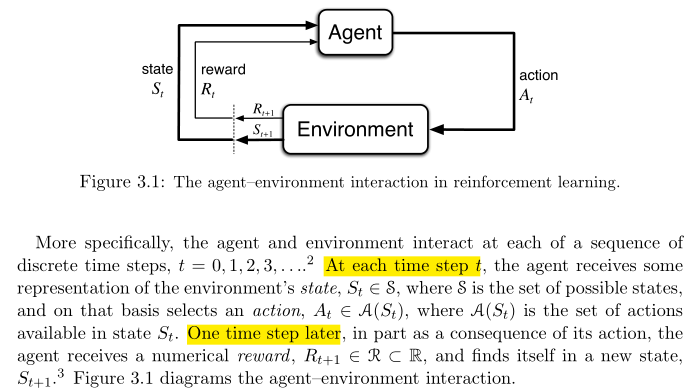
# 第二课

## 2.0强化学习典型结构



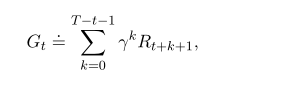
名词解释环节：

Policy：从state到probility分布的映射，强化学习就是在于环境的交互中去学习 更新policy。

Return：强化学习的目的是奖励最大化，确切的来说是从maximize the cumulated reward，那么表述方法也要episode、continue、整合三种：



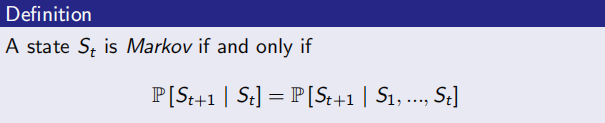




## 2.1 马尔科夫过程

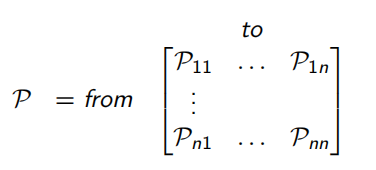
Markov decision process 是用来对环境建模的模型，这个环境是fully observable的，即便是partially observable也可以转化为MDP。所以在强化学习领域，几乎所有的问题都可以转化为MDP模型。

### 2.1.1 Markov property

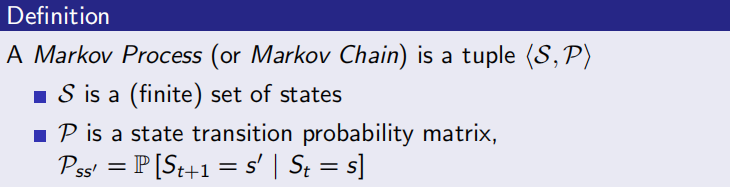


### 2.1.2 State Transition Matrix

假如agent有不同的状态，可以用状态转移矩阵，描述了不同状态之间转移的概率。如下图，每一行的概率和为1.

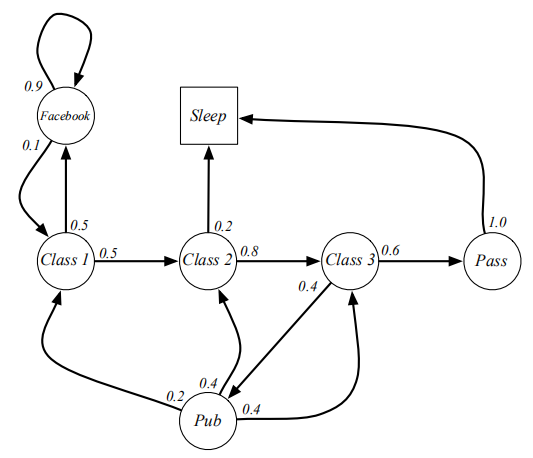


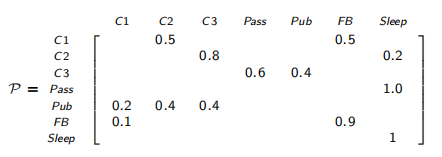
### 2.1.3 Definition of Markov Process



可以把马尔科夫过程描述成状态空间S和转移矩阵组成的二元组。

一个简单的例子，学生上课的状态转移图：



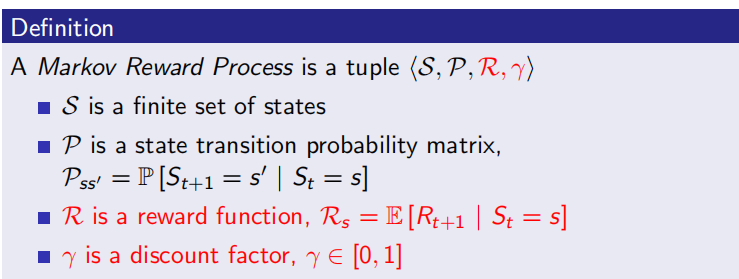


期间，有学生提问讲，如何应对随着时间的流逝，转移概率变化的情况，David提出的解决方案思路是动态的演变转移图，但这并不改变Markov的结构。

## 2.2 Markov Reward Process

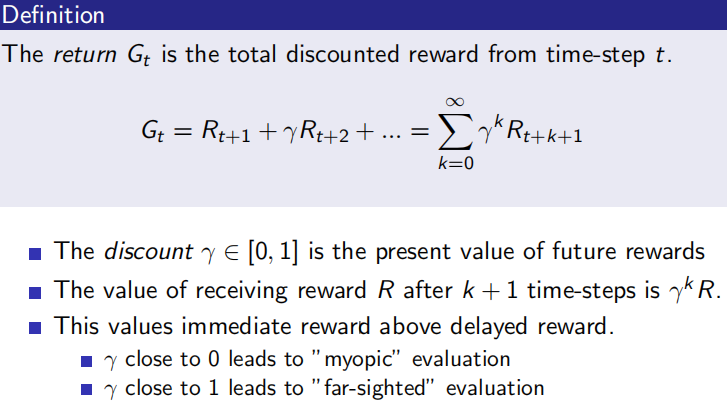
### 2.2.1 definition

给马尔科夫过程加上状态转移对应的奖励值。这样将马尔科夫过程扩充为四元组(S,P,R,γ)，其中R是即时奖励，γ为discount rate。



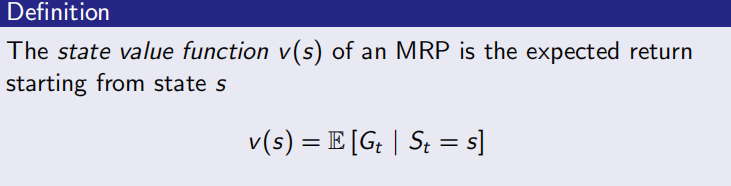
### 2.2.2 Gt

对于一个trajectory而言，获得的奖励G（对应李宏毅讲解的Advantage function Atθ），就是：



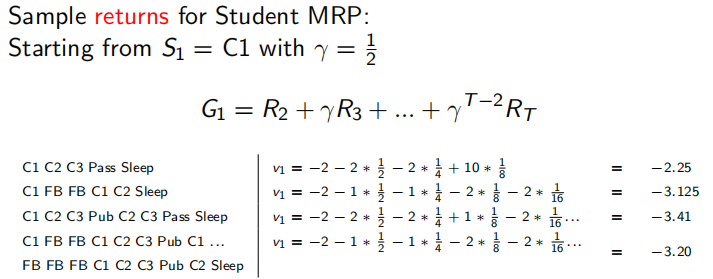
**NOTE:此时的Rt并不是期望值。为什么采用折现因子呢？主要是简化循环和对于未来不确定性的考虑。**

### 2.2.3 Value function



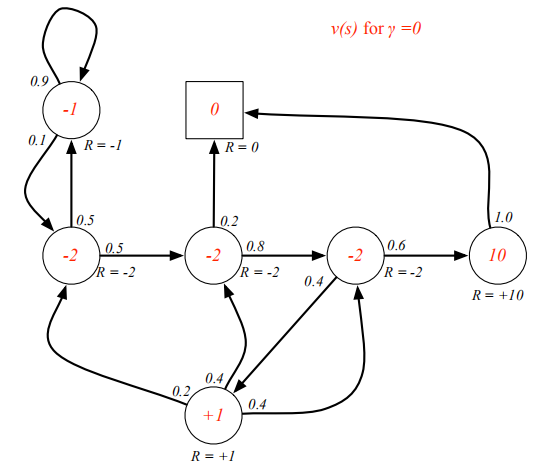
可以看出，无论是G还是V都是指从当前开始计算的奖励值，区别在于G是每一笔数据的值，而V是一个期望值。

**G以及V的计算：**

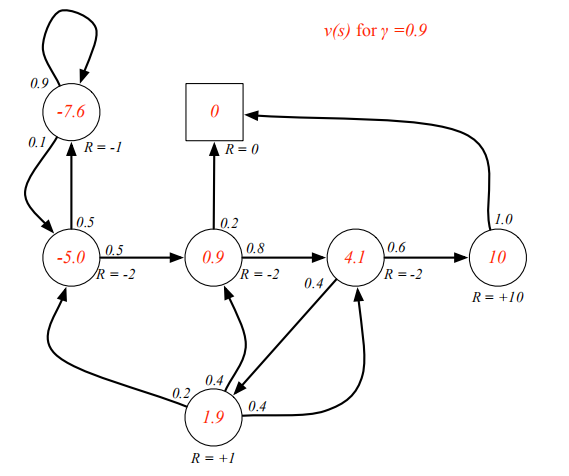


上图计算了以C1作为起点的每一笔实验数据的奖励值的计算。G

下面考虑value-function：



当γ为0，表示计算奖励值时只考虑当前action带来的奖励值，既不看过去，也不管将来，所以如果上C1，奖励值就是-2，去酒吧奖励值为+1，参加考试并通过就获得奖励+10。



当γ=0.9时，我们复盘它的计算过程：

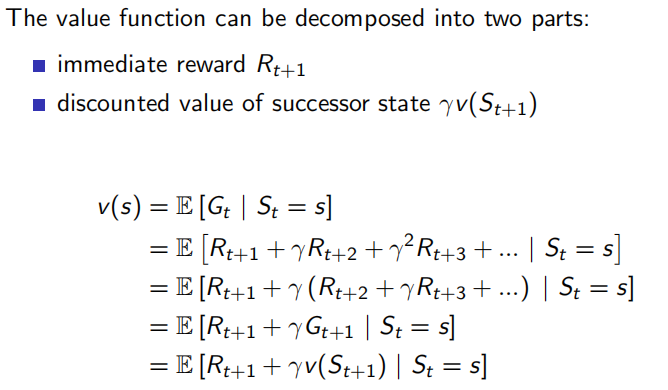
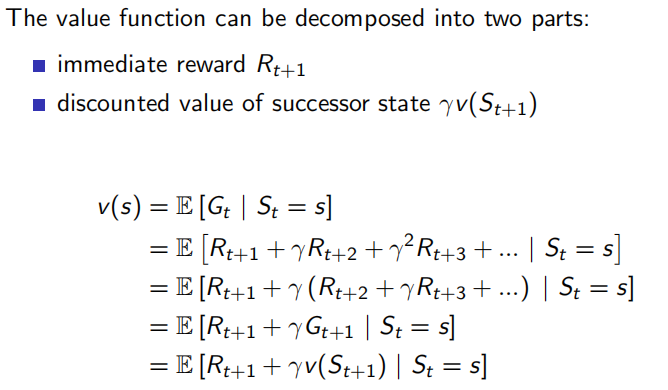
1. 采样大批数据，基于γ=0.9根据G的公式，计算单笔奖励值。G1,G2......
2. 将G数据带入到V的公式计算Value值。

**问题：0.9和4.1直接具有计算关系吗？**

**答：下面的Bellman等式就是这个关系**

**0.9 = -2 + 0.9 \* (0.8 \* 4.1 + 0.2 \* 0) 但是显然这组数据有误差。**

### 2.2.4 Bellman Equation



其实没有必要对V加期望，因为本身就是G的期望值。关于记法的说明：

第一种：s，a，||r，s，a，||r，s，a，||r......

第二种：s，a，r，||s，a，r，||s，a，r......

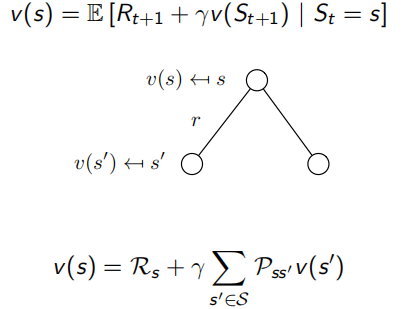
第一种记法认为采取action后环境变了，就进入到下一个时间步，之后的所有参数的下标都为t+1。此处的ppt采用第一种记法。有时候David会混用，比如下面。总之理解为即时奖励即可。

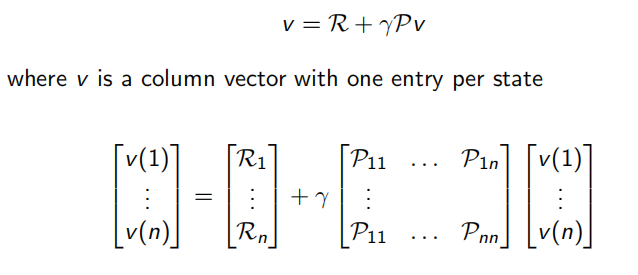
**问题**：假如同一个序列中，不同的时间步对应的同一个S，那么V相等吗？

**答**：后面的课程会讲到相关的知识，例如采用蒙特卡洛方法，会有两种策略：

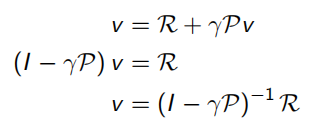
1. first sight：只考虑第一次看到，不存在不同时间问题；
2. every sight：会考虑不同的时间步然后做期望。

简化写法：





转移矩阵可能有很多0，因为是线型方程组，解为：

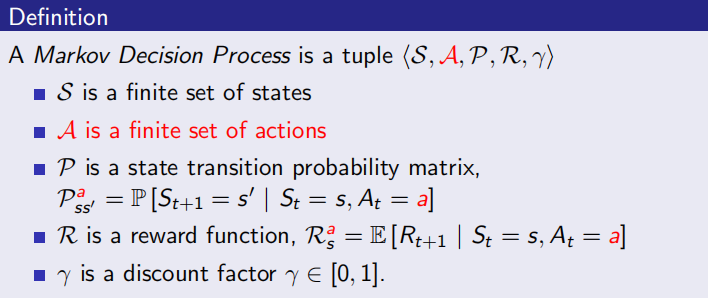


也就是说，V可以完全由γ，转移矩阵p和不同状态对应的即时奖励期望值R所确定。

## 2.3 Markov decision process(MDP)

### 2.3.1 Definition

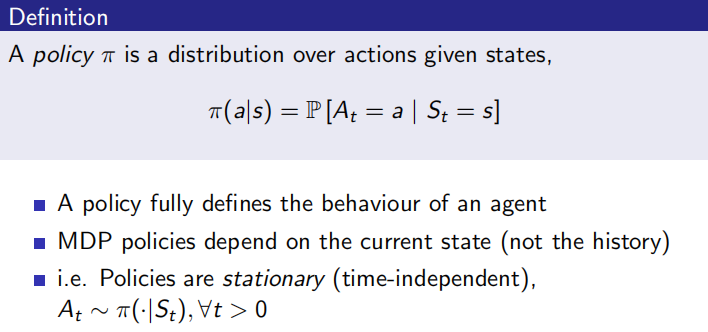
条件：环境中所有的state都具有马尔科夫属性。定义为：



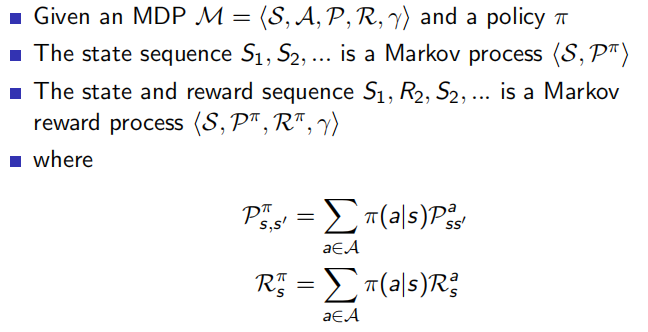
需要注意的是，此时转移矩阵和奖励值是计算都加上了action条件。

### 2.3.2 Policy

是state到采取所有可能action的概率的映射。

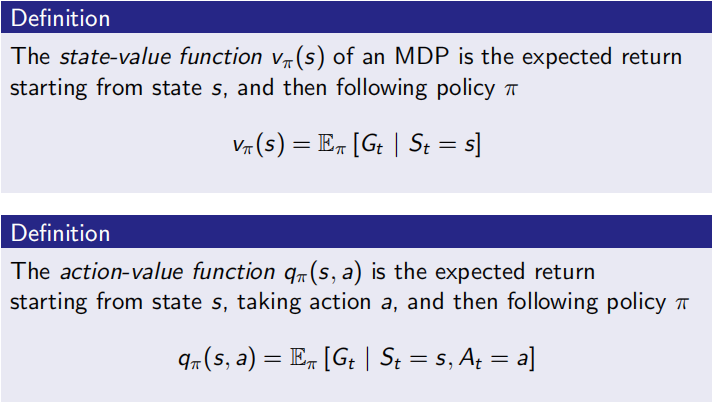


可以看出，MDP policy的输出只取决于当前的state，而与历史无关。而且具有**时间稳定性:同一种状态即便出现在不同的时间点，其输出是一样的。**



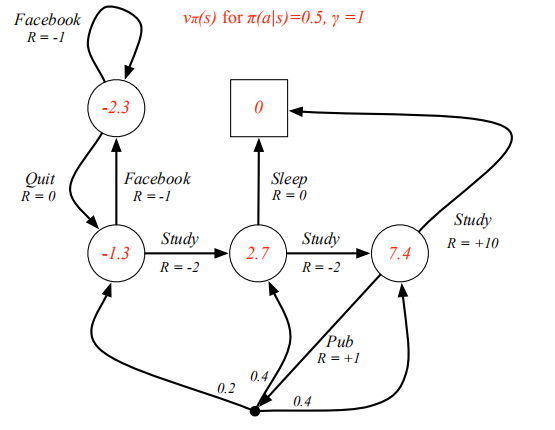
假如，我们的模型符合MDP，那么其输出的状态序列(S,R) tuple都具有Markov属性。其计算过程需要用到概率论中的加法公式。

### 2.3.3 Value function



有两种Value function：分别为对state的评估和对给定state下action的评估。

举例而言：



计算的过程中只要考虑出度：

**-1.3 = [(-2.3-1)+(2.7-2)]/2**

**2.7 = [(7.4-2)+0]/2**

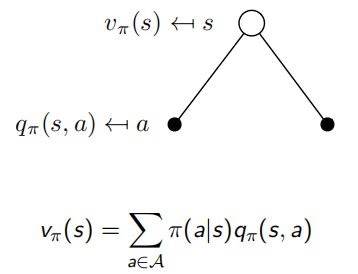
### 2.3.4 Bellman Expectation Equation



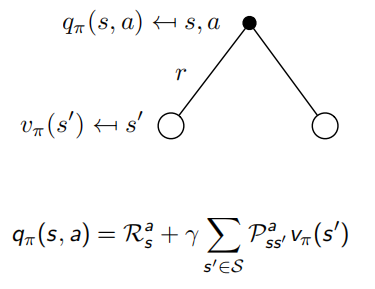


Note:不管是V还是Q，本身都是G的期望，前面为什么还要加上E呢？

接下来，在两类Value function中构建关系：

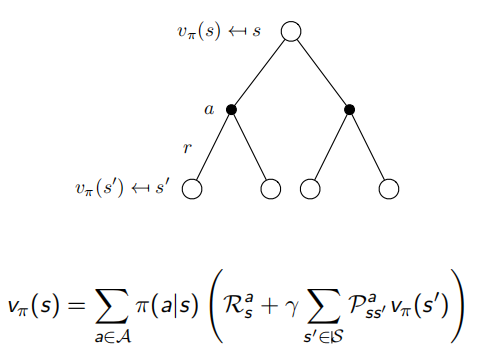


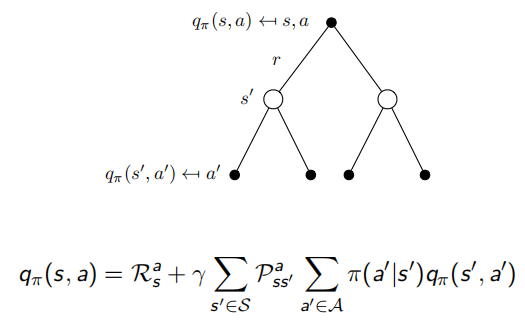
解释：V是期望值，是Q的期望值，是s时采取不同action的期望值。



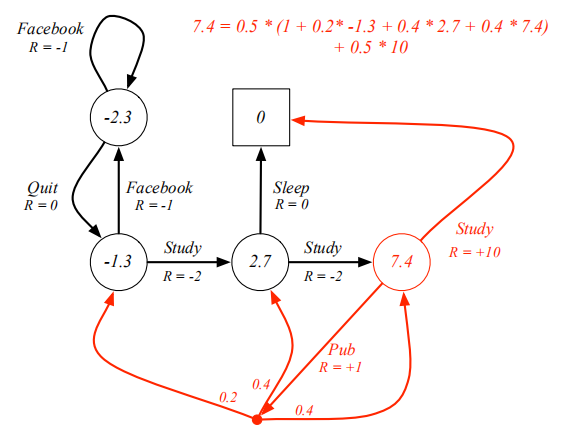
解释：Q是期望值，是采取action后进入不同state的期望值，是下一个时间步V的期望值乘以折损率 + 即时奖励

将V和Q互相代入，有：

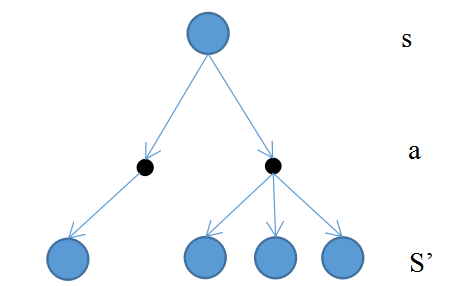




以上两幅图：在state确定是时，采取同一个action，可能进入不同的状态。



Note:此时的去酒吧是一个action而不是一个状态，去过酒吧之后可能会进入3中不同的状态，对应的模型图应该为：



### 2.3.5 Bellman Equation 矩阵形式

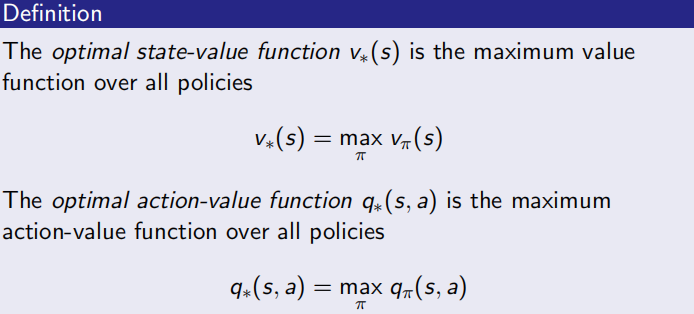
问题是转移矩阵的概率和policy的概率可以统一吗？可以的，总和为Pπ



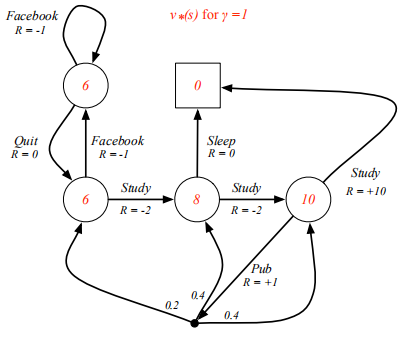


### 2.3.6 Optimal Value Function

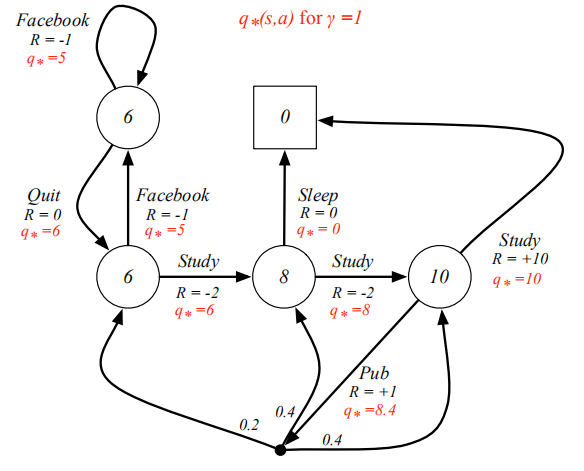
强化学习可以理解为找到最优policy的过程，因为基于此可以获得奖励的最大化，那么首先要对optimal形式化的定义如下：



当真的找到optimal function的时候，我么可以说，已经解决了MDP问题。就像回到学生MDP的例子。

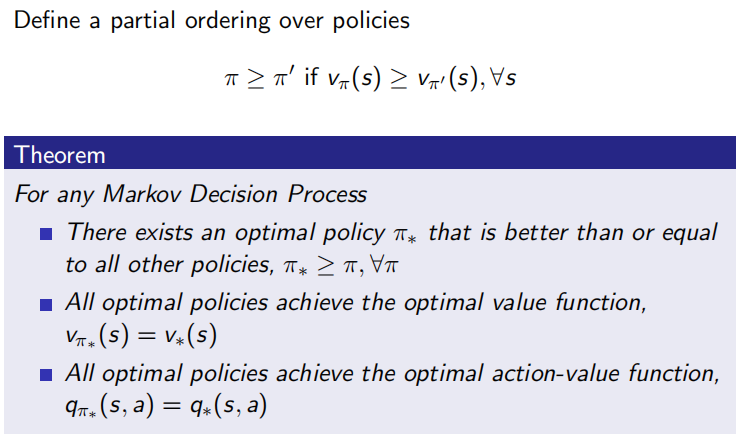


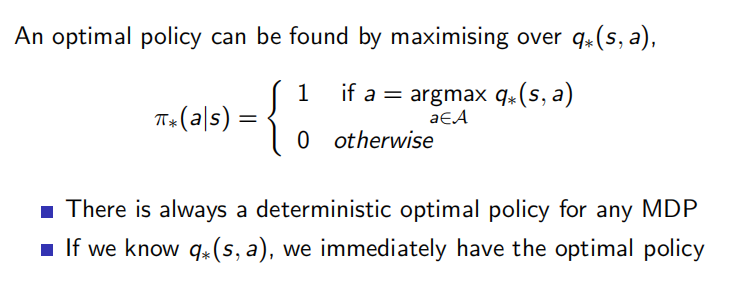
最佳的Value Function示意（即时奖励+下一状态的V），例如在C1可以是5或者6，单取一个最大值，所以是6.



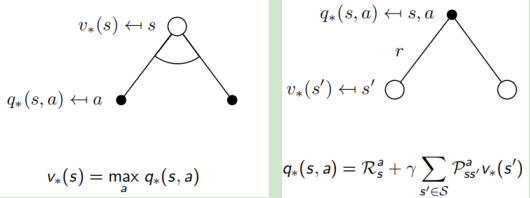
最佳的Q\_function示意图

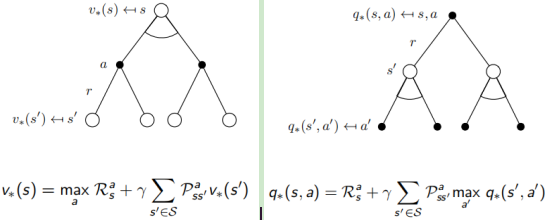
### 2.3.7 Optimal Policy

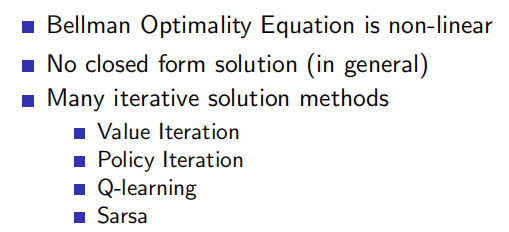
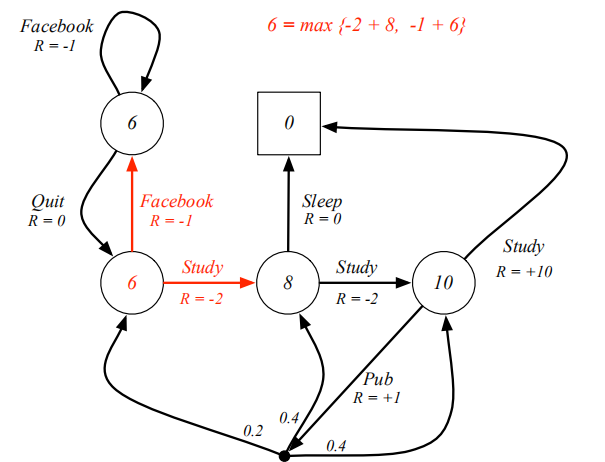




**最核心的思想**就是，选action时，考虑奖励最大化。评估state时，求不同情况的均值。

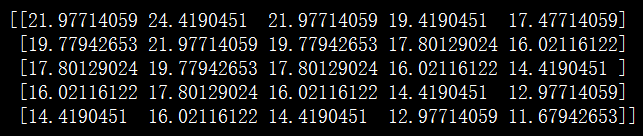


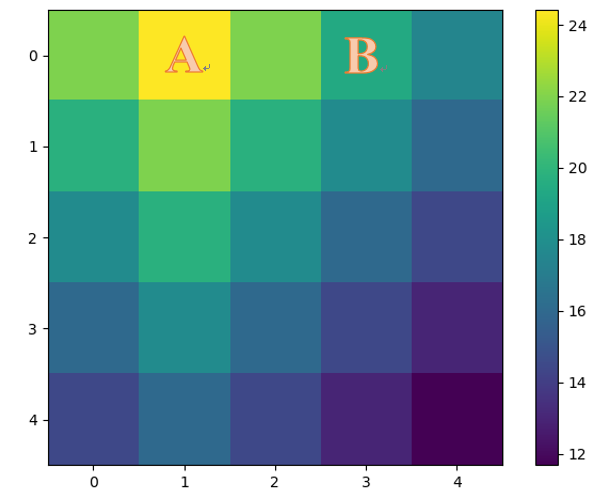




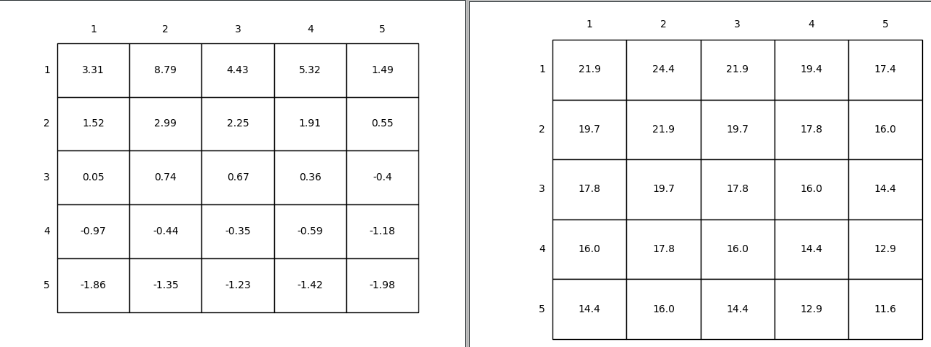
网格世界的例子：

全0初始化，迭代103轮之后：





还有一个很重要的信息就是：



由于采用了optimal value function，使得AB之间的网格的value介于AB之间，从未避免了B陷入局部最优解。至此，只想说一句optimal真香。但是，没完：因为这种optimal value function的直接解法本质上是一种暴力搜索：必须满足三个条件：

1、对环境的完全理解（例如位置的奖励等）

2、足够的算力

3、MDP属性

很难用于实践的瓶颈在于第二点，实践中往往是APPROXIMATION的方法。