COMP 3804 — Tutorial March 18, 2026

Problem 1: Let $a_1, a_2, \ldots, a_n$ be a sequence of n numbers. In class, we have seen that the length of a longest increasing subsequence can be computed by converting the sequence to a directed acyclic graph and then doing a longest path computation in this graph.

Use the three step approach from class to design a dynamic programming algorithm that computes the length of a longest increasing subsequence.
Hint: For $i = 1, 2, \ldots, n$, define $d(i)$ to be the length of a longest increasing subsequence that ends with a_i.

Problem 2: When is a decision problem L in the class **NP**?

- Give the informal definition.
- Give the formal definition.

Problem 3: A *Hamilton cycle* in an undirected graph is a cycle that visits every vertex exactly once.

Consider the decision problems

$$\text{HamiltonCycle} = \{G : \text{ graph } G \text{ contains a Hamilton cycle}\}$$

and

$$\text{NonHamiltonCycle} = \{G : \text{ graph } G \text{ does not contain a Hamilton cycle}\}.$$

- Explain, informally, why HamiltonCycle is in the class **NP**.
- Justin Bieber claims that NonHamiltonCycle is in the class **NP**. Here is Justin's reasoning:
  - HamiltonCycle is a YES/NO problem.
  - NonHamiltonCycle is also a YES/NO problem.
  - Since HamiltonCycle is in **NP**, by flipping YES and NO, it follows that NonHamiltonCycle is in **NP**.

  Why should Justin fail COMP 3804? (By the way, it is not known whether or not NonHamiltonCycle is in **NP**.

Problem 4: Consider the decision problem

$$\textsc{NonPrime} = \{x : \ x \text{ is an integer which is not a prime number}\}.$$

- Explain, informally, why NonPrime is in the class **NP**.
- Give a formal proof that NonPrime is in the class **NP**.

Problem 5: Consider the decision problem

$$\textsc{Clique} = \{(G, K) : \ \text{graph } G \text{ has a clique of size } K\}.$$

Assume you have a polynomial-time algorithm A that decides, for any input (G, K), whether or not $(G, K) \in$ Clique. Note that this algorithm only returns YES or NO; it does not return anything else.

Design a polynomial-time algorithm B that takes (G, K) as input.

- If $(G, K) \in$ Clique, then B returns a clique of size K.
- If $(G, K) \notin$ Clique, then B returns NO.

Your algorithm B may use algorithm A as a black box.