Assignment 8

Aleksandr Salo

Due November 25, 2014

1 NP-completeness proofs

1. (10 points) Consider the following language:
QUAD3SAT = $\{< \phi > | \phi$ is a 3-CNF formula having at least 4 different satisfying assignments$\}$
For example, the following string is in QUAD3SAT: $(x \vee x \vee \neg x) \wedge (y \vee y \vee \neg y)$ because there are four different assignments that satisfy it (00, 01, 10, and 11). Prove that QUAD3SAT is NP-complete.

1. In order to proof QUAD3SAT is NP-complete we have to prove that it is in NP and that some known NP-complete language is poly-time reducible to it.

2. Clearly, QUAD3SAT $\subset$ 3SAT, thus it is in NP. Direct proof would contain a certificate, which is simply 4 different satisfying assignments. Clearly, a DTM would check this certificate in poly-time and it'd take exactly 4 times longer than verifying 3SAT.

3. Let us now reduce 3SAT, which is known to be NP-COMPLETE, to QUAD3SAT in poly-tyme.

4. Let F be a DTM, where:
   F = "on input $< \phi >$
   1. Construct $\psi = \phi \wedge (x_1 \vee x_1 \vee \neg x_1) \wedge (x_2 \vee x_2 \vee \neg x_2)$, where x_1, x_2 are new variables not in ϕ.
   2. Print $< \psi >$."

5. This reduction clearly works in poly time as we just added one simple clause.

6. Note, that original 3cnf formula $\phi \in 3SAT$ iff new formula $\psi \in QUAD3SAT$. If ϕ has at least one satisfying assignment then ψ must have at least 4 satisfying assignments, because, clearly, $(x_1 \vee x_1 \vee \neg x_1) \wedge (x_2 \vee x_2 \vee \neg x_2)$ has 4 different satisfying assignments itself (11, 10, 01, 00). On the other hand, if ϕ doesn't have a satisfying assignment, then ψ would be "ruined" because of necessary conjunction with ϕ.

7. That proves that QUAD3SAT is NP-COMPLETE.

2. (15 points) Consider the following language:
LCS = $\{< G1, G2, k > \mid$ G1 and G2 are graphs that have isomorphic subgraphs with k edges each$\}$
Recall the following definitions:
- A subgraph G' = (V', E') of a graph G = (V, E) has the following properties: V' $\subseteq$ V and E' $\subseteq$ E.
- Two graphs G = (V, E) and H = (V', E') are isomorphic if $|V| = |V'|$ and $|E| = |E'|$ and there is a function $f : V \to V'$ such that $(u, v) \in E \Leftrightarrow (f(u), f(v)) \in E'$.

1. In order to proof LCS is NP-complete we have to prove that it is in NP and that some known NP-complete language is poly-time reducible to it.

2. Firstly, LCS is in NP and the certificate is a mapping ϕ from subset of nodes in G1 to the subset of nodes in G2. The verifier then checks whether for each edge $e = (u, v)$ in G1 the edge $(\phi(u), \phi(v))$ is also in G2, and if $e = (u, v)$ is NOT in G1 the edge $(\phi(u), \phi(v))$ is also NOT in G2. That is easy to implement in any reasonable programming language with two simple nested loops and thus would take at most O(N^2) time, which is poly time.

3. Let us now reduce CLIQUE, which is known to be NP-COMPLETE, to LCS in poly-tyme.
Note, that in general, it seems that CLIQUE must be a special case of LCS, where one of the graphs is the k-clique. While CLIQUE is NP-C, LCS must surely be NP-C as well as more general problem.

4. Let F be a DTM, where:
F = "on input $< G, k >$
1. Define $G_1 = G$.
2. Define G_2 to be the complete graph on k verticies.
3. Print $< G_1, G_2, k >$".

5. Clearly, this reduction runs in poly time because all it does is copying one graph and defining another of size k, where k $\leq |V| \in G$.

6. Showing that reduction actually works is more subtle task. We want to convert the question of one language to the question of another. That is $< G, k > \in CLIQUE >$ iff $< G_1, G_2, k > \in LCS$. By our design, G_2 will always be a k-clique. Any other k-clique will be isomorphic to it. Moreover, to be isomorphic to G_2 a graph must be k-clique. Thus G_1 must have a k-clique in order to contain a subgraph that is isomorphic to a subgraph (graph itself) in G_2. While G_1 is a copy of G, if G_1 has a k-clique so does G, which answer the question of CLIQUE.
On the other hand, if G_1 doesn't have an isomorphic subgraph with G_2, then G_1 doesn't have a k-clique and hence G doesn't have a k-clique. Thereby the reduction works as desired.

7. That proves that LCS is NP-COMPLETE.