**分布式匈牙利算法调研**

大致流程：首先给机器人和任务设置一个标杆，分别为αi和βj，Cij代表机器人完成任务所需的代价，Mi表示与机器人i匹配的任务，全局最小松弛δ。建立两棵树，分别为任务t的树F1和机器人r的树F2，每个机器人有四个指针，分别指向父节点，两个兄弟和儿子节点。

初始化，αi=min(Cij)，βj=0，Mi=0(无匹配)，父节点和儿子节点为空，兄弟节点指向附近机器人，每个机器人分别计算自己的最小松弛slack。

通过DFS计算全局最小松弛δ，并通过DFS把信息传给所有机器人，开始时最小松弛为0，通过DFS查找是否有为匹配的机器人，如果有则为它匹配，如果找到一个任务，使得αi+βj=Cij，并且匹配成功即该任务没有被匹配，则把该机器人连接到匹配到的任务下，即加到F1中，并且所有机器人的最小松弛将减小δ。

如果匹配失败，该机器人会通过任务传来的信息，把空闲的边和已匹配的边进行合适的交换，然后从树F1最底层开始回溯，把第一个没有父节点的任务连接到机器人的子节点上，并把该机器人和任务的信息传递给所有机器人，然后重新计算最小松弛。

继续DFS更新全局最小松弛，然后传给所有机器人，机器人和任务更新α和β的值，如果任务tj有机器人父节点，则把该机器人和任务的信息传递给所有机器人，然后继续进行上述匹配，最后直至DFS没有找到未匹配机器人。

总结：通过DFS传递信息，各种信息并不占用实际空间，所以内存空间占用少，但是每次迭代前的DFS传递信息可能会增加时间复杂度？