Problem 1 (8 points). Using the heap data structure to design a data structure that maintains a (multi-)set $S$ of numbers, and supports the following two operations:

- median: return the $\lceil (n + 1)/2 \rceil$-th smallest number in $S$, where $n = |S|$;
- add($e$): add the number $e$ to $S$.

The running time for each operation should be $O(\lg n)$, where $n$ is the current size of $S$. The following is an example for a sequence of operations: add(5), median returns 5, add(10), median returns 5, add(7), median returns 7, add(1), median returns 5, add(6), median returns 6.

Problem 2 (8 points). An independent set of a graph $G = (V, E)$ is a set $U \subseteq V$ of vertices such that there are no edges between any two vertices in $U$. The maximum independent set problem asks for the independent set of $G$ with the maximum size. The problem is very hard on general graphs. Here we want to solve the problem on trees: given a tree $T = (V, E)$, find the maximum independent set of the tree. For example, the maximum independent set of the tree in Figure 1 has size 7.

![Figure 1: The green vertices shows that the maximum independent set of the tree has size 7.](image)

Design an $O(n)$-time greedy algorithm for the problem, where $n$ is the number of vertices in the tree. We assume that the vertices of the tree are $\{1, 2, 3, \ldots, n\}$. For simplicity, we assume the tree is already rooted at vertex 1 and the parent of each vertex $i \in \{2, 3, \ldots, n\}$ is a vertex $j < i$. In the input, we give the parent of $i$ for each $i \in \{2, 3, \ldots, n\}$. The instance in Figure 1 is the parent array $(0, 1, 1, 1, 2, 5, 5, 5, 4, 4)$. (The parent of 1 is not defined; so we use 0.) We algorithm should return the set $\{2, 3, 6, 7, 8, 9, 10\}$. 
Problem 3 (8 points). Given a set of \( n \) points \( X = \{x_1, x_2, \cdots, x_n\} \) on the real line, we want to use the smallest number of unit-length closed intervals to cover all the points in \( X \). For example, the points \( X \) in Figure 2 can be covered by 3 unit-length intervals. Design a greedy algorithm to solve the problem.

![Figure 2: Using 3 unit-length intervals (denoted by thick lines) to cover points in \( X \) (denoted by the solid circles).](image)

Problem 4 (8 points). Consider a \( 2^n \times 2^n \) chessboard with one arbitrary chosen square removed. Prove that any such chessboard can be tiled without gaps by L-shaped pieces, each composed of 3 squares. Figure 3 shows how to tile a \( 4 \times 4 \) chessboard with the square on the left-top corner removed, using 5 L-shaped pieces.

![Figure 3: Using 5 tiles to cover a chessboard of size \( 4 \times 4 \), with the left-corner missing.](image)

Problem 5 (8 points). Suppose there has \( n \) balls (indexed by \( 1, 2, \cdots, n \)) with different weights and let \( b \) be an integer between 2 and \( n \). There is a magic machine which, given a set \( S \subseteq \{1, 2, 3, \cdots, n\} \) of size at most \( b \), can tell you the lightest ball in \( S \). Your goal is to sort the \( n \) balls according to their weights, using only a few queries to the machine.

1. Give an algorithm that sorts the \( n \) balls using \( c \lg_b(n!) \) queries, where \( c \) is an absolute constant independent of \( b \) and \( n \).

2. Prove that any correct algorithm to sort the \( n \) balls needs at least \( \lceil \log_b(n!) \rceil \) queries to the machine.