Complexité avancée - TD 8

Benjamin Bordais

November 24, 2021

In all the exercises, write carefully the bound on the probabilities involved in all algorithms.

**Exercise 1** (Matrix multiplication). Consider the following decision problem: given three \( n \times n \) squared matrices \( A, B, C \), decide if \( A \times B = C \). The best known bound for deterministic algorithm deciding this problem is \( O(n^{2.3728576}) \) (by using what is called a ‘galactic’ algorithm, can you guess why?). Exhibit a probabilistic algorithm running in time \( O(n^2) \) deciding this problem (with a \( \text{coRP} \)-like semantic).

**Solution 1.** Given two matrices \( A \) and \( B \), we chose randomly a vector \( x \in \{0, 1\}^n \) and then compute in time \( O(n^2) \) the product \( A \times (B \times x) \). We then compare it with \( C \times x \) (also computed in time \( O(n^2) \)) and accept if and only if \( (A \times B) \times x = C \times x \). Straightforwardly, we always accept when \( A \times B = C \). Assume now that it is not the case. That is, there is at least one vector \( (A \times B)_i \neq C_i \). Let \( d := (A \times B)_i \) and \( e := C_i \). Let us show that the probability (over random vectors chosen uniformly in \([0, 1]^n\)) that \( d \times x \neq e \times x \) is at least \( \frac{1}{2} \).

Let \( E_- := \{x \in \{0, 1\}^n \mid d \times x = e \times x\} \) and \( E_+ := \{0, 1\}^n \setminus E_- \). Let also \( j \) be the smallest index such that \( d_j \neq e_j \). We define the injective function \( f : \{0, 1\}^n \to \{0, 1\}^n \) such that, for all \( x \in V \), we have \( f(x)_k = x_k \) for all \( k \neq j \) and \( f(x)_j = 1 - x_j \). Then, let \( x \in E_- \). We have \( d \times f(x) - e \times f(x) = d \times x - e \times x + (d_j - e_j) \cdot (1 - 2 \cdot x_j) = (d_j - e_j) \cdot (1 - 2 \cdot x_j) \neq 0 \). That is, \( f(x) \in E_+ \). In fact, \( f(E_-) \subseteq E_+ \). By injectivity of \( f \), we have \( |E_-| \leq |E_+| \), which proves the result. Overall, the probability to accept when \( A \times B \neq C \) is at most \( \frac{1}{2} \).

We recall the definition of BPP. A language \( L \) is in BPP if there exists a Turing machine \( \mathcal{M} \) running in polynomial time \( p(n) \) on all input \( x \) such that \( |x| = n \) and random tape \( r \) of size \( p(n) \) such that:

- If \( x \in L \), then \( Pr_r[\mathcal{M}(x, r) = \top] \geq 2/3 \);
- If \( x \notin L \), then \( Pr_r[\mathcal{M}(x, r) = \top] \leq 1/3 \).

We also recall the Chernoff’s bound: Let \( X_1, \ldots, X_N \) be random independent variables with value in \( \{0, 1\} \) with the same law \( Pr(X_i = 1) = p \), then:

\[
Pr(X_1 + X_2 + \ldots + X_N \geq (1 + \theta) \cdot p \cdot N) \leq e^{-\frac{\theta^2 \cdot p \cdot N}{2}}
\]

**Exercise 2** (BPP and expected running time). What complexity class do we get if, in the definition of BPP, we consider Turing machine that may not terminate but whose expected running time is polynomial?
Solution 2. We still have BPP. Consider a Turing machine $M$ with expected running time polynomial for a language $L$. By definition, there exists $c$ s.t. $E[T_M(x,r)] < |x|^c$. For $K$ a polynom, we define $M_K$ a TM which executes $M$ on $x$ and rejects if the number of steps taken exceeds $K(|x|)$. Then:

- If $x \notin L$, $Pr[M_K(x,r) = \top] \leq Pr[M(x,r) = \top] \leq 1/3$
- If $x \in L$, $Pr[M_K(x,r) = \bot] \leq Pr[M(x,r) = \bot] + Pr[T_M(x,r) \geq K(|x|)]$

By Markov’s inequality, $Pr[T_M(x,r) \geq K(|x|)] \leq \frac{E[T_M(x,r)]}{K(|x|)} = \frac{|x|^c}{K(|x|)}$.

If we set $K(n) = 12 \cdot n^c$ (for instance), we have $Pr[M_K(x,r) = \top] = 1 - Pr[M_K(x,r) = \bot] \geq 1 - (\frac{1}{3} + \frac{1}{12}) \geq \frac{7}{12}$. Hence, in both cases, the probability of error is lower than or equal to 5/12. That is, $L \in \text{BPP}(5/12) = \text{BPP}$.

Exercise 3 (BPP and PSPACE).

- Argue that BPP(1/2) = \{ all languages \} and BPP = coBPP.

- Give a direct proof that BPP \subseteq PSPACE.

Solution 3.

- For an arbitrary language $L$, we can consider the randomized Turing machine that accepts with probability 1/2 regardless of that input. Furthermore, from a probabilistic Turing machine such that $L \in \text{BPP}$, we can swap the accept and reject so that we also have $L$ in BPP.

- Consider $M$ a PTM for a language $L$ in BPP. By definition, we have $c \in \mathbb{N}$, such that $T_M(x,r) \leq |x|^c$, for all $r$ of length lower than $|x|^c$. Let $x$ be a word and $n = |x|$. There are $\max(x) = |\Sigma|^n$ different $r$ to test if $r$ is written on the finite alphabet $\Sigma$. We use the following pseudocode:

\[
\text{Simulation}(x): \\
\quad \text{let } nacc = 0 \\
\quad \text{let } nrej = 0 \\
\quad \text{for } r = 0 \text{ to } \max(x) - 1 \text{ do} \\
\quad \quad \text{res} = \text{Execute } M(x,r) \\
\quad \quad \text{if (res) then } nacc ++ \text{ else } nrej ++ \\
\quad \text{end if} \\
\quad \text{return (nacc > nrej)}
\]

The values $r,nacc$ and $nrej$ have a (bit) length lower than $n^c$. Moreover, by definition, executing $M(x,r)$ takes polynomial time, so a fortiori, also polynomial space. It follows that $L \in \text{PSPACE}$.

Exercise 4 (BPP-completeness).

1. Let $L = \{(M,x,1^t)\mid M \text{ accepts } x \text{ in time at most } t\}$, where $M$ is the code of a non-deterministic Turing machine, $x$ an input of $M$ and $t$ a natural number. Show that this language is NP-complete.

2. Let now $L$ be the language of words $(M,x,1^t)$ where $M$ designates the encoding of a probabilistic Turing machine and $x$ a string on $M$'s alphabet such that $M$ accepts $x$ in at most $t$ steps, for at least $2/3$ of the possible random tapes of size $t$.

Is $L$ BPP-hard? Show that if it is in BPP, it would imply NP \subseteq BPP.
3. A PTM $M$ has error probability $\leq 1/3$ if, for all input $x$, we have either $\Pr(M \text{ accepts } x) \leq 1/3$ or $\Pr(M \text{ accepts } x) \geq 1/3$. Show that deciding if polynomial time PTM $M$ has error probability $\leq 1/3$ is undecidable.

Solution 4.  
1. • $L \in \text{NP}$. Let $M$ be the code of a non-deterministic Turing machine, $x$ an input of $M$ and $t$ a natural number. Notice that the timeout $t$ we set for the execution of $M(x)$ is lower than the length of $(M, x, 1^t)$.

So the algorithm which simulates $M$ on $x$ in the input $(M, x, 1^t)$ is non-deterministic and runs in polynomial time. Then we can check that $(M, x, 1^t) \in \{(M, x, 1^t) \mid M \text{ accepts on input } x \text{ in time at most } t\}$. Therefore, $L \in \text{NP}$.

• $L$ is NP-hard. Given $L' \in \text{NP}$, $M$ a NDTM for $L'$, and $p$ a polynomial associated. For an instance $x$ of $L'$ we can build (in logspace) the instance $(M, x, 1^{p(|x|)})$. Then, by definition of $L$, $(M, x, 1^{p(|x|)}) \in L \Leftrightarrow x \in L'$.

2. • $L$ is BPP-hard: (for exactly the same reasons). Given $L' \in \text{BPP}$, $M$ a probabilistic Turing machine for $L'$, and $p$ his polynomial associated. For an instance $x$ of $L'$ we can build (in logspace) the instance $(M, x, 1^{p(|x|)})$. And, by definition of $L$, $(M, x, 1^{p(|x|)}) \in L \Leftrightarrow x \in L'$.

• Consider a propositional formula $\varphi$ with $n$ variables. Consider a machine $M$ that accepts with probability $2/3 - 1/2^{n+2}$ or tests a random valuation and accepts iff the valuation satisfies the formula. Then, the machine $M$ on the input $\varphi$ accepts with probability $\geq 2/3$ iff the formula is satisfiable.

3. Let $L := \{(M, x) \mid M \text{ accepts } x\}$ be an undecidable problem. We reduce it to the problem we are considering. Given a Turing machine $M$ and an input $x$, we construct the following PTM $M'$: on an input $y$, choose a random bit $b$. If $b = 1$ reject, otherwise simulate $M$ on $x$ for $|y|$ steps and accept if and only if $M$ accept within that timeout. By definition, we have $M'$ with error probability $\leq 1/3$ if and only if $M$ does not accept $x$.

Exercise 5 (Deciding undecidable problems). Exhibit a real number $\rho$ such if a polynomial time algorithm access to random tape whose bit have probability $\rho$ to be equal to 1, it can decide an undecidable language (with a BPP-like semantic).

Solution 5.  
1 Let $f : \mathbb{N} \rightarrow 0, 1^*$ such that $f(n)$ is the binary writing of $n$ (without the initial 1). Consider an undecidable language $H$, for instance:

$$H = \{ x \in \{0, 1\}^* \mid x \text{ is the binary encoding a Turing machine not halting on the empty word}\}$$

Now, let $L = \{1^{p(k)} \mid f(k) \in H\}$ be another undecidable language for some computable function $q : \mathbb{N} \rightarrow \mathbb{N}$ that we will define later. Let us now define $p \in \{0, 1\}$ as $p = 0, p_1 p_2 \ldots$ such that, for all $i \geq 1$: $(p_{2i-1}, p_{2i}) = (1, 0)$ if $f(i) \in H$ and $(0, 1)$ otherwise.

We consider now a Turing machine $M$ that, on an input $x \in \{0, 1\}^*$, does the following:

1. Check that $x$ is of the form $1^{q(k)}$ and compute $k$ (otherwise reject)

2. Pick $N$ randoms bits (we will see later what to choose for $N$) and count the number $n$ of 1

3. Compute the $2k - 1$-th bit of the fraction $n/N$ of 1s and accept iff it is 1.

\[https://cstheory.stackexchange.com/questions/32947\]
The running time is in $O(N)$ and we have to choose $N$ and $q$ such that the running time is polynomial in $|x|$. Then, note that $p_{2k+1} = 1$ iff $1^{q(k)} \in L$. Let us show that for well-chosen $N, q$ the $(2k - 1)$-th bit of $n/N$ is equal to $p_{2k-1}$ with probability greater than or equal to $2/3$. First, note that since we have $p_{2k+1} \neq p_{2k+2}$, if we have $|n/N - p| < 2^{-2k-2}$ then for all $i \leq k$, we have the $i$-th bit of $n/N$ equal to $p_i$. In particular, the output of the algorithm is correct in that case. With the Chernoff’s bound, we have:

$$Pr(n \geq (1 + \theta) \cdot p \cdot N) \leq \exp^{-\frac{\theta^2 \cdot p \cdot N}{4}}$$

This also holds for $Pr(n \leq (1 - \theta) \cdot p \cdot N)$ since each random bits follows a Bernouilli law. Overall, we obtain:

$$Pr(|n/N - p| \geq \theta \cdot p) \leq 2 \exp^{-\frac{\theta^2 \cdot p \cdot N}{4}}$$

By setting $\theta := 1/(2^{2k+2} \cdot p)$, we obtain:

$$Pr(|n/N - p| \geq \frac{1}{2^{2k+2}}) \leq 2 \exp^{-\frac{N}{3 \cdot p \cdot 4^{2k+2}}}$$

Then, if we set $N = 3 \cdot 4^{2k+2} \cdot \ln(6)$, we obtain:

$$Pr(|n/N - p| \geq \frac{1}{2^{2k+2}}) \leq 2 \exp^{-\ln(6)/p} < 2 \exp^{-\ln(6)/p} = 1/3$$

Hence, $Pr(|n/N - p| < \frac{1}{2^{2k+2}}) > 2/3$ and the algorithm is correct with probability at least $2/3$. By setting $q(k)$ to $4^k$, we obtain that $N(k) = O(q(k)^2)$. 

4