The problem requires creating a directed graph where the nodes are tasks and the directed edges are dependencies. To perform each task, the robot has to do all the tasks that needs to be performed before it. As the robot can do any number of tasks at a given time, the number of nodes at each level can be ignored. It is only required to find the highest (i.e. the senior most) ancestor of each node. As the distance and time to perform a task, both are units, so distance from the highest node to the current node is the highest amount of time required to perform the current task.

This can be done in a number of ways. One approach would be to find the lowest possible descendant for each node using DFS/BFS or finding the highest possible ancestor for an inverted graph using DFS/BFS. Another approach can be to sort the nodes topologically and then the difference in levels would be the maximum distance.

As the number of queries can be 10^5, the answers for all the nodes need to be saved before answering queries for faster processing.