语言：python 3

概述：三个平台，N个任务，N个任务寻找合适的平台进行计算，在每一个平台完成任务有不同的方法，有不同的总成本。目的是最小化三个平台的总和成本。

详细描述：

初始化：

N个任务，定义为

表示计算Rn所需的输入数据的大小（单位比特），取值：（10,100）

表示完成计算任务后的以比特为单位的输出数据大小；取值（1，10）

Dn表示完成计算任务Rn每比特所需的CPU周期总数，取值：（0.01,0.1）

表示任务Rn的最大可容忍时间，取值：（0.1-10）

3 个平台 1，2，3，定义为（time，size，f，r）

Time是任务在该平台计算的最小时间，定义为0.1， 0.5 ，1

Size是平台共有的CPU周期数5 30 100

F是平台的计算能力（为每个任务可提供的CPU周期数），2， 5， 20（这里平台1和平台3的计算能力是直接用的，平台2的计算能力需要由q-learning分配）

r是平台对应的传输速率初始值为1,4,2（这里平台1和平台3的r是直接用的，就是r1和r3，r2需要通过q-learning来确定）

为资源1的分配向量，可以表示为。

为资源2的分配向量，可以表示为。

主函数：

输入N个任务（N在最初可指定，这N个任务都是随机生成的，满足上述取值范围）

使用函数1，确定选择的平台【最后需要得到一个平台选择向量，例如有3个任务，最后执行完，应该得到一个向量，例如[1,2,2]代表这三个任务选择的平台分别为1,2,2】

如果任务n选择为平台1，则继续判断（是一个定值，初始值设置为1（可调））,如果是，则将选择的平台改为2，并加入平台2任务集合,如果不是，得到在该平台下的该任务的成本

成本=

【最后需要计算在该平台下所有任务成本之和的总成本】

如果选择平台为2，将任务n加入平台2的任务集合

如果选择平台3，直接计算其成本，公式为 (这里的为定值，可以改变，这里假设为2)

【最后需要计算在该平台下所有任务成本之和的总成本】

对于平台2任务集合，调用算法2，得到最佳的A和B(在初始化部分有解释)，带入公式得到该平台（平台2）的总成本

总成本计算公式：a1+…+an+b1+…+bn的成本【其实就是完成一个任务对应一个ai和一个bi，在平台2的每个任务的成本就由ai和bi组成，A所对应的成本就是它本身的那个数值，B所对应的成本由其对应的计算公式】

N个任务里每个任务分配到的a和b都由Q-learning得到。

最后得到了N个任务的总成本之后，画图。

X轴为任务数（1到5），y轴为总成本

（循环100次求平均值，每一次任务随机生成）

X任务数就是N的取值，Y为每个N对应的总成本

（x=1对应的y就是这一个任务的总成本，x=3对应的y就是完成这两个任务的总成本）

函数1：

使用KNN算法，首先读取一系列数据（txt），格式如下：

0.3,2, 1

2,18,3

0.5,4,2

0.1,1,1

0.5,4,2

0.4,6,2

0.2,2,1

1.2,14,3

1,12,3

读取前两个数作为训练样本（time，cpusize）

第三个数为平台选择

得到模型后输入任意任务（，Dn）

得到平台

目标：任务选择合适平台计算

（这里我有个写好的你可以直接用）



函数2

指定N的值，例如为2（2个任务）（指定任务数，随机生成2个任务，并进行q-learning，这一步可以放在主函数，但调用q-learning需要单独调用）

【分配两次资源需要两次q-learning，计算奖励的定义也不同，拆成两个函数，分别调用，因为分配资源2的时候需要已经分配好的资源1的值】

首先进行资源1（W）的分配

进行Q-learning算法

状态 行为 奖励

状态：初始状态0状态，可转移的状态数WN（如果W=3.则可转移状态数9）

行为：由为n个任务分配的资源1的向量a组成，例如如果有两个任务则备选的行为有{[1,1],[1,2],[1,3],[2,1],[2,2],[2,3],…,[3,3]}

奖赏：根据行为进行计算：

If 行为向量里的数的和大于W，则奖赏为-1（如[1,3]：1+3=4＞W=3，奖赏为-1【超出总资源量，该分配为无效】）

这里有个最大值和一个最小值rmax，rmin（直接在函数里设置，假如为0.5,3，可调）

If a1<rmin or a1>rmax or a2<rmin or a2>rmax(a1,a2为行为向量里对应的值，如行为为[1,2],则a1=1,a2=2) ，则奖赏为-1

Else

奖励R=对应行为向量里数值之和的倒数，（[1,1]的奖励为1/(1+1)=1/2）

Q-learning主要是先确定所有的状态和所有的行为，计算所有行为对应的奖赏。然后不断迭代得到一个稳定的Q\_table，（迭代值可人为设置）

之后使用这个Q\_table，就可以知道新的任务应该分配的资源1的数目了（选择Q-table里值最大的对应的行为）

其次是资源2(U)的分配

进行Q-learning算法

状态 行为 奖励

状态：初始状态0状态，可转移的状态数(U+1)N（如果U=3.则可转移状态数16）

行为：由为n个任务分配的资源2的向量b组成，例如[0,0][0,1],[0,2],[0,3],[1,0],[1,1],[1,1],…,[3,3]

奖赏：根据行为进行计算：

If 行为向量里的数的和大于U，则奖赏为-1（如[1,3]：1+3=4＞U=3，奖赏为-1）

If b中存在0(如[1,0]) ,或者(b如果为([1,2]，则1或2对应的任务1和任务2有一个满足上述公式)

【这里的r2就需要用到资源1分配完之后得到的值a了，计算公式如下：



】

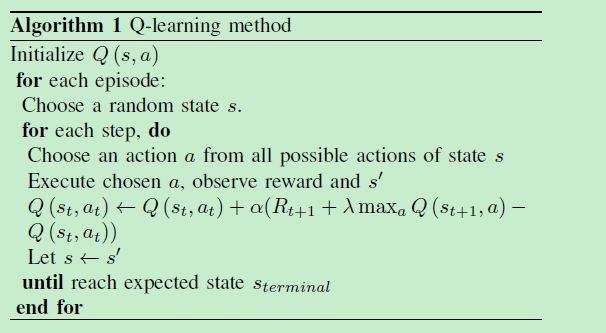
则奖励=（bn就是任务n对应的行为向量里的值）

Else

奖励=

这里的的计算需要上一步分配资源1得到的，资源1和资源2数量都是相对于相同的N个任务分配的。但分配资源2时需要首先分配资源1，得到最佳的行为a,为每个任务的赋值。

其余的都是q-learning的传统实现，伪代码如下：



最后希望可以得到1张图：

X轴为任务数（1到5），y轴为总成本

（循环100次求平均值，每一次任务随机生成）

总结一下，就是有N个任务，需要确定他们的计算平台，确定好计算平台后需要计算每个任务n在其分配的平台下的成本。如果分配到平台2，就需要使用q-learning实现确定分配给它的资源1，确定好后需要确定资源2，都确定好之后带入成本计算公式计算成本（在平台2的任务是需要整体考虑的，所以需要q-learning进行资源分配）。都完成后就可以得到N个任务的总成本。