Homework for Computational Complexity | Created: 11/06/2020
Spring 2020, UPC Barcelona | Last modified: 11/06/2020
Instructor: Albert Atserias

Homework 2 – Solutions

Exercise 1 - Solution: Here is the algorithm. Let G have vertices $v_1, \dots, v_n$. The algorithm starts by testing whether G is in k-COL. If it is not, then we say so and stop. If it is, then we build a valid k-coloring for G. The idea is to iteratively extend a partial valid assignment of colors $c_1, \dots, c_\ell \in [k]$ for the vertices $v_1, \dots, v_\ell$, for $\ell = 1, 2, \dots, n$, by forcing v_ℓ to take color c_ℓ through a gadget, and testing every possible color for $v_{\ell+1}$, also through a gadget. Precisely, given an assignment $c_1, \dots, c_\ell \in [k]$ of colors for the vertices $v_1, \dots, v_\ell$, let $G_k(c_1, \dots, c_\ell)$ be the graph obtained from G by adding a k-clique on k many new vertices $u_1, \dots, u_k$, and adding the edges (v_i, u_j) for each $i = 1, \dots, \ell$ and $j \in [k] \setminus \{c_i\}$. This graph is k-colorable if and only if G is k-colorable by a coloring that extends the assignment $v_i \mapsto c_i$ for $i = 1, \dots, \ell$. Thus, we can let our algorithm run through $c = 1, \dots, k$ and query whether $G_k(c_1, \dots, c_\ell, c)$ is in k-COL until we find a valid coloring extension for the vertices $v_1, \dots, v_{\ell+1}$. By the time $\ell = n$ we will have produced a valid k-coloring. The running time of this algorithm is $k \cdot n \cdot T(n + k)$, where $T(m)$ is the bound on the running time for k-COL on graphs with m vertices. If k-COL is solvable in polynomial time, then this running time is polynomial in n, and we are done.

Exercise 2 - Widthdrawn. Explanation: What I had in mind asking was to show that if k-COL is in BPP then it is also in RP. But I asked for ZPP because I wrongly thought "since BPP = co-BPP, it just follows". This was wrong; I'm sorry for the confusion this caused (brr... and it's not the first time I make this mistake...).

Exercise 3 - Solution: It is well known (and easy to show) that a graph is 2-colorable if and only if it is bipartite, which happens if and only if it doesn't contain any cycle of odd length. Equivalently, a graph is in the complement NON-2-COL of 2-COL if and only if it contains a closed walk of odd length. We can test the latter in non-deterministic logarithmic space by guessing an initial vertex in the walk, which we keep in memory, and then guessing each successive vertex in the walk, forgetting the earlier ones, until we return to the initial vertex. Along the way we can keep an alternating bit that indicates the parity of the length of the path visited so far. These are $2\log(n) + 1$ bits of memory, where n is the number of vertices in the graph. If by the time we return to the origin the bit indicates that the length is odd, then the graph is not 2-colorable. And if the graph is indeed not 2-colorable, then some execution of this algorithm will detect so. This shows that NON-2-COL is in NL, from which it follows that 2-COL is in NL since NL = co-NL by the Immerman-Szelepszcenyi Theorem.