Readings: Sections 7.1, 7.2, 7.3, 7.4, 7.5

Problem 1: Let’s review the following new terms and concepts.

1. Time complexity
2. Asymptotic, worst-case analysis
3. Polynomial vs exponential bounds
4. The class \( P \): the class of languages where membership can be decided in polynomial time.
5. The class \( NP \): the class of languages where membership can be verified in polynomial time.
6. Cook-Levin Theorem: \( SAT \in P \) iff \( P=NP \). That is, \( SAT \) is NP-complete.

Problem 2: Let’s get some practice with asymptotic bounds. Roughly, you can think of these notations as follows (see Section 7.1 for precise definitions):

1. Big-O: \( a(n) = O(f(n)) \) means that \( a(n) \) is less than or equal to a constant multiple of \( f(n) \) for every \( n \), once \( n \) is sufficiently large (i.e., an “upper bound”).
2. Big-Ω: \( c(n) = \Omega(f(n)) \) means that \( c(n) \) is greater than or equal to a constant multiple of \( f(n) \) for every \( n \), once \( n \) is sufficiently large (i.e., a “lower bound”).
3. \( \Theta \): \( d(n) = \Theta(f(n)) \) means that \( d(n) = O(f(n)) \) and \( d(n) = \Omega(f(n)) \).
4. Small-o: \( b(n) = o(f(n)) \) means that \( b(n) = O(f(n)) \) and \( b(n) \neq \Omega(f(n)) \).

Now, answer TRUE or FALSE for each of the following.

1. \( n^2 = O(n^2+n) \). TRUE
2. \( 2^n = 5^{O(n)} \). TRUE
3. \( n^{1000000} = o(1.0000001^n) \). TRUE
4. For \( c_1 < c_2 \), \( n^{c_1} = o(n^{c_2}) \). TRUE

Problem 3: Prove that NP is closed under the star operation.

Solution 3: For \( A \in NP \), let \( M \) be the NTM that decides \( A \). We construct NTM \( M' \) that decides \( A^* \) in polynomial time.

\( M' \) = “On input \( w \):
1. Nondeterministically break \( w \) into pieces \( w_1w_2\ldots w_k \).
2. For each substring \( w_i \) run \( M \) on \( w_i \), reject if \( M \) rejects.
3. Accept.”

Step 1 takes time proportional to \( |w| \) because the splitting is done using nondeterminism, so there is just a
matter of marking the boundaries of the substrings. Step 2 takes polynomial time because there are at most \(|w|\) substrings and \(M\) runs in polynomial time on each. And, obviously, step 3 takes polynomial time.

**Problem 4:** Let MAXCUT = \{⟨G, k⟩\} \(G = (V, E)\) is an undirected graph and \(V\) can be partitioned into disjoint sets \(V_L\) and \(V_R\) such that the number of edges in \(E\) with one endpoint in \(V_L\) and the other in \(V_R\) is at least \(k\). Prove that MAXCUT is in NP.

**Solution 4:** Describe a polynomial time verifier for MAXCUT. Given a certificate specifying the edges in the cut and the sets \(V_L\) and \(V_R\):
1. Verify that the number of edges in the cut is at least \(k\).
2. Verify that every vertex in \(V\) is in either \(V_L\) or \(V_R\) but not both.
3. For each edge in \(G\) not in the cut, check that both of its vertices are in the same partition, i.e. check that the edge wasn’t left out of the cut.
4. For each edge in the cut, check that its vertices are in different partitions, i.e. that it really is an edge in the cut.

**Problem 5:** Describe the error in the following fallacious proof that \(P \neq NP\). Consider an algorithm for the problem 3COLOR = \{⟨G⟩ \(G\) is a graph that can be colored “properly” with at most 3 colors\}: “On input a graph \(G\), try all possible colorings of the nodes with 3 colors. If any of these colorings is proper, accept. Else, reject.” Clearly, this algorithm requires exponential time. Thus 3COLOR has exponential time complexity. Therefore 3COLOR is not in \(P\). Because 3COLOR is in NP, it must be true that \(P \neq NP\). (Aha, thats it!! where is my million-dollar prize?!)\(^1\)

**Solution 5:** The fact that you can come up with an exponential time algorithm doesn’t mean that a polynomial time algorithm doesn’t exist. The claim “Thus 3COLOR has exponential time complexity.” is unsubstantiated.

**Problem 6:** Let HALF-CLIQUE = \{⟨G⟩ \(G\) is an undirected graph having a clique of size at least \(n/2\), where \(n\) is the number of vertices in \(G\) \}. Show that HALF-CLIQUE is NP-complete. (Build on the CLIQUE problem).

**Solution 6:** We give a polynomial time mapping reduction from CLIQUE to HALF-CLIQUE. The input to the reduction is a pair ⟨\(G, k\)⟩ and the reduction produces the graph ⟨\(H\)⟩ as output where \(H\) is as follows. If \(G\) has \(M\) nodes and \(k = m/2\) then \(H = G\). If \(k < m/2\), then \(H\) is the graph obtained from \(H\) by adding \(j\) nodes, each connected to every one of the original nodes and to each other, where \(j = m - 2k\). Thus \(H\) has \(m + j = 2m - 2k\) nodes. Observe that \(G\) has a \(k\)-clique iff \(H\) has a clique of size \(k + j = m - k\) and so ⟨\(G, k\)⟩ ∈ CLIQUE iff ⟨\(H\)⟩ ∈ HALF-CLIQUE. If \(k > m/2\), then \(H\) is the graph obtained by adding \(j\) nodes to \(G\) without any additional edges, where \(j = 2k - m\). Thus \(H\) has \(m + j = 2k\) nodes, and so \(G\) has a \(k\)-clique iff \(H\) has a clique of size \(k\). Therefore ⟨\(G, k_k\)⟩ ∈ CLIQUE iff ⟨\(H\)⟩ ∈ HALF-CLIQUE.

We also need to show that HALF-CLIQUE ∈ NP. The certificate is simply the clique.

**Problem 7:** (Sipser 7.28) Show that, if \(P=NP\), we can factor integers in polynomial time.

(Note: NP is a class of languages and the factoring problem is a function. Thus simply saying that, “because factoring is in NP, you are done” isn’t enough!)

**Solution 7:** We can formulate the problem is a language such as: \(L = \{⟨x, y⟩\} y\) has a factor \(z\) s.t. \(1 < z < x\). Then, we can perform a binary search (or just iterate through values of \(x\)) using our polynomial time algorithm to decide instances of \(L\) to find the smallest factor of \(y\). We can divide \(y\) by that factor and then repeat this step on the quotient, and so on until we have found all of the factors. The number of factors is proportional to the log of the number. For each time we repeat this process the binary search requires \(O(\log(n))\) time, the algorithm to decide an instance of \(L\) requires polynomial time (since we are assuming \(P=NP\)) and the division can be done in polynomial time.

\(^1\)http://www.claymath.org/millennium/P_vs_NP/

6: Time Complexity, P, and NP, NP-Completeness-2
Note: we must represent our numbers $x$ and $y$ using binary or some other reasonable representation. If we treat this problem is nondeterministically dividing up a string into equal length substrings we would be using a unary representation of our number, which is exponentially larger than the binary representation.