Advanced Complexity Exam (2020-21)

You may answer in English or in French. If you answer in French, and you do not know the French equivalent of some English word that I am using, by all means do not invent your own translation: use the English word instead in that case.

Let me stress the value of rigor. In all arguments, you must: (a) stress the important idea; (b) give explicit values for all required bounds (errors, probabilities, time and space usage, and so on); (c) give explicit references to the results you use, preferably by name (e.g., "the Immerman-Szelepcsényi theorem").

In the whole question, # is a a separator (a letter) different from all other letters. Section 1 and Section 2 are largely independent, but not completely.

1 Lipton fingerprints and graph coloring

The only interesting question in this section is **Question 4**, and will account for more that half of the points of the whole question. (My solution to it takes almost two pages, which is far more than any of the other questions.) The preceding questions are simply warm-ups, the questions after that are easy variants, and will not account for much either.

Two lists $[x_1, x_2, \ldots, x_k]$ and $[y_1, y_2, \ldots, y_{k'}]$ are equal *up to permutation* if and only if $k = k'$ and there is a permutation π of $\{1, 2, \cdots, k\}$ such that $y_{\pi(1)} = x_1$, $y_{\pi(2)} = x_2$, ..., $y_{\pi(k)} = x_k$.

Given a prime number p, we will admit without proof that two lists $\ell \overset{\text{def}}{=} [x_1, x_2, \ldots, x_k]$ and $\ell' \overset{\text{def}}{=} [y_1, y_2, \ldots, y_{k'}]$ of natural numbers in $[0, p-1]$, with $k, k' < p$, are equal up to permutation if and only the *Lipton polynomial*:

$$P(\ell, \ell') \overset{\text{def}}{=} \prod_{i=1}^{k} (X - x_i) - \prod_{j=1}^{k'} (X - y_j), \tag{1}$$

which is a polynomial in $\mathbb{Z}/p\mathbb{Z}[X]$, is the zero polynomial. Indeed, the Lipton polynomial is zero if and only if the two polynomials $\prod_{i=1}^{k}(X - x_i)$ and $\prod_{j=1}^{k'}(X - y_j)$ have the same roots, counting multiplicities. We require $k, k' < p$ in order to ensure that one can reason equivalently on polynomials or on polynomial functions, without running the risk of mistakes.

The *Lipton fingerprint* (in French: empreinte de Lipton) $Lip(\ell, r)$ of a list $\ell \stackrel{\text{def}}{=} [x_1, x_2, \ldots, x_k]$ as above, taken at r ($0 \leq r \leq p-1$), is the value of the polynomial $\prod_{i=1}^{k}(X - x_i)$ at $X \stackrel{\text{def}}{=} r$ mod p; in other words, $\prod_{i=1}^{k}(r - x_i)$ mod p.

We define the following *Lipton protocol.* Its purpose is to decide whether two lists $\ell \stackrel{\text{def}}{=} [x_1, x_2, \ldots, x_k]$ and $\ell' \stackrel{\text{def}}{=} [y_1, y_2, \ldots, y_{k'}]$ of natural numbers in $[0, p-1]$ (p prime, $k, k' < p$) are equal up to permutation, with high probability. The Lipton protocol draws a number r uniformly at random between 0 and $p-1$, and computes the fingerprints $Lip(\ell, r)$ and $Lip(\ell', r)$. We accept if the two fingerprints match (are equal), we reject otherwise.

Question 1 Show that the Lipton protocol does not make a mistake if ℓ and ℓ' are equal up to permutation; in other words, it accepts in this case.

Question 2 What is the probability that the Lipton protocol makes a mistake when ℓ and ℓ' are not equal up to permutation? I am expecting an easy upper bound, bounded by a polynomial in k and k' (when p is fixed) and tending to 0 as p tends to $+\infty$.

Question 3 Why does the Lipton protocol (a) only require space $O(\log p)$? (b) $O(\log p)$ random bits on average? (c) How much time does it take on average? You must state explicitly how you pick r at random, the space complexity of arithmetic operations, and the algorithm you use in order to evaluate the various Lipton fingerprints.

A *graph* $G \stackrel{\text{def}}{=} (V, E)$ is a pair of a finite set $V \stackrel{\text{def}}{=} \{1, 2, \cdots, N\}$ of so-called *vertices* (singular: vertex; in French: sommet) and a subset $E \subseteq V \times V$ or *edges* (French: arcs). Note that N is the number of vertices; all of them are represented as numbers in binary.

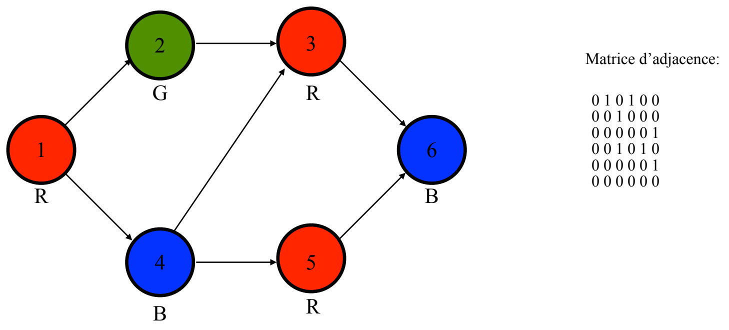
We represent G by its adjacency matrix. That is an $N \times N$ matrix of bits M, which we write as a sequence of N^2 bits. We will represent G as the word $0^N 1 M$, namely N zeros followed by a 1 (in order to specify N), followed by the N^2 entries of M. For any two vertices u and v (with $1 \leq u, v \leq N$), there is an edge $u \to v$ in G if and only if the bit at position $Nu + v$ in the word representation of the graph is equal to 1. Positions start at 0.

Let $Col \stackrel{\text{def}}{=} \{R, G, B\}$ be the set of so-called *colors.* A *3-coloring* of G is a map $col \colon V \to Col$ such that for every edge $u \to v$ in G, $col(u) \neq col(v)$. We will say that $col(u)$ is the *color* of u (with respect to col).

The *3-coloring problem* 3-COL is:
INPUT: a graph G;
QUESTION: does G have a 3-coloring?

This is an **NP**-complete problem under logspace reductions.

We will use a variant of the Lipton protocol in order to decide 3-COL between Arthur and Merlin, using as little resources (space, number of random bits) as we can. Let me use an example in order to explain the idea. Let us imagine that Arthur and Merlin wish to decide whether the following graph has a 3-coloring.



Of course, the answer is yes in this case, and I have taken the opportunity to show one possible 3-coloring.

We encode a 3-coloring candidate function *col* as a list of words $pos\#b\#c$ where:

- pos is a natural number in binary such that $N+1 \leq pos \leq N^2+N$; in other words, pos is a position in the M part of the input word representing G;
- $b \in \{0,1\}$; 0 means "source" and 1 means "target";
- $c \in Col$.

Such a word is meant to say that pos is the position of an edge $u \to v$ (namely, that $pos = Nu+v$ and that $u \to v$ really is an edge of G), and that c is the color, either of its source u (if $b=0$) or of its target v (if $b=1$). In the example such a list might be:

$$\begin{array}{lll} [& 8\#0\#R, 8\#1\#G, & \text{(edge } (1,2)) \\ & 10\#0\#R, 10\#1\#B, & \text{(edge } (1,4)) \\ & 15\#0\#G, 15\#1\#R, & \text{(edge } (2,3)) \\ & 24\#0\#R, 24\#1\#B, & \text{(edge } (3,6)) \\ & 27\#0\#B, 27\#1\#R, & \text{(edge } (4,3)) \\ & 29\#0\#B, 29\#1\#R, & \text{(edge } (4,5)) \\ & 36\#0\#R, 36\#1\#B \quad] & \text{(edge } (5,6)) \end{array} \tag{2}$$

(The comments between parentheses on the right are just comments, and are not part of the words, or of the whole list of words. Numbers are shown in decimal notation for readability.)

The protocol is in two phases. In each one, Merlin sends a list, which he claims are equal up to permutation. Each of these two lists is meant to be sorted, but according to different criteria.

In *phase 1*, Merlins sends a list ℓ_1 as above to Arthur, and claims that ℓ_1 is sorted by increasing vertex numbers. The *vertex number* of $pos\#b\#c$, where $pos = Nu + v$, is u if $b = 0$, v if $b = 1$. Merlin also claims that the colors are *consistent*, namely that any two words with the same vertex number are given the same color. In the example, an honest Merlin would play:

$$\begin{array}{ll} [\ 8\#0\#R, 10\#0\#R, & \text{(vertex 1)} \\ \ \ 8\#1\#G, 15\#0\#G, & \text{(vertex 2)} \\ \ \ 15\#1\#R, 27\#1\#R, 24\#0\#R, & \text{(vertex 3)} \\ \ \ 10\#1\#B, 27\#0\#B, 29\#0\#B, & \text{(vertex 4)} \\ \ \ 29\#1\#R, 36\#0\#R, & \text{(vertex 5)} \\ \ \ 24\#1\#B, 36\#1\#B\ \ ] & \text{(vertex 6)} \end{array}$$

In *phase 2*, Merlin sends a similar list ℓ_2. He claims that ℓ_1 and ℓ_2 are equal up to permutation, but that ℓ_2 is sorted by *increasing positions k* on the input tape, that all the edges are listed exactly twice, once for each end of each edge, and that the colors of those two ends are distinct. In the example, an honest Merlin would play the list (2) mentioned earlier.

Question 4 Using this idea, show that 3-COL is in the class $\mathbf{IP}_1$ of interactive proofs such that:

(a) the playing order is **MAM** (Merlin plays, then Arthur, then Merlin; then Arthur decides);

(b) Merlin's answers have polynomial size, and Arthur can only read them from left to right when his turn comes, and cannot go back;

(c) Arthur does not ask *any* question when his turn comes (and therefore does not reveal any random bit, although it will have to draw some bits at random);

(d) Arthur works in logarithmic space (not counting the space used by the communication tape between Arthur and Merlin, or by the input tape) and polynomial time;

(e) Arthur only draws a logarithmic amount of random bits.

We agree that the final verification phase is done by Arthur, so all resource constraints that apply to Arthur are also relevant for the final verification phase. You will use the notations G, V, E, n, N, M, etc., given above. Let me remind you that "logarithmic" means $O(\log n)$, where n is the size of the input. Also, as usual, the communication tape between Arthur and Merlin is reused, hence erased at each turn.

The acceptance conditions are:

(f) If G has a 3-coloring, then Merlin has a way of making the protocol accept.

(g) Otherwise, whatever Merlin's strategy is, the protocol will accept with probability at most 1/4.

You will pay special care to the details. For instance, time must be polynomial in the *worst* case, not on average. Also, you can test whether a given number p is prime in various ways, see Appendix A, but not all of them may fit our purposes here. Finally, say explicitly how all of Merlin claims are to be verified, and explain why all the various constraints on resources are met, how many registers you need, holding numbers in binary or in unary, and so on.

Question 5 Modify the previous interactive proof slightly, and show that 3-COL is also in the class $\mathbf{IP}_2$ of interactive proofs where:

(a) the playing order is: Arthur plays once, then Merlin plays polynomially many times;

(b) each of Merlin's answers is exactly 1 bit long,

and conditions (c)–(g) are unchanged.

Question 6 Deduce that $\mathbf{NP} \subseteq \mathbf{IP}_1$ and that $\mathbf{NP} \subseteq \mathbf{IP}_2$.

2 Strategies

A *strategy* is just what we called a Merlin map in the lectures. It maps (public) histories $x\#q_1\#y_1\#q_2\#y_2\# \cdots \#q_i$ to an answer y_i by Merlin. We require all questions q_i and all answers y_i to be of size bounded by a fixed polynomial in the size n of x.

IP(logspace, lograndbits) is the class of languages decided by interactive proofs where Arthur uses only $O(\log n)$ space and $O(\log n)$ random bits. The acceptance conditions are: if $x \in L$, then the interactive proof will accept with probability at least 3/4 (if Merlin plays honestly); if $x \notin L$, then it will accept with probabliity at most 1/4, whichever strategy Merlin uses.

Question 7 We are given an **IP** protocol between Arthur and Merlin, with polynomially many rounds, say $p(n)$. Show that, if Arthur uses only $O(\log n)$ random bits in that **IP** protocol, then there is a concise way of representing any of Merlin's strategies M on input x. By *concise*, we mean of polynomial size. By a representation, we mean a data structure $S(x)$, depending on the input x, and a polynomial-time computable function f such that $f(S(x), x\#q_1\#y_1\#q_2\#y_2\# \cdots \#q_i) = M(x\#q_1\#y_1\#q_2\#y_2\# \cdots \#q_i)$ for all possible public histories $x\#q_1\#y_1\#q_2\#y_2\#\cdots\#q_i$ that may result from the interaction between Arthur and Merlin on input x.

Question 8 Show that any language L in $\mathbf{IP}(\mathrm{logspace}, \mathrm{lograndbits})$ has an $\mathbf{MA}$ protocol (an Arthur-Merlin protocol where Merlin plays first, Arthur plays second and there is no other round), with resource constraints as in the lectures, except that Arthur uses only logarithmically many random bits; and where the error probability is at most 1/4.

Question 9 Show that $\mathbf{IP}(\mathrm{logspace}, \mathrm{lograndbits}) \subseteq \mathbf{NP}$.

Question 10 Conclude that all the classes $\mathbf{IP}(\mathrm{logspace}, \mathrm{lograndbits})$, $\mathbf{NP}$, $\mathbf{IP}_1$ and $\mathbf{IP}_2$ are equal.

A Primality tests

Let p be a positive natural number. Its size is $|p| = \lceil \log_2 p \rceil + 1$.

1. One can test whether p is prime in deterministic polynomial time by an algorithm due to Kayal, Agrawal, and Saxena. The space used is polynomial.

2. Primality is in **coNP**. One can check that p is prime by verifying that for every x with $2 \leq x \leq p-1$, x does not divide p.

3. Primality is in **NP**, by a result of Pratt, using a recursive version of Lucas' primality test. The idea is that a *proof of primality* of p is a tree whose root is labeled by p and by a natural number x in $[1, p-1]$, and whose immediate subtrees are proofs of primality of numbers $p_1, \ldots, p_k$ in $[1, p-2]$. In order to verify such a proof, one checks that: either $p = 2$ and $k = 0$; or $p \geq 3$ is odd, x is between 1 and $p-1$, the immediate subtrees are valid proofs of primality of $p_1, \ldots, p_k$, $\prod_{i=1}^{k} p_i = p - 1$, x is prime with p, $x^{p-1} = 1 \bmod p$, and $x^{(p-1)/p_i} \neq 1 \bmod p$ for every i with $1 \leq i \leq k$.

4. The Rabin-Miller test. We assume p odd: the even case is trivial. Then one can write p as $1 + 2^k q$ where q is odd. We draw x uniformly at random between 1 and $p-1$, we verify that $x^q = 1 \bmod p$ or that $x^{2^j q} = -1 \bmod p$ fr some j with $1 \leq j < k$. If p is prime, the test always succeeds. Otherwise, it fails with probabilty at least 3/4.

5. The Solovay-Strassen test. We again assume p odd. We draw x at random as above, and we check that $x^{(p-1)/2} = \left(\frac{x}{p}\right) \bmod p$, where the Jacobi symbol $\left(\frac{x}{p}\right)$ is computed recursively by the following formulae (where n is odd):

   - $\left(\frac{m}{n}\right) = \left(\frac{m \bmod n}{n}\right)$, if $m \geq n$;

- $\left(\frac{1}{n}\right) = 1$;
- $\left(\frac{2m}{n}\right) = \left(\frac{m}{n}\right)$ if n is equal to 1 or 7 mod 8, $\left(\frac{2m}{n}\right) = -\left(\frac{m}{n}\right)$ if n is equal to 3 or 5 mod 8;
- when m is odd and $m < n$, $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)$ if n or m is equal to 1 mod 4, $\left(\frac{m}{n}\right) = -\left(\frac{n}{m}\right)$ if $n = m = 3$ mod 4.

If p is prime, the test always succeeds. Otherwise, it fails with probabilty at least 1/2.