动态规划算法实验报告

吴孟周 2100013053

## 一、实验要求

（1）阅读《基于动态规划的路径规划-实验指导书》，尝试运行并理解动态规划算法在冰湖路径规划问题上的示例代码。

（2）在示例代码的基础上进行修改，至少实现一种异步动态规划算法，并与原算法比较策略收敛所需的迭代次数。

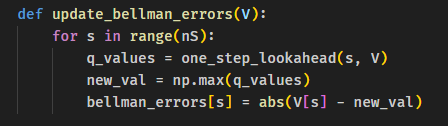
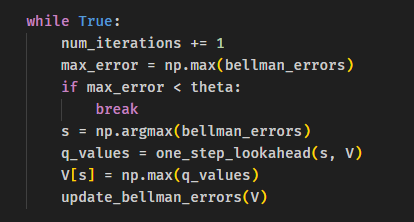
## 二、代码修改思路

在value\_iteration的基础上，加入优先级的考虑，做优先级动态规划，实现了value\_iteration\_priority。在优先级动态规划中，我们使用贝尔曼误差来评估状态的优先级，贝尔曼误差越大的越先更新，以加快收敛速度。

修改代码遇到的挑战有两个，一是如何快速找到最需要更新的状态，二是在更新状态s之后，s和s的前驱状态的贝尔曼误差会发生改变，如何快速做出这个改变。

一般的解决方法是使用优先队列，这样就可以在O(logn)的时间代价和O(n)的空间代价内进行修改和弹出最大值的操作。但同时也有一个问题，在状态s进行修改之后需要修改s所有前驱的贝尔曼误差，重新在优先队列中更新，这在状态转移图较稠密时的时间代价是巨大的。

本次实验的规模很小，且主要任务在于实现一个异步动态规划算法并进行分析，而不是追求极致的性能。因此我在本次实验中通过数组简单的模拟了优先队列的行为，用一个数组表示每个状态的贝尔曼误差。之后不断迭代，每次取出数组的argmax状态并更新，直到数组的最大值小于设定的阈值 theta。



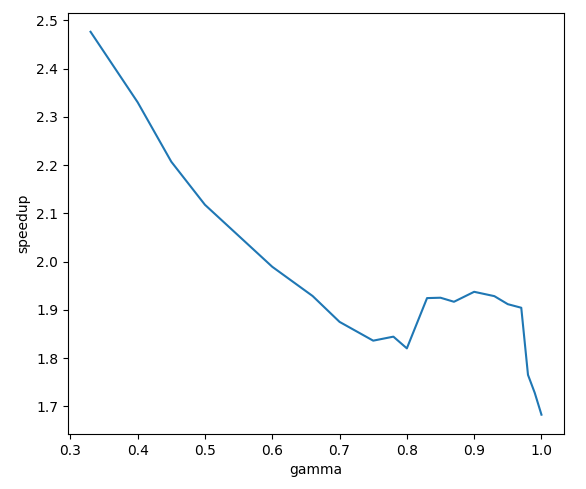
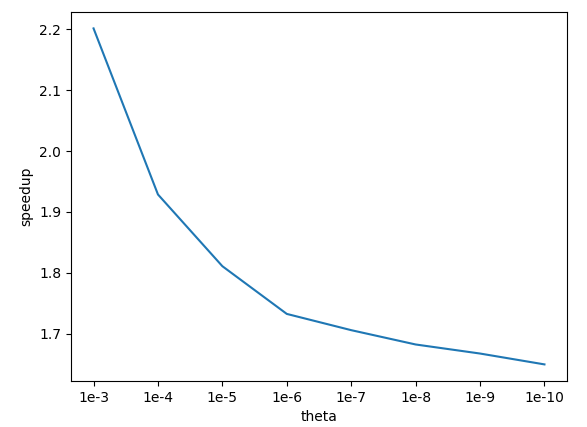
## 三、实验结果和分析

对于value\_iteration\_priority算法，我们使用迭代的状态数作为迭代次数。

对于value\_iteration算法，我们使用迭代轮数\*16（这是本次实验的状态数）作为迭代次数。

记speedup表示加速比，即value\_iteration迭代次数除以value\_iteration\_priority迭代次数。

在正常的参数下，speedup约为2。这说明了使用优先级做value\_iteration有一定的提升。但因为本次实验的模型是完全已知的，所以状态转移的效率是很高的，与其花费时间在维护优先队列上（造成了远大于2倍的时间浪费），不如增加迭代的次数，这导致了在运行时间指标上使用优先级一定是负优化。

下面是分别调整两个参数theta和gamma的实验结果。

在调整参数theta时，我们将gamma固定为1。注意到随着theta降低（即要求收敛得更彻底），speedup下降。这是符合直观的，使用优先级的value\_iteration更适合处理具有特殊性质的问题，而不是一般性的问题。当theta降低时，更需要的是长期的迭代，实验场景变得更加一般化。

在调整参数gamma时，我们将theta固定为1e-8。注意到随着gamma升高，speedup首先下降，但在0.9附近短暂爬升，在接近1时再次下降。

* 整体的下降趋势是容易解释的，在gamma更大时，agent更关注长期的收益，因此问题是更加一般化的。在gamma较小时，agent基本只需关注几步内的收益，因此问题具有一些特殊的性质，适合使用优先级优先关注特殊的状态。
* 对于0.9附近出现的爬升，一个可能的解释是：考虑本次实验场景，在gamma很大时agent一定倾向于选择尽可能安全的策略以到达终点，这导致了问题较为一般性。在gamma取0.9附近时，agent除了最安全的策略，还有一种策略是冒险采取最短路，这样可以更快的到达终点以获得更多的奖励。这两种不同的策略导致环境出现了更多的特殊性质，而优先级动态规划可以更多的关注到这些特殊性质所对应的状态，因此speedup出现了一段提升。