numbers that up to 7 is 1/6, so the expected number of throws is 6.

2

Question: A skewed coin throws stack throws with probability $p = \frac{1}{2}$ and face with probability q = 1-p. Each coin throw is independent of the previous ones. The value of p is not known. Find a process to get a sequence from the room of random bits unbiased.

Solution: To generate a random bit (0 or 1), simply make two successive throws of the biased coin. If two piles or two faces are obtained, we start again. If not,

- if we get stack then face, we generate bit 0,
- if we get face and stack, we generate bit 1.

These two events have an equal probability of being realized, namely pq. The probability to get one or the other knowing that both coin throws produced a pile and a face is $\frac{1}{2}$. An unbiased random bit sequence is obtained by repeating this method.

3

Question: Consider a problem that has more than two possible answers. So he is possible that for some instances of the problem, several answers are correct. Let MC3 be the algorithm that executes a non-biased MC algorithm three times and returns the majority answer if there is one, or, which in case of a tie, randomly returns uniformly one of the answers obtained. Show that in this case, the amplification can decrease the probability that the algorithm returns a correct answer. More precisely, show that there is a 75% -correct Monte-Carlo MC algorithm such as MC3 is not 71% -correct.

Solution: Let $A_1, A_2, ..., A_n$ have good answers and B a bad answer. Is MC the algorithm that returns A_i with probability $\frac{3}{4n}$ for all i = 2, ..., n, and B with probability $\frac{1}{4}$. Note that MC returns a good answer with probability $n \frac{3}{4n} = \frac{3}{4}$ and so is 75% -correct.

The probability that MC3 returns a good answer is $p(n) = f(n) + g(n) + \frac{2}{3} h(n)$ where

- f (n) is the probability that MC3 gets three good answers
- g (n) is the probability that MC3 gets a wrong answer and twice the same right answer

- h (n) is the probability that MC3 draws a wrong answer and two good answers different.

We have

$$f(n) = \underbrace{\frac{3}{4}}_{\text{right answer}} \underbrace{\frac{3}{4}}_{\text{right answer}} \underbrace{\frac{3}{4}}_{\text{right answer}}$$

$$g(n) = 3 \left(\underbrace{\frac{1}{4}}_{\text{wrong answer}} \cdot \underbrace{\frac{3}{4} \cdot \frac{3}{4n}}_{\text{2 same good answers}} \right)$$

$$h(n) = 3 \left(\underbrace{\frac{1}{4}}_{\text{wrong answer}} \cdot \underbrace{\frac{3}{4} \cdot \left(\frac{3}{4} - \frac{3}{4n}\right)}_{\text{2 different answers}} \right)$$

So,

$$\begin{aligned} p(n) &= f(n) + g(n) + \frac{2}{3} h(n) \\ &= \left(\frac{3}{4}\right)^3 + \left(\frac{3}{4}\right)^2 \cdot \frac{3}{4n} + \frac{2}{3} \cdot \left(\frac{3}{4}\right)^2 \cdot \left(\frac{3}{4} - \frac{3}{4n}\right) \\ &= \frac{27}{64} + \frac{27}{64n} + \frac{3(3n-3)}{32n} \\ &= \frac{45n+9}{64n} \end{aligned}$$

For n = 21, we have p (21) = 954/1344 = 159/224 <0.7099. So for a problem with 21 correct answers, MC3 is not 71% -correct. In fact, we can even prove that from more than 21 correct answers, the MC3 algorithm is not 71% -correct.

4

Question: Give a Monte Carlo algorithm to determine if two polynomials p and q on Z, presented by arithmetic circuits, are equal. We can use without proof the fact that the degree of a polynomial is limited by the height of the circuit.

Solution: Consider the following example:

The circuits C_p and C_q respectively calculate the following polynomials:

$$p(x_1, x_2, x_3) = ((6 + (x_1 x_2)) - (x_1 x_2)(7 - x_3)) + 3(x_1 x_2)(7 - x_3))$$
$$q(x_1, x_2, x_3) = (6 + (x_1 x_3)) - 3(x_2 - x_3)(7x_2 - 2)$$

To determine if $p = q$, we set $r = p-q$ and test if r is zero. Other said, we test if the circuit $C_r = C_p - C_q$ computes the polynomial 0. To do this, we choose a random assignment of $x_1, x_2, ..., x_m$, that is, a set of numbers $a_1, a_2, ..., a_m \in Q$ and we evaluate $r(a_1, a_2, ..., a_m)$. If $r(a_1, a_2, ..., a_m) = 0$, then it is impossible that r is the polynomial 0. On the other hand if $r(a_1, a_2, ..., a_m) = 0$, it is all of even if r is not polynomial 0, if we have chosen a root of r.

In the case where um = 1, that is to say that the polynomial is univariate, it suffices to choose a_1 in a set of size $k \cdot \deg(r)$ to have an error probability of at most $1 / k$. In indeed, r has at most deg (r) roots.

In the multivariate case, the number of roots of r is no longer limited by deg (r). In fact, r can have an infinity of roots. Indeed, the polynomial $p(x_1, x_2, x_3) = x_1$ has an infinity of roots of the form $(0, x_2, x_3)$. We will see thanks to a lemma that the Univariate case strategy is generalized to multivariate polynomials.

Let h_p and h_q be the height of C_p and C_q respectively. We give the algorithm following:

1. Set $S = \{1,2, ..., \max 4 \cdot (2^{h_p}, 2^{h_q})\}$
2. Choose randomly and independently at $_1$, $a_2, ..., a_m \in$ uniform (S)

3. Ask $r = p - q$

4. If $r(a_1, a_2, ..., a_m) = 0$ then return yes, otherwise return no.

When the algorithm responds no, we are sure that p and q are not equal.
When the algorithm responds yes, it is possible that p and q are not equal and that the algorithm is wrong. To evaluate the probability that the algorithm is wrong in the multivariate case, we need the following lemma:

Schwartz-Zippel's Lemma: let $p \in Q[x_1, x_2, ..., x_m]$ be a nonzero polynomial. Let $S \subset Q$ a finite subset and has $_1, a_2, ..., a_m$ chosen in S uniformly and independent. So,

$$\Pr[p(a_1, a_2, ..., a_m) = 0] \leq \frac{\deg(p)}{|S|}. \tag{1}$$

Proof: The lemma is demonstrated by induction on the number of variables.

Base case ($m = 1$): in this case, p is univariate and has at most deg (p) roots, so we get that

$$\Pr[p(a_1) = 0] \leq \frac{\deg(p)}{|S|}.$$

Induction stage: we can put each power of x_1 in p and get the next rewrite.

$$p(x_1, x_2, ... x_m) = \sum_{i=0}^{d} x_1 \, p_i(x_2, ... x_m)$$

Since p is non-zero, there exists at least one such that p_i is non-zero. Let i be the largest such index, then by induction hypothesis:

$$\Pr[p_i(a_2, ... a_m) = 0] \leq \frac{\deg(p_i)}{|S|} \leq \frac{\deg(p) - i}{|S|} \tag{2}$$

If $p_i(a_2, ... a_m) = 0$, then $p(x_1, a_2, ..., a_m)$ is a polynomial with a variable (x_1) of degree i, because i is the largest degree of x_1 nonzero in p. Thus by induction hypothesis:

$$\Pr[p(a_1, a_2, ..., a_m) = 0 \mid p_i(a_2, ... a_m) = 0] \leq \frac{i}{S} \tag{3}$$

If we call At the event $p(a_1, a_2, ..., a_m) = 0$ and B the event $p_i(a_2, ... a_m) = 0$, then we have

$$\begin{aligned}
\Pr[A] &= \Pr[A \mid B]\Pr[B] + \Pr[A \mid B^c]\Pr[B^c] \\
&\leq \Pr[B] + \Pr[A \mid B^c] \\
&\leq \frac{\deg(p) - i}{|S|} + \frac{i}{|S|} \quad \text{by (2) and (3)} \\
&= \frac{\deg(p)}{|S|}.
\end{aligned}$$

Which concludes the proof. D

Consider now the case where the algorithm above answers yes. Let w be the probability that the algorithm is wrong, in other words that $r(a_1, a_2, \ldots a_m) = 0$ knowing that r is no no. We have:

$$\begin{aligned}
w &\leq \frac{\deg(r)}{|S|} && \text{by (1)} \\
&\leq \frac{\max(\deg(p), \deg(q))}{|S|} && \text{because } \deg(r) \leq \max(\deg(p), \deg(q)) \\
&= \frac{\max(\deg(p), \deg(q))}{4 \cdot \max(2^{hrs_p}, 2^{hrs_q})} && \text{def. from S} \\
&\leq \frac{\max(\deg(p), \deg(q))}{4 \cdot \max(\deg(p), \deg(q))} && \text{because } \deg(p) \leq 2^{h_p} \text{ and } \deg(q) \leq 2^{h_q} \\
&= \frac{1}{4}
\end{aligned}$$

So the algorithm is wrong with probability at most $\frac{1}{4}$ when he answers yes.

Execution time analysis: We analyze the execution time of the algorithm according to the size of the entrance, ie according to the degree of p and q and according to the number of variables m. If you can draw a coin in O (1), you can select a number a_i in S by binary search in time $\log_2 |S| = \log_2(4 \cdot \max(2^{h_p}, 2^{h_q})) = 2 + \max(h_p, h_q)$ coin shots. The execution time of the random selection algorithm is therefore in $O(m \max(h_p, h_q))$. Evaluating the circuit r is done in polynomial time. The weather execution of the algorithm is therefore polynomial in the size of the input.