Assignment 9

Aleksandr Salo

Due December 11, 2014

1 Proofs

1. (10 points) Consider the following language:
IST = $\{< G, T > | G$ is a graph with a spanning tree isomorphic to T $\}$

1. In order to proof IST 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. First, IST is clearly in NP since we can use DTM verifier with spanning tree T' description as a certificate. DTM would check if there is a bijection between vertex sets of T and T'. To be more general, spanning tree isomorphism problem is just a graph isomorphism problem and hence it is in NP.
3. Second, note that it seems like HAMPATH, which is known to be NP-COMPLETE, is a special case for a broader IST problem. For example, if a given T has a max degree of its vertices equals two and T is a path then finding a T in G is HAMPATH problem, which is NP-COMPLETE.
   Let us now reduce from HAMPATH to IST in poly-tyme.
4. Let F be a DTM, where:
   F = "on input $< G >$
   1. Construct the path (spanning tree) T by applying breadth first search algorithm to s:
   1.1 add s to T
   1.2 go through each of the neighbors: if neighbor is not in T - stop cycle and do bfs(neigbor)
   2. Check the sizes: if $|G| > |T|$, REJECT 3. Output $< G, T >$.
5. This reduction clearly works in poly time since bfs is in NP and we don't even modify G.
6. Note, that we constructed the spanning tree T in such a way that its vertices max degree is two. Thus if G has a Hamiltonian path then IST accepts $< G, T >$. Conversely if IST accepts $< G, T >$ then there exist a Hamiltonian path of length n in G that means HAMPATH would accept $< G >$. If there is no Hamiltonian path - IST rejects $< G, T >$ by design.
7. That proves that IST is NP-COMPLETE.

2. (5 points) Problem 8.11 from Sipser:
Show that if every NP-hard language is also PSPACE-hard, then PSPACE = NP.

1. Assume, for the sake of showing the inevitable result, that every NP-hard language is also PSPACE-hard.
2. We know that $NP \subseteq PSPACE$ *(Sipser, p.336)*
3. Consider language SAT which is in $PSPACE$ *(Sipser, p.332)*
4. We know also that $SAT \in NP - COMPLETE$ and consequently, by definition of $NP - C$, SAT is NP-hard.
5. Thus, by our assumption, SAT should be also PSPACE-hard.
6. By definition of PSPACE-hard: $\forall L \in PSPACE : L \leq_p SAT$
7. Since $SAT \in NP$ we derive that $\forall L \in PSPACE : L \in NP$. Therefore $PSPACE \subseteq NP$.
8. We already know the opposite inclusion $NP \subseteq PSPACE$ hence $PSPACE = NP$.
9. Thus assumption "every NP-hard language is also PSPACE-hard" let us derive that PSPACE = NP.

3. (10 points) Problem 10.20 from Sipser:
Define a ZPP-machine to be a probabilistic Turing machine that is permitted three types of output on each of its branches: accept, reject, and ?. A ZPP-machine M decides a language A if M outputs the correct answer on every input string w (accept if w $\in$ A and reject if w $\notin$ A) with probability at least $\frac{2}{3}$, and M never outputs the wrong answer. On every input, M may output ? with probability at most $\frac{1}{3}$. Furthermore, the average running time over all branches of M on w must be bounded by a polynomial in the length of w. Show that $RP \cap coRP = ZPP$, where ZPP is the collection of languages that are recognized by ZPP-machines.

1. First, we show that $RP \cap coRP \subseteq ZPP$
   Let $L \in RP \cap coRP$ then $\exists$ poly-time probabilistic TMs M_1 and M_2 that decide L to be in RP and coRP respectively with the following properties:

   (a) if $w \in L, P(M_1(w,r) \text{ accepts}) \geq 1/2$ and

   (b) if $w \notin L, P(M_1(w,r) \text{ accept}) = 0$

   (a) if $w \notin L, P(M_2(w,r) \text{ accepts})1$ and

   (b) if $w \in L, P(M_2(w,r) \text{ accept}) \leq 1/2$

   That is M_1 never wrong about its "YES" answers, and M_2 is never wrong about its "NO" answers.
   Let us now construct another poly-time probabilistic TM N with $L(N) \in ZPP$:
   N = "on input $< w, r >$,
   1. repeat 2 times:
   1.1. run M_1 on $< w, r >$ and if it accepts, ACCEPT
   1.2. run M_2 on $< w, r >$ and if it rejects, REJECT
   2. HALT in "I don't know" state."
   Note the following about TM N:

   (a) N runs in poly-time since it simulates two polytime machines at most two times each.

   (b) N is never wrong since it only accepts if M_1 accepts and rejects if only M_2 rejects.

   (c) N would print "I don't know" with $p \leq 1/3$

   That proves that $RP \cap coRP \subseteq ZPP$

2. Second, we show that $ZPP \subseteq RP \cap coRP$.
   Suppose we have a TM N that implements a ZPP algorithm. It may print "I don't know" with chance less or equal 1/3. Let us construct a TM M that uses ZPP algorithm to implement RP algorithm:
   M = "on input $< w, r >$,
   1. Run N on $< w, r >$ and if N accepts or rejects - do the same.
   2. REJECT (in case N prints "I don't know")
   Note the following about TM M:

   (a) M runs in poly-time since it simulates a polytime machine.

   (b) M is never wrong with its "YES" answer.

   (c) M is wrong with its "NO" answer with $p \leq 1/2$

   The same construction could be used to make algorithm for coRP. In that case we should ACCEPT in case N prints "I don't know"
   That proves that $RP \cap coRP \subseteq ZPP$

3. Because we proved those two inclusions we proved also that $RP \cap coRP = ZPP$