Advanced Algorithms and Data structures
Assignment four

Magnus Goltermann (xzb187), Thomas Busk-Jepsen (tnr653)
Mia Rahlff Pedersen(bvx284)

December 17, 2024

1 34.2-10

Since we know that NP $\neq$ co-NP, we can let $L \in$ NP but $L \notin$ co-NP. Since we know that P $\subseteq$ NP $\cap$ co-NP, and we know that $L \notin$ NP $\cap$ co-NP, we have that $L \notin$ P. And therefore P $\neq$ NP

2 34.3-6

To prove that this is the case we will first prove that the languages $\emptyset$ and $\{0,1\}^*$ are not complete for P as they are trivial cases that can't be used, and then that all other languages are.
The language $\emptyset$ will reject all strings and therefore cannot encode any other problems in P. This is the same for the language $\{0,1\}^*$ but here we however just accepts all strings.
If we then look at any other language L not $\emptyset$ or $\{0,1\}^*$ then we have L' that is in P. We then have a $\notin$ L' and b$\in$ L'. Since L' $\in$ P we have a polynomial time algorithm , that can return 1 if x $\in$ L and 0 if $\notin$ L. Then we can define our reduction to map over the accepting states in polynomial time.

3 34.4-3

Professor Jaggers strategy have n free variables. This means that the truth table for the formula will have 2^n, as it needs to consider every possible assignments for every n variable. This is means, that this strategy will grow exponentially and not polynomial.

4 34.4-6

If we use algorithm A, which can denote if a formula is satisfiable or not in $p(x)$ time, we can just change one variable to a true one at a time. If the formula

is still satisfiable using $p(x)$, we know that the full formula can be satisfiable with that set to true (and if not then set to false). We then just iterate over all variables, and thus it must take at most $2n$ to iterate over all variables, and thus a full time complexity of $p(x) \cdot 2n = O(p(x))$

5 34.5-1

To prove that the problem is NP-complete we must first prove that the problem is in NP and further that all other problems reduces to this.
The problem is in NP as we can just use the injection from G_1 to G_2 where G_1 is isomorphic as the certificate
To prove that it is NP-complete we must reduce another NP-complete to this problem in polynomial time. Here we will use the clique problem. Here we will take G from the clique problem, then construct G_1 as the complete graph K_k, then set $G_2 = G$. Now if we can solve the isomorphic problem in polynomial time, we can also solve the clique problem as K_k is our clique we are looking for.

6 34.5-3

To show that the integer linear-programming problem is NP- complete, we need to show why the problem is in NP, and how to reduce the 0-1 integer-programming problem, as we may assume that it is NP-hard.
We start of by restating the problem.
Given an integer $m \times n$ matrix A and an integer m-vector b, determine whether there exists an integer n-vector x with elements in $\{0, 1\}$, such that:

$$Ax \leq b$$

Ax can be computed in $O(nm)$ time, and the pairwise comparison of Ax and b can be done in $O(m)$ time. So verifying $Ax \leq b$ can be done in polynomial time.
Now reducing the 0-1 integer-programming problem, we start of noting that the $x_i \in \{0, 1\}$ constraint is valid in the integer linear-programming problem as it takes all integer values. Then to ensure that the variables x_i remains 0 or 1, we add the constraint

$$0 \leq x_i \leq 1 \quad \text{and} \quad x_i \in \mathbb{Z}$$

If the original 0-1 integer-programming problem has a solution, then the corresponding integer linear programming problem also has a solution. On the other hand, if the integer linear programming problem has a solution, it must satisfy the 0-1 constraints, so it is also a solution to the original 0-1 integer-programming problem. The transformation between these two remains in polynomial time, as it only involves a simple bound $0 \leq x_i \leq 1$ on each variable. Since we now know that the 0-1 integer-programming problem reduces to the Integer linear programming problem, and we may assume that the 0-1 problem

is NP-hard, we have now shown that the Integer linear programming problem is NP-complete.

7 34.5-6

First, we need to show that HAM-PATH is in NP, and then show that some other NP-complete problem reduces to HAM-PATH. We see that a certificate containing vertices $\{v_1, v_2, \ldots, v_n\}$, which can easily be verified in polynomial time to be a Hamiltonian path (since we just check for each vertex is visited just once).
We know that the problem HAM-CYCLE is NP-complete and thus we do a reduction: $HAM\_CYCLE \leq_p HAM\_PATH$ in polynomial time. We can do this by taking the graph $G = (V, E)$ and add two new vertices s and t, where each of these new vertices have an edge to all other vertices in G such that we get $G' = (V', E')$ where

$$V' = V \cup \{s, t\}$$

$$E' = E \cup \{(s, v) | v \in V\} \cup \{(t, v) | v \in V\}$$

This transformation can be done in polynomial time, as it just adds to vertices and $2|V|$ edges.
It is trivial that if a Hamiltonian cycle is present, then there must also be a Hamiltonian path, as we can just break the cycle at any vertex and create a path. It is not directly clear however for a Hamiltonian path to be a cycle. Here we can utilize the newly formed G' with a path starting in s and ending in t, and since they are connected to all other vertices, it must also be able to form a cycle.
And since we can reduce $HAM\_CYCLE \leq_p HAM\_PATH$, HAM_PATH is NP-complete.

8 34.5-7

Firstly, the decision problem would be:
Given a graph G and a positive integer k determine whether G has a cycle with length at least k
We now need to show that this decision problem is NP-complete, by showing that it is in NP, and that it has a reduction, which is NP-hard.
A solution for the problem is a sequence of vertices, that represents a simple cycle with k vertices. Verification will be in polynomial time, as k-length and distinct vertices will be done in time $O(k)$ and edge existence will take $O(k)$ look-ups.
We now need to reduce a NP-complete problem to the longest-simple-cycle problem. We will choose to go with the Hamiltonian cycle problem, as it takes a given graph $G = (V, E)$ and determine whether there exists a simple cycle that visits all $|V|$ vertices exactly once. To make it the Longest-simple-cycle problem, we use the same graph as in Hamiltonian cycle($G = (V, E)$), and set $k = |V|$.

If G has a Hamiltonian cycle, then there exists a simple cycle in G that visits all $|V|$ vertices. So the answer to the Longest-Simple-Cycle Decision Problem with $k = |V|$ is "yes." Also if the Longest-Simple-Cycle Decision Problem with $k = |V|$ has a "yes" answer, then G has a simple cycle that visits all $|V|$ vertices, which is a Hamiltonian cycle.
Since the Hamiltonian Cycle Problem reduces to the Longest-Simple-Cycle Decision Problem in polynomial time (the transformation only involves setting $k = |V|$), the Longest-Simple-Cycle Decision Problem is NP-hard. We have now shown that the problem is NP, and reduces to a NP-complete problem, making it NP-hard, and thus making it NP-complete.