1 2 3

4 = [1,2]

5 = [1,3]

6 = [2,3]

7 = [4,5,6]

# 运动策略

影响运动就线速度和角速度

而且仅仅是赋值最高速度，并不能瞬间修改速度

线速度： -2<= v <=6

考虑F=ma ， F<=250

角速度： [-π，π]

最大力矩M=50

考虑M=j\*a，j=m\*r\*r

得 a = M/（m\*r²）

线速度方向不一定和robot朝向是相同方向，如果不同会出现漂移

对于角速度即决定旋转，那么有

1. >x->y，角速度的变化使得积分为所偏移角度

Robot点和工作台点

D

S

A

B

//////////////////////////////////////////////

# 调度策略

///////////////////////////////////////////////////

无非就是robot买产品和卖产品

还是一句话，有需求才有动力

每一帧，有需求的原料格发出信号，生产完毕的工作台发出信号，robot接受信号

1. robot手上恰好有对应产品
2. Robot没有任何产品

对于①，robot直接前往，对于②，robot去找买，随后再卖

很明确的是，一个信号，不可以被多个robot消费

对于①，robot确定要前往卖掉手中的产品时，应该消费掉对于原料格发出的信号，其他robot会接受这个信号，但不可消费该信号。对于下一帧，因为robot可能还没送到，所以该原料格仍然会发出信号，所以我们要禁止该原料格发出信号，直到robot送货到达

对于②，如果robot确定要消费该信号，那么就要去买对应的产品，此时原料格信号被消费，并且也要消费掉工作台发出的信号，下一帧同样不可再次发出相同信号。

但会发现一个矛盾的点，robot手上没产品时，依靠原料格发出的信号去买东西，而此时原料格信号已经被消费，买到东西的时候，转换为第①种情况，但是对于原料格已经没原料格信号了，怎么指引robot呢？

所以可以有

1. 对于一个robot确定其买和卖的地点
2. 给原料格两种信号，一种是指引robot去买的信号A，另一种是指引robot来卖的信号B，每种信号只能被消费一次

试想一下，初始状态每个robot均不携带产品，那么只有第②种情况，第1号策略是指定路径，买了东西就去A信号来源去卖，第2种策略却不一定去A信号来源卖，例如，两种同类型原料格均发出信号A，两个robot接受A，买到产品后是依靠信号B来卖的

如果初始robot携带产品，那么存在第①种情况。

我选择2号策略，那么有以下

接受信号A时，对A上锁，并去买东西，之后就不必管了，此时A信号来源的原料格仍然可以发出B信号，能不能由该robot送达无关紧要，只要在B信号被消费后且送达后将AB信号均解锁即可。

特殊的是，如果robot接受B信号并消费，应该将AB均上锁，不让原料格指引其他robot去买，送达后将AB信号均解锁即可。

上述会导致死锁，例如某一时刻原料格q发出A信号，robot消费该信号，在robot拿到产品时，另一个原料格w发出B信号，该robot接受，那么w的AB均被上锁，此时没有多余的A信号告诉其他robot要去拿q所需要的产品，导致q的A信号死锁，这是为什么第一版出现死机的问题

改进：在接受B信号的同时判断A信号有没有被锁，A信号没被锁，则跳过

选择该策略的优点是可以优化最短路径

具体实现（目前仅仅是按照顺序来实现的）

需要确定3种信号，A:原料格指引robot买 B:原料格指引robot卖 C:工作台指引robot买

AB信号用二维数组，C信号用一维数组即可

对于A信号，应该将7种原料分为【7,654,321】3个等级，以此保证不会对某个信号产生执着的信念，例如如果按照【7,6,5,4,3,2,1】会出现robot一直拿3号，接着拿2，接着拿1，这是不好的

对于B信号，接受此信号的robot手上是有产品的，为了快速定位，应该选择【7,6,5,4,3,2,1】的分法，快速定位该原料格的位置

对于C信号，同B