CS 7910 Computational Complexity
Final Exam

Gopal Menon

May 4, 2016

1. **(10 points)** What is NP? What is $NP-Complete$?

   NP stands for Non-Deterministic Polynomial. NP is the set of all problems for which algorithms run in polynomial time on a non-deterministic machine. This machine can be thought of one that can take all possible paths in parallel. On a real machine with finite resources, all NP algorithms will not run in polynomial time (unless if $P = NP$, which is highly unlikely). The set of problems for which algorithms run in polynomial time is called P and $P \subseteq NP$. Another property of NP algorithms is that even though they cannot all run in polynomial time, given a certificate that is a solution to an NP problem, it can be verified in polynomial time that the certificate is correct.

   $NP-Complete$ is the set of the hardest problems in NP. All NP problems can be reduced to $NP-Complete$ problems in polynomial time. Actually if an NP problem can be reduced to another problem, then the resulting problem is said to be in the set $NP-Hard$. The intersection of the sets NP and $NP-Hard$ is the set $NP-Complete$.

2. **(15 points)** For each of the following statements, please tell whether it is true and briefly explain your answer.

   a) There is no polynomial-time algorithm that can solve the vertex cover problem.

      This is true. The Vertex Cover problem is in $NP-Complete$. This means that any problem in NP can be reduced to a Vertex Cover problem in polynomial time. As a result the Vertex Cover problem is among the hardest problems in NP. If there exists a polynomial-time algorithm that can solve the Vertex Cover problem, then any problem in NP (including $NP-Complete$ problems) can be solved in polynomial time by first reducing it to the Vertex Cover problem and then solving the Vertex Cover problem in polynomial time. This would mean that $P = NP$. Since this is very unlikely, then it is safe to say that there is no polynomial-time algorithm that can solve the Vertex Cover problem.

   b) If there exists a polynomial-time algorithm that can solve the vertex cover problem, then there exists a polynomial-time algorithm that can solve the load balancing problem.

This is true. Both the Vertex Cover and Load Balancing problem (where jobs need to be efficiently assigned to machines) are in $NP-Complete$. This means that the Load Balancing problem can be reduced to the Vertex Cover problem in polynomial time. If there exists a polynomial-time algorithm that can solve the Vertex Cover problem, then the Load Balancing problem can be solved in polynomial time by first reducing it to an instance of the Vertex Cover problem and then solving the Vertex Cover problem in polynomial time.

c) If there does not exist a polynomial-time algorithm that can solve the vertex cover problem, then there does not exist a polynomial-time algorithm that can solve the load balancing problem.

This is true. The Vertex Cover problem can be reduced to the Load Balancing problem in polynomial time since they are both in $NP-Complete$. This means that the Load Balancing problem is at least as hard as the Vertex Cover problem. If there does not exist a polynomial-time algorithm that can solve the Vertex Cover problem, then a polynomial time algorithm to solve the Load Balancing problem cannot exist since it is at least as hard as the Vertex Cover problem.

d) Quasi-polynomial, weakly-polynomial, and pseudo-polynomial are all polynomial.

This is false.

i. Quasi-polynomial (QP) run time is greater than polynomial time and less than exponential time - $n^d < QP < 2^n$, for a constant $d \geq 0$, where n is the input size. So this is not polynomial time.

ii. Weakly-polynomial running time is a function of the number of input bits. For example $O(n \cdot logM)$. Here M is the input value and $logM$ is the number of input bits. So this is not polynomial time.

iii. Pseudo-polynomial time is a polynomial function of the input value (as opposed to input size). An example is the Dynamic Programming solution for a Subset Sum problem which is $O(n \cdot M)$, where n is the input size and M is the input value for the target sum. This can be shown to be of the order of $O(n \cdot 2^m)$, where $m = logM$. So if we need to find the subset sum of 100 bit integers, we need to have a table if size $n \cdot 2^{100}$, in order to find a subset sum for a target value that uses the 100 bits. So this is actually exponential in nature.

iv. Polynomial time, or strongly polynomial time is where the run time is a function of the input size - $O(n^d)$, where $d \geq 0$.

e) Suppose A is an approximation algorithm for the vertex cover problem and the approximation ratio of A is 2; then it is also correct to say that the approximation ratio of A is 3.

Since A is an approximation algorithm for the Vertex Cover problem, which is a minimization problem, then $\frac{C}{OPT} \leq 2$, where C is the cost of the approximation algorithm and OPT is the cost of the optimal solution. If $\frac{C}{OPT} \leq 2$, then it is also true that $\frac{C}{OPT} \leq 3$. So it is also correct to say that the approximation ratio of A is 3.