1. **Dynamic Programming and Greedy Algorithms** (AY 22/23 Sem 2 Finals B.2, Modified). You are given a **tree** graph with n vertices and $n-1$ edges. (A tree is a connected graph with no cycles.) Your task is to devise an algorithm to find the size of the maximum independent set of the tree, that runs in $O(n)$ time.

   (a) Let's first attempt to solve the problem using dynamic programming.

      i. Identify the optimal substructure in the problem.
      (Hint: What is a feasible order in making the decisions? What are the "subproblems" in this case?)

      ii. Based on the optimal substructure identified, design a $O(n)$ dynamic programming algorithm for the problem.

   (b) Indeed, it is possible to solve this problem using a greedy algorithm.

      i. State the greedy choice property and provide a formal proof.
      (Hint: You can argue that certain vertices in the tree can be chosen in some optimal solution.)

      ii. State the optimal substructure property that can be used together with the greedy choice property and provide a formal proof.
      (Hint: It is different from the optimal substructure in (a). You could simply remove some vertices from the tree.)

      iii. Combine the greedy choice property and the optimal substructure property to design a $O(n)$ time algorithm for the problem.

2. **Amortized Analysis** (AY 22/23 Sem 2 Finals B.3). Consider a standard (LIFO) stack that supports the following operations in $O(1)$:

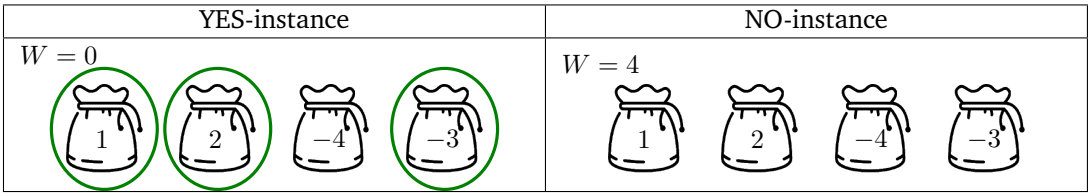
   - PUSH(x): Add item x at the top of the stack.
   - PULL(): Remove and return the top item present in the stack.

   Using *two* such stacks, implement a standard (FIFO) queue such that PUSH, PULL and SIZE operations remain amortized $O(1)$. Use **potential method** to show that in any intermixed sequence of PUSH, PULL and SIZE operations, the amortized cost of each of them is $O(1)$. (Assume we start from an empty queue.)

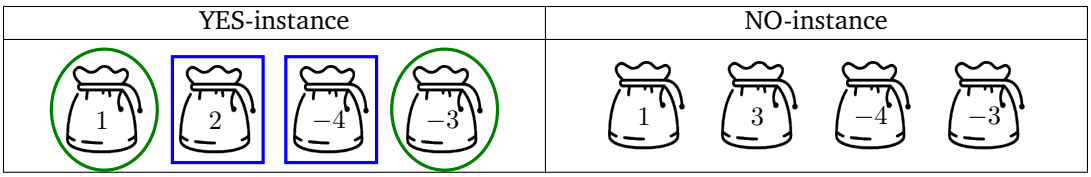
   - PUSH(x): Add item x at the end of the queue.
   - PULL(): Remove and return the first item present in the queue.
   - SIZE(): Return the number of elements present in the queue.

3. **Reductions** (New Question). Consider the following two problems.

   - **SUBSET-SUM**: Given a set of n non-negative integers $S = \{w_1, \ldots, w_n\}$ and a target W, decide whether there exists a subset $I \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in I} w_i = W$. (In other words, decide whether there exists a subset of S with sum W.)



   - **PARTITION**: Given a set of n non-negative integers $S = \{w_1, \ldots, w_n\}$, decide whether $\{1, 2, \ldots, n\}$ can be partitioned into two subsets I_1, I_2 such that $\sum_{i \in I_1} w_i = \sum_{i \in I_2} w_i$. (In other words, decide whether S can be partitioned into two subsets with equal sum.)



   Show that SUBSET-SUM $\leq_p$ PARTITION.