Assignment 16:

R 13.1: *Has Professor Amongus proven that P=NP? Why or why not?*

- No, Professor Amongus has not proven that P=NP. Proving that a problem LL is polynomial-time reducible to an NP-complete problem MM and showing that LL can be solved in polynomial time only implies that LL is in the class PP, but it does not prove that all NP-complete problems can be solved in polynomial time. To prove P=NPP=NP, one must show that every problem in NPNP can be solved in polynomial time, not just one specific problem. Thus, while LL is in PP, the relation between PP and NPNP remains unresolved.

R 13.3: *Show that the problem SAT is NP-complete.*

- To prove that SAT (the Boolean satisfiability problem) is NP-complete, we must show two things:
    1. **SAT is in NP**: Given a Boolean formula SS, if we are provided a truth assignment that satisfies SS, we can verify in polynomial time whether the assignment satisfies SS. Thus, SAT is in NP.
    2. **SAT is NP-complete**: We need to reduce every problem in NP to SAT in polynomial time. Cook's theorem shows that any problem in NP can be transformed into an instance of SAT. This is done by encoding the computation of a non-deterministic Turing machine as a Boolean formula. Hence, SAT is NP-complete.

R 13.13: *Is there a subset of the numbers in {23, 59, 17, 47, 14, 40, 22, 8} that sums to 100? What about 130?*

    - For 100: We can check subsets of the set {23, 59, 17, 47, 14, 40, 22, 8}. One valid subset is {23, 47, 14, 8}, which sums to 23+47+14+8=10023 + 47 + 14 + 8 = 100. Therefore, a subset exists that sums to 100.
    - For 130: Checking for subsets that sum to 130, the subset {59, 47, 14, 8, 22} sums to 59+47+14+8+22=13059 + 47 + 14 + 8 + 22 = 130. Therefore, a subset also exists that sums to 130.

A. Prove that the Set-Partition decision problem is a member of class NP:

- To prove that Set-Partition is in NP, we must show that if we are given a potential solution, we can verify in polynomial time whether it is correct.
    - **Verification process**: Given a set SS and a partition of SS into two disjoint subsets P1P1 and P2P2, we can calculate the sum of the elements in each partition and check whether the sums are equal. If they are, the solution is correct.
    - This verification can be done in polynomial time, as summing the elements and comparing two sums is linear in the size of the input set. Therefore, Set-Partition is in NP.

B. Evaluate the Subset-Sum to Set-Partition reductions:

There are four proposed algorithms to reduce Subset-Sum to Set-Partition. We need to determine which are valid and which are not.

- **Algorithm SS2SP_v1(S, T)**:
- sum ← 0
- for each i in S do
- sum ← sum + i
- S.insertLast(sum)
- return S

Invalid: This algorithm adds the total sum of the set to itself without considering the target TT. A counterexample would be S={3,5}S = \{3, 5\} and T=5T = 5. Adding the total sum (which is 8) to the set doesn't solve the problem as we are looking for subsets that sum to TT, not to the total sum.

- **Algorithm SS2SP_v2(S, T)**:
- sum ← 0
- for each i in S do
- sum ← sum + i
- S.insertLast(T)
- S.insertLast(sum-T)
- return S

Valid: This algorithm works because it inserts both the target TT and sum−Tsum - T, creating an artificial partition where one subset sums to TT and the other to sum−Tsum - T. If such a partition exists, the Set-Partition problem would succeed.

- **Algorithm SS2SP_v3(S, T)**:
- S.insertLast(T)
- return S

Invalid: This algorithm simply adds TT to the set without altering the set in any meaningful way. It does not account for finding subsets that sum to TT.

- **Algorithm SS2SP_v4(S, T)**:
- sum ← 0
- for each i in S do
-   sum ← sum + i
- S.insertLast(T)
- S.insertLast(sum+T)
- return S

Invalid: This algorithm adds the sum of the set and TT to the set, but this does not help in solving the partition problem for the subset-sum. A counterexample would be S={3,5}S = \{3, 5\} and T=5T = 5. Adding the total sum plus TT doesn't lead to a solution.

Thus, **Algorithm SS2SP_v2(S, T)** is the valid reduction, while the others are not.