# HW4R — Randomized algorithms & Random structures

MPRI 1.24 — Wed. Dec. 16, 2015 - **Due on Wed. Jan. 6, 2016**

You are asked to complete the exercise marked with a [★] and to send me your solutions at:
nicolas.schabanel@cnrs.fr
(or drop it in my mail box at the 4th floor of Sophie Germain) on **Wed. Jan. 6, 2016**.

■ **Exercise 1 (Using fingerprints to check matrix multiplication).** [★]
The best known algorithms for multiplying two $n \times n$-matrices take time $O(n^\alpha)$ for some $\alpha > 2$. However, there is a simple randomized algorithm for verifying matrix products with high probability in $O(n^2)$ time.

Suppose I claim that $AB = C$ where $A$, $B$, and $C$ are integer-valued $n \times n$-matrices. You can confirm this by applying both sides of this equation to a random vector $v$, and checking that $ABv = Cv$.

► ***Question* 1.1)** *Describe how to check if $ABv = Cv$ in $O(n^2)$ time, i.e. without computing the $n \times n$-matrix product $AB$.*
Answer. ▷ Since the matrix product is associative: $ABv = A(Bv)$. Evaluating $A(Bv)$ and $Cv$ consists then in computing three vector-matrix products which take $O(n^2)$ time in total by the trivial algorithm. ◁

We desire to improve furthermore the checking procedure by using fingerprints. Assume that $AB \neq C$ and let $\alpha = \max_{i,j}\{|A_{ij}|, |B_{ij}|, |C_{ij}|\}$ bound the maximum coefficient in $A$, $B$ and $C$.

► ***Question* 1.2)** *Explain how to choose a prime number $p = O(\log(n\alpha) \log\log(n\alpha))$ such that with probability at least $\frac{9}{10}$, we have $AB - C \neq 0 \mod p$.*
▷ Hint. *Use that the $k$th prime number $p_k$ verifies $0.91k \ln k < p_k < 1.7k \ln k$, as proved by Felgner in 1990.*
Answer. ▷ As all coefficients in $A$, $B$ and $C$ are at most $\alpha$, all coefficients in $AB - C$ are at most $n\alpha^2 + \alpha$. As $M = AB - C \neq 0$, then there is a non-zero coefficient $M_{ij} \neq 0$. Since $|M_{ij}| \leqslant n\alpha^2 + \alpha$, it has at most $\beta = \log_2(n\alpha^2 + \alpha)$ prime factors. It follows that if we pick a prime number $p$ uniformly at random between 2 and $k = 1.7 \cdot 10\beta \cdot \ln(10\beta) = O(\log(n\alpha) \log\log(n\alpha))$ (so that there are at least $10\beta$ prime numbers between 2 and $k$), $p$ divides $M_{ij}$ with probability at most $\frac{1}{10}$, , i.e. $\Pr\{ M_{ij} = 0 \mod p \} \leqslant \frac{1}{10}$. It follows that with probability at least $\frac{9}{10}$, $AB - C \neq 0 \mod p$. ◁

Assume now that we are lucky and $AB - C \neq 0 \mod p$.

► ***Question* 1.3)** *Show that if $v$ is chosen uniformly at random from $\{0, 1, \ldots, p-1\}^n$, then the probability that $ABv = Cv \mod p$ holds is at most $1/p$.*
Answer. ▷ If $M = AB - C \neq 0$, then the application $v \mapsto (AB - C)v$ is a non-zero linear application over the vector space $\mathbb{Z}_p^n$ ($p$ is prime). It follows that $\dim\ker M + \dim\mathrm{Im}\, M = n$, and since $\dim\mathrm{Im}\, M \geqslant 1$ then $\dim\ker M \leqslant n-1$ and $\#\{v \in \mathbb{Z}_p^n : (AB-C)v = 0\} \leqslant p^{n-1}$. It follows that if $AB \neq C$, the probability that $ABv = Cv$ for an uniform random vector $v \in \mathbb{Z}_p^n$ is at most $p^{n-1}/p^n = 1/p$. ◁

■ **Exercise 2 (FPT algorithm for spotting $k$ disjoint triangles).** [★]
Given $G = (V, E)$ an undirected graph ($n = |V|$ and $m = |E|$) and $k$ an integer, we are looking for $k$ vertex-disjoint triangles in $G$. Note that this problem is $NP$-complete when $k$ is part of the input. We are looking for an algorithm of time complexity $O(f(k)n^a m^b)$ where the exponents $a$ and $b$ are constant, independent of $k$. Such an algorithm is called FPT for Fixed Parameter Tractable, which means that the complexity is a fixed-degree polynomial in the size of the input for any fixed value of the parameter $k$. Consider the following randomized algorithm:

**Algorithm 1** FPT randomized algorithm for $k$ disjoint triangles

- Choose independently for each vertex $u$ a color $c_u \in \{1, \ldots, 3k\}$ uniformly at random.
- **return** "Yes" if there is a *colorful* solution, i.e. a set of $k$ triangles whose $3k$ vertices use exactly once each color; **return** "I don't know" otherwise.

► ***Question* 2.1)** *Show that if $G$ contains $k$ disjoint triangles, then the probability that the algorithm answer "Yes" is at least $e^{-3k}$.*
▷ Hint. *use that : $k! \geqslant (k/e)^k$ for all $k$.*

Answer. ▷ Assume that $G$ contains $k$ disjoint triangles. Then, the probability that each of their $3k$ vertices gets a different color is $(3k)!/(3k)^{3k} \geqslant e^{-3k}$. ◁

► ***Question* 2.2)** *How many times should you run this algorithm to improve success probability to $1/2$?*

Answer. ▷ Note that if the algorithm outputs "Yes", then it has found $k$ colorful triangles which are thus necessarily vertex-disjoint. Then, if $G$ does not have $k$ disjoint triangles, the algorithm will always output "I don't know".

Assume that we repeat $t$ times the algorithm and answer "Yes" whenever one of the $t$ executions outputs "Yes", and "No" otherwise. If $G$ does not have $k$ disjoint triangles, the new algorithm will always answer "No" and thus the algorithm is always successful. If $G$ has $k$ disjoint triangles, $k$ disjoint triangles are detected with probability at least $e^{-3k}$ for each of $t$ independent executions. The new algorithm will then output "Yes" with probability at least $1 - (1 - e^{-3k})^t = 1 - \exp(t \ln(1 - e^{-3k}) \geqslant 1 - \exp(-t \cdot e^{-3k}) \geqslant \frac{1}{2}$ for $t \geqslant e^{3k} \ln 2$. ◁

In order to check whether a colorful solution exists, we propose to try all permutations $\pi$ on $\{1, \ldots, 3k\}$ and check if there is are triangles of colors $(\pi_1, \pi_2, \pi_3), \ldots, (\pi_{3k-2}, \pi_{3k-1}, \pi_{3k})$.

► ***Question* 2.3)** *Describe an algorithm that decides if such a collection of triangles exists. (Explicit the exact data structure you are using). What is the overall expected time complexity in $k$, $n$ and $m$, of the algorithm that uses this method to return $k$ disjoint triangles with probability at least $1/2$ if they exists in $G$? What is the time complexity if $k$ is fixed?*

Answer. ▷ Deciding if a given graph $H$ has a triangle takes $O(m_H n_H)$ time by scanning all of its edges $uv$ and all the vertices $w \neq u, v$.[1]

To solve our problem, we scan the edges of $G$ and dispatch them into $k$ subgraphs $H_i$, with $1 \leqslant i \leqslant k$, where $H_i$ contains only the vertices of colors $\pi_{3i-2}, \pi_{3i-1}, \pi_{3i}$ and the edges between them; the other edges are discarded. We then run $k$ times the triangle-detection algorithm above, one for each $H_i$, and answer "Yes" iff a triangle is detected in every $H_i$. The overall time complexity for the test is $n+m+\sum_{i=1}^{k} O(m_{H_i} n_{H_i}) = O(mn)$.

The overall complexity of our $k$-disjoint triangles algorithm with success probability $\geqslant \frac{1}{2}$ is then $O((3k)!e^{3k}mn)$, that is to say $O(mn)$ if $k$ is a fixed constant. ◁

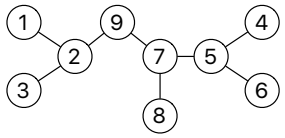
[1] There exists more efficient algorithms for detecting triangles, see wikipedia "Triangle-free graph".

■ **Exercise 3 (Binomial laws composition).** Show that when the laws considered are independent, $\mathsf{Bin}(\mathsf{Bin}(n,p),q) \sim \mathsf{Bin}(n,pq)$ for all $n \in \mathbb{N}$ and $p, q \in [0,1]$.
Answer. ▷ Let $X \sim \mathsf{Bin}(n,p)$ and $Y \sim \mathsf{Bin}(X,q)$ where the two binomial laws are independent. Then, for all $y \in \{0,\dots,n\}$

$$\begin{aligned}
\Pr\{Y=y\} &= \sum_{x=y}^{n} \Pr\{X=x\} \Pr\{Y=y \mid X=x\} \\
&= \sum_{x=y}^{n} \binom{n}{x} p^x (1-p)^{n-x} \binom{x}{y} q^y (1-q)^{x-y} \\
&= \sum_{t=0}^{n-y} \frac{n!}{(y+t)!\,(n-y-t)!} p^{y+t} (1-p)^{n-y-t} \frac{(y+t)!}{y!\,t!} q^y (1-q)^t \\
&= (pq)^y (1-p)^{n-y} \frac{n!}{y!(n-y)!} \sum_{t=0}^{n-y} \frac{(n-y)!}{(n-y-t)!\,t!} \left(\frac{p(1-q)}{1-p}\right)^t \\
&= (pq)^y \frac{n!}{y!(n-y)!} (1-p)^{n-y} \left(1+\frac{p(1-q)}{1-p}\right)^{n-y} \\
&= \binom{n}{y} (pq)^y (1-pq)^{n-y}.
\end{aligned}$$

◁

■ **Exercise 4 (Prüfer sequence).** Consider a (unrooted) tree with $n$ nodes labelled from $1$ to $n$. The Prüfer sequence encoding this tree is obtained by removing vertices from the tree until only two vertices remain. Specifically, at step $i$, remove the leaf with the smallest label and set the $i$th element of the Prüfer sequence to the label of this leaf's neighbor. For instance, the Prüfer sequence for the tree bellow is $2, 2, 5, 5, 7, 7, 7$.




► ***Question* 4.1)** *Show that every labelled unrooted tree corresponds a unique sequence in $\{1,\dots,n\}^{n-2}$ and reciprocally.*
▷ Hint. *First, show how to compute the degree of every node from a given Prüfer sequence.*
Answer. ▷ (Sketch) By induction, every node of degree $d$ appears $d-1$ times in the sequence. To decode the sequence, first compute the degrees and then select iteratively the degree-$1$ node with the smallest label and decreases its degree and its neighbor's degree. Connect the last two nodes with positive degrees. ◁

► ***Question* 4.2)** *Conclude that there are $n^{n-2}$ labelled unrooted trees with $n$ nodes.*
Answer. ▷ (Sketch) There are $n^{n-2}$ Prüfer sequences. ◁

► ***Question* 4.3)** *More precisely, show that there are*

$$\binom{n-2}{d_1-1,\ d_2-1,\ \dots,\ d_n-1} = \frac{(n-2)!}{(d_1-1)!(d_2-1)!\cdots(d_n-1)!}$$

*labelled unrooted trees with $n$ nodes whose $i$th vertex has degree $d_i$.*
Answer. ▷ (Sketch) Remark that $i$ appears exactly $d_i-1$ times in the Prüfer sequence. Start with the sequence of length $n-2$ where each $i$ appears exactly $d_i-1$ times with $i$

increasing from left to right. There are $(n-2)!$ possible shuffles of this sequence and for each $i$, $(d_i - 1)!$ shuffles of this shuffle that leaves it unchanged. The formula follows. $\triangleleft$