## 实验题目

荒野求生：

在一个1600X900的空间内有若干个探险小队，每个探险小队有初始的位置和速度，速度的方向有八个(U，D，L，R，LU，LD，RU，RD)，探险小队若碰撞到空间边缘则会转弯，转弯规则为（U->RD,D->LU,L->RU,R->LD,LU->R,LD->U,RU->D,RD->L）,速度大小不变。若在某个时刻，有多个小队同时到达某一位置，则会发生冲突，冲突后速度最慢小队会生存下来，若最慢的小队不只一个，则所有此位置的小队全部同归于尽。

输入数据包含若干行，第一行为一个整数T，表示结束时间，单位为s。其余每一行表示一个小队在0s时的状态。前两列为队伍的x坐标和y坐标。左下角为0坐标，向上为y坐标，向右为x坐标。第三列表示方向。第四列表示速度，为非负整数，单位为（格/s）。输出为若干时间后存活的小队的位置和速度大小以及方向。

注：仅考虑整数秒时的冲突。队伍的转弯和冲突是瞬时的，不消耗时间。

## 串行算法思路及伪代码

使用一个list维持当前存活的小队的状况，使用一个map保存当前时间所有位置以及在该位置的小队集合的对应关系。每隔一秒更新一次所有小队的位置，创建map，并进行一次冲突检测，同时更新冲突发生后的list。

输入：各小队初始状态，时间T。

输出：存活小队的状态。

**Begin：**

初始化L为一个保存所有小队状态的list。

**for** t=1 to T, **do**

初始化一个map数据类型M保存一秒后所有位置以及在该位置的小队集合的对应关系。

**for** each team in L, **do**

更新team运动1s后的位置pos。

向M[pos]中加入team。

**end for**

**for** each pos in M, **do**

**if** M[pos] 中包括一个以上的team（发生了冲突），**then**

**if** 只有一个最慢的小队，**then**

在L中删除M[pos]中除了最慢小队外所有的小队。

**else**

在L中删除M[pos]中的所有小队。

**end if**

**end if**

**end for**

**end for**

输出T时间后的L。

**End**

## 并行算法思路

并行算法的思路可以有很多种，大家可以自由发挥，这里仅提供一种思路。

将原始的矩形区域分块，每个线程/进程负责一块，每个线程/进程的处理步骤类似串行算法。如果经过一秒后，一个小队从某一线程/进程负责的块运动到另一个线程/进程p负责的块，则需要进行数据同步。每一次迭代结束后需要进行同步（即当所有线程/进程都完成了当前一次迭代的计算后，才能开始计算下一次的迭代）。