AGENDA

Miscellaneous Problem Solving:

- Optimal Strategy for a Game
- Nodes at a Distance K in a Binary Tree
- Understanding topological sort based problems:
  Does your problem match this pattern?

  Yes, if either of these conditions is fulfilled:
  a. Dependency relationships: The problem involves tasks, jobs, courses, or elements with dependencies between them. These dependencies create a partial order, and topological sorting can be used to establish a total order based on these dependencies.
  b. Ordering or sequencing: The problem requires determining a valid order or sequence to perform tasks, jobs, or activities, considering their dependencies or prerequisites.

  No, if either of these conditions is fulfilled:
  a. Presence of cycles: If the problem involves a graph with cycles, topological sorting cannot be applied because there is no valid linear ordering of vertices that respects the cyclic dependencies.
  b. Dynamic dependencies: If the dependencies between elements change dynamically during the execution of the algorithm, topological sorting may not be suitable. Topological sorting assumes static dependencies that are known beforehand.

Optimal Strategy for a Game

You are given an integer array **arr[]** of size **n**. The array elements represent **n** coins of values $v_1, v_2, ...v_n$. You play against an opponent in an alternating way. In each turn, a player selects either the **first** or **last** coin from the **row**, removes it from the row permanently, and **receives the coin's value**. You need to determine the **maximum** possible amount of money you can win if you **go first**.
Note: Both the players are playing optimally.

Examples:

Input: arr[] = [5, 3, 7, 10]
Output: 15
Explanation: The user collects the maximum value as 15(10 + 5). It is guaranteed that we cannot get more than 15 by any possible moves.

Input: arr[] = [8, 15, 3, 7]
Output: 22
Explanation: The user collects the maximum value as 22(7 + 15). It is guaranteed that we cannot get more than 22 by any possible moves.

Nodes at a Distance K in a Binary Tree

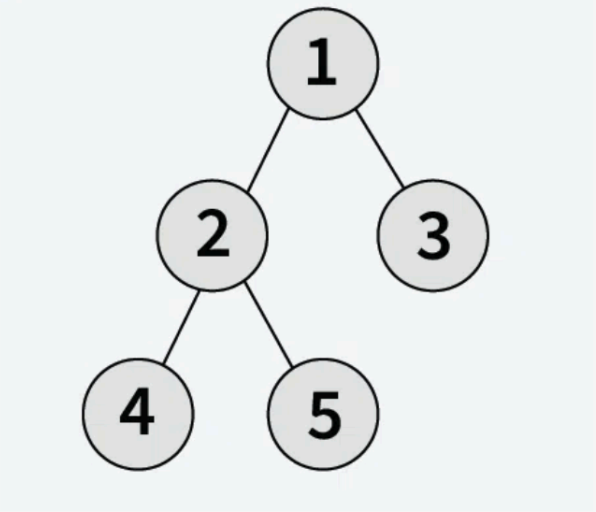
Given a binary tree, a target node in the binary tree, and an integer value k, find all the nodes that are at a distance k from the given target node. No parent pointers are available.

Note:

- You have to return the list in sorted order.
- The tree will **not** contain **duplicate** values.

Examples:

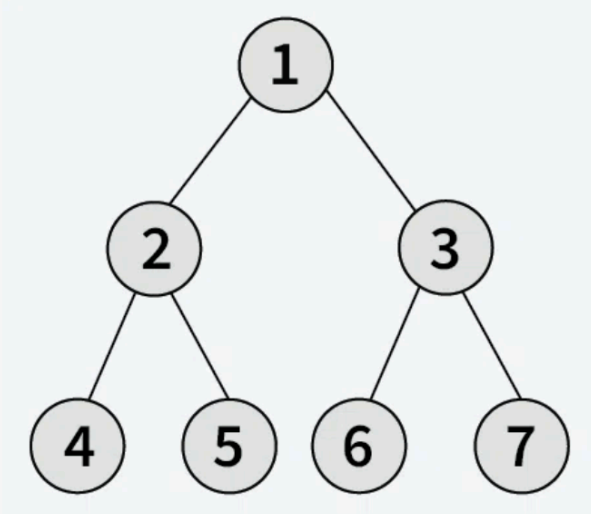
Input: root = [1, 2, 3, 4, 5], target = 2, k = 2



Output: [3]

Explanation: Nodes at a distance 2 from the given node 2 is 3.

Input: root = [1, 2, 3, 4, 5, 6, 7], target = 3, k = 1



Output: [1, 6, 7]

Explanation: Nodes at a distance 1 from the given target node 3 are 1, 6 & 7.