Name: _____ Wisc id: _____

The Feast of the Great Houses

In Westeros, a grand feast is being organized to celebrate the peace among the Great Houses. To ensure fairness and maintain this fragile peace, the feast's resources must be divided equally among the 9 *Great* Houses.

Each of the n Noble Houses has pledged a certain value of resources (represented by integer v_i from Noble House i) for the feast. Your task is to prove that determining whether these resources can be partitioned into 9 subsets with equal total values, representing the 9 *Great* Houses, is NP-complete

The Secret Dragon Egg Cache

During the reign of King Viserys I, deep in the ancient Targaryen vaults, a secret cache of dragon eggs has been discovered, each egg i possessing unique magical properties and values of m_i represented as an integer. To prevent the rise of a single dominant power, Viserys decided to distribute these eggs equally among two factions of House Targaryen: The Greens and the Blacks.

However, the distribution is not merely about equalizing the total magical value of the eggs; it is also essential that each faction receives the same number of eggs to maintain a balance.

The task is to determine if it's possible to partition the dragon eggs into two groups where each group has not only an equal total magical value but also an equal number of eggs. This intricate problem, central to the future power dynamics of Westeros, is suspected to be NP-complete, and you are tasked with proving this.

Intractability recap

Polynomial-Time Reductions

$Y \leq_p X$

- Consider any instance of problem Y.
- Assume we have a black-box solver for problem X.
- Efficiently transform an instance of problem Y into a polynomial number of instances of X that we solve via black-box solver for problem X, and aggregate the solutions efficiently to solve Y.

Corollaries from polynomial time reductions:

- Suppose $Y \leq_p X$. If X is solvable in polynomial time, then Y can be solved in polynomial time.
- Suppose $Y \leq_p X$. If Y cannot be solved in polynomial time, then X cannot be solved in polynomial time.
- Suppose $Z \leq_p Y$ and $Y \leq_p X$. Then, $Z \leq_p X$.

Intractability Definitions

Decision Problems

- Binary output: yes / no answer.
- Our complexity definitions assume decision problem versions of the problems.
- No less powerful: we can go between decision and optimization version of problems.

Easy vs Hard Problems

- *Easy Problems*: problems that can be solved by efficient algorithms.
- *Hard Problems*: problems for which we do not know how to solve efficiently.

Input Formalization

- Let s be a binary string that encodes the input.
- $|s|$ is the length of s, i.e., the # of bits in s.

Complexity class: P

- Polynomial run-time: Algorithm A has a *polynomial run-time* if run-time is $O(\mathsf{poly}(|s|))$ in the worst-case, where $\mathsf{poly}(\cdot)$ is a polynomial function.
- P is the set of all problems for which there exists an algorithm A that solves the problem with polynomial run-time.

Efficient Certification: Certifier $B(s, t)$ for a problem P:

- s is an input instance of P.
- t is a certificate; a proof that s is a yes-instance.
- Efficient: For every s, we have $s \in P$ iff there exists a t, $|t| \leq \mathsf{poly}(|s|)$, for which $B(s, t)$ returns yes.

Complexity class: NP

- Set of all problems for which there exists an efficient certifier $B(s,t)$.
- I.e., the set of all problems for which it is efficient to verify a potential solution.
- **N**on-deterministic, **P**olynomial time: can be solved in polynomial time by testing every certificate (t) simultaneously (non-deterministic).

Complexity class: NP-Hard: Problem X is NP-Hard if:

- For all $Y \in \mathsf{NP}$, $Y \leq_p X$.
- NP-Hard problem may or may not be in NP.

Complexity class: NP-Complete: Problem X is NP-Complete if:

- For all $Y \in \mathsf{NP}$, $Y \leq_p X$.
- X is in NP.

Showing that Problem X is NP-Complete: Cook Karp Reduction

Step 1: Prove that $X \in \mathsf{NP}$.

(a) Define a certificate (t) for X.

(b) Define an efficient certifier (algorithm) $B(s,t)$ for X and t as defined in (a).

Step 2: Choose a problem $Y \in \mathsf{NP}$-Complete.

- Y must be a problem that is known to be NP-Complete.
- It will be used to show that $Y \leq_p X$ in step 3:
  - Since $Y \in \mathsf{NP}$-Complete, then all NP problems $\leq_p Y$.
  - Therefore, showing $Y \leq_p X \implies$ all NP problems $\leq_p X \implies X \in \mathsf{NP}$-hard.

Step 3: $Y \leq_p X$ ($X \in \mathsf{NP}$-hard).

- Karp Reduction: For an arbitrary instance s_Y of Y, show how to construct, in polynomial time, an instance s_X of X such that s_y is a yes iff s_x is a yes.
  Steps:
  (a) Provide efficient reduction.
  (b) Prove $\Rightarrow$: if s_Y is a yes, s_X is a yes.
  (c) Prove $\Leftarrow$: if s_X is a yes, then s_Y had to have been a yes.