

高级算法

Advanced Topics in Algorithms

陈旭

数据科学与计算机学院



中山大學
SUN YAT-SEN UNIVERSITY

Policy Iteration for Optimal Policies

Policy evaluation on MDP

- ① Objective: Evaluate a given policy π for a MDP
- ② Output: the value function under policy v^π
- ③ Solution: iteration on Bellman expectation backup
- ④ Algorithm: Synchronous backup
 - ① At each iteration $t+1$
update $v_{t+1}(s)$ from $v_t(s')$ for all states $s \in \mathcal{S}$ where s' is a successor state of s

$$v_{t+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) (R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) v_t(s'))$$

- ⑤ Convergence: $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v^\pi$

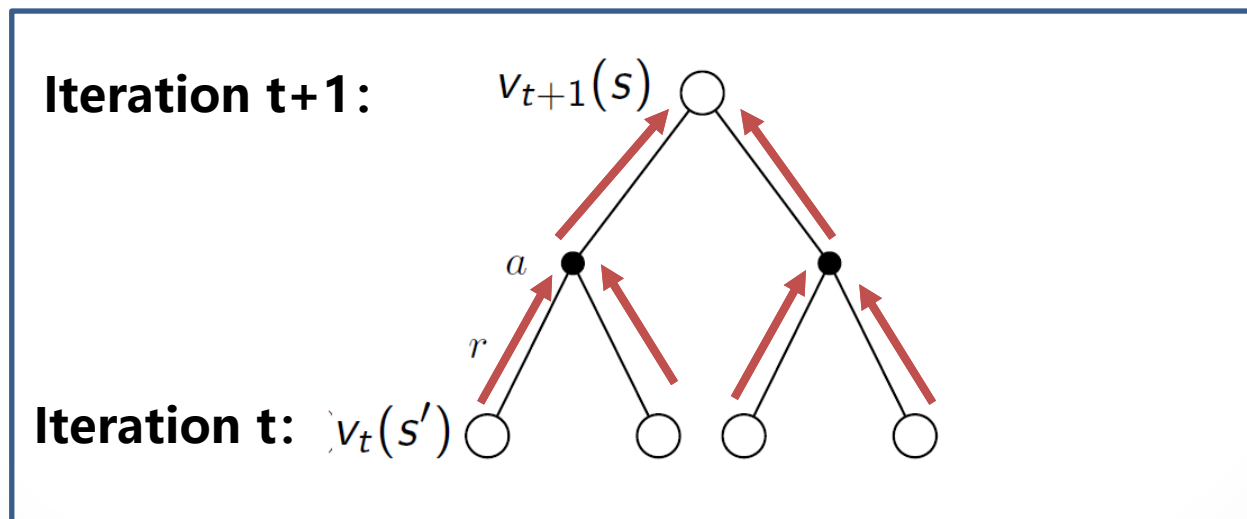
Policy evaluation: Iteration on Bellman expectation backup

Bellman expectation backup for a particular policy

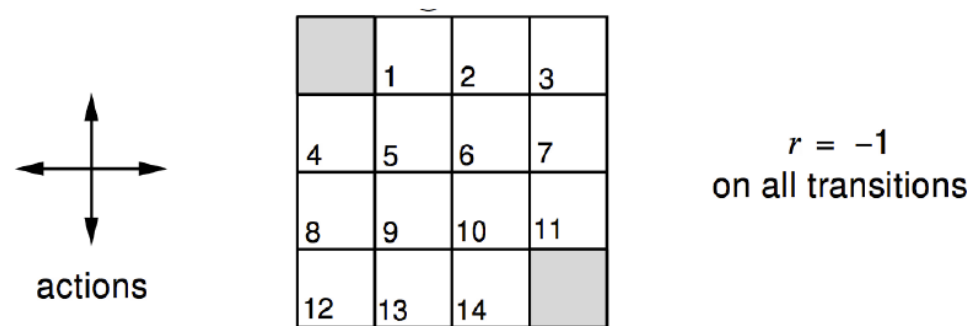
$$v_{t+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) (R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) v_t(s'))$$

Or if in the form of MRP $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}, \gamma \rangle$

$$v_{t+1}(s) = R^\pi(s) + \gamma P^\pi(s'|s) v_t(s')$$



Evaluating a Random Policy in the Small Gridworld



- Undiscounted episodic MDP ($\gamma = 1$)
- Nonterminal states $1, \dots, 14$
- One terminal state (shown twice as shaded squares)
- Actions leading out of the grid leave state unchanged
- Reward is -1 until the terminal state is reached
- Agent follows uniform random policy

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

Iterative Policy Evaluation in Small Gridworld

U_k for the
Random Policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

How to Improve a Policy

- Given a policy π
 - **Evaluate** the policy π

$$v_{\pi}(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s]$$

- **Improve** the policy by acting greedily with respect to v_{π}

$$\pi' = \text{greedy}(v_{\pi})$$

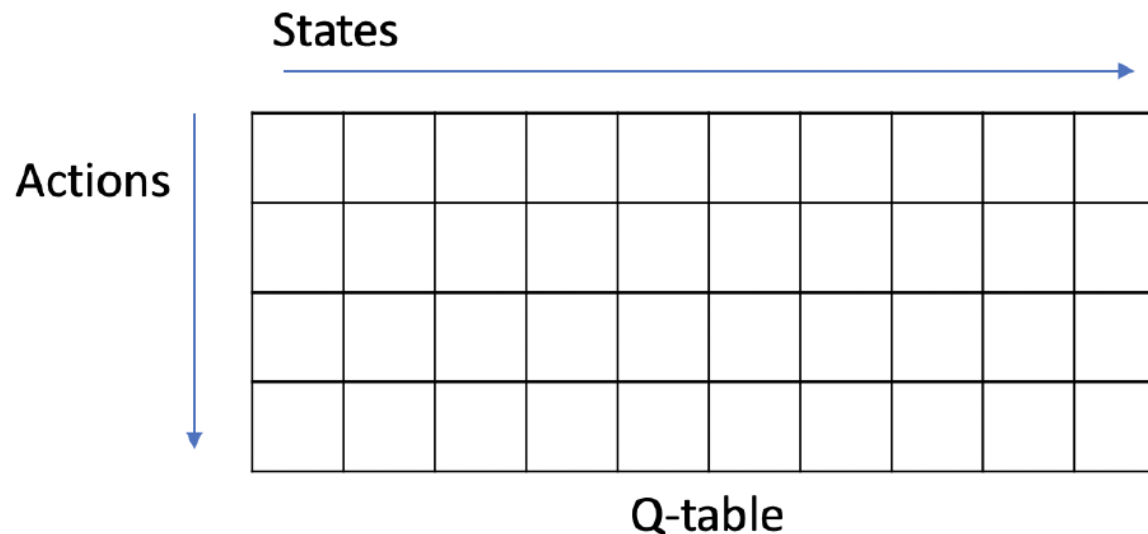
Policy Improvement

- 1 Compute the state-action value of a policy π :

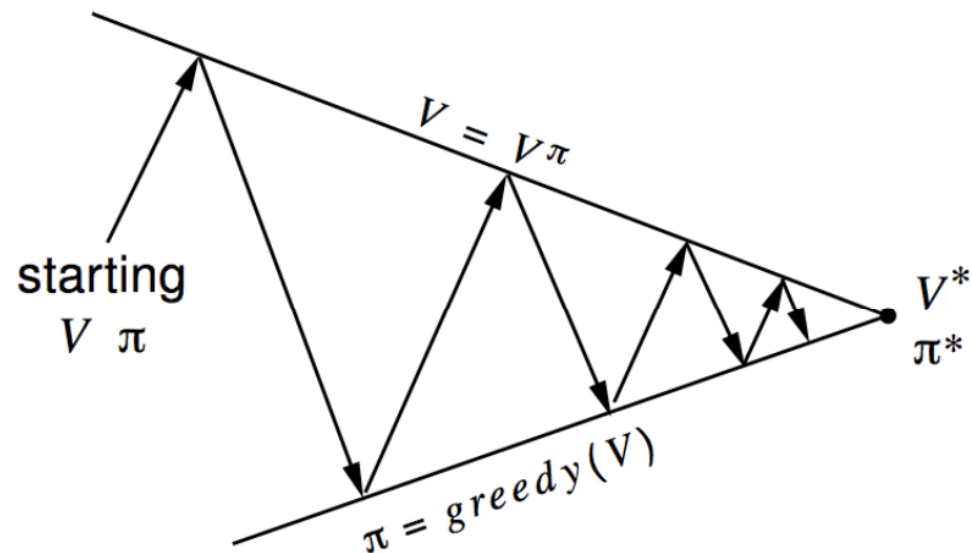
$$q^{\pi_i}(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) v^{\pi_i}(s')$$

- 2 Compute new policy π_{i+1} for all $s \in \mathcal{S}$ following

$$\pi_{i+1}(s) = \arg \max_a q^{\pi_i}(s, a)$$

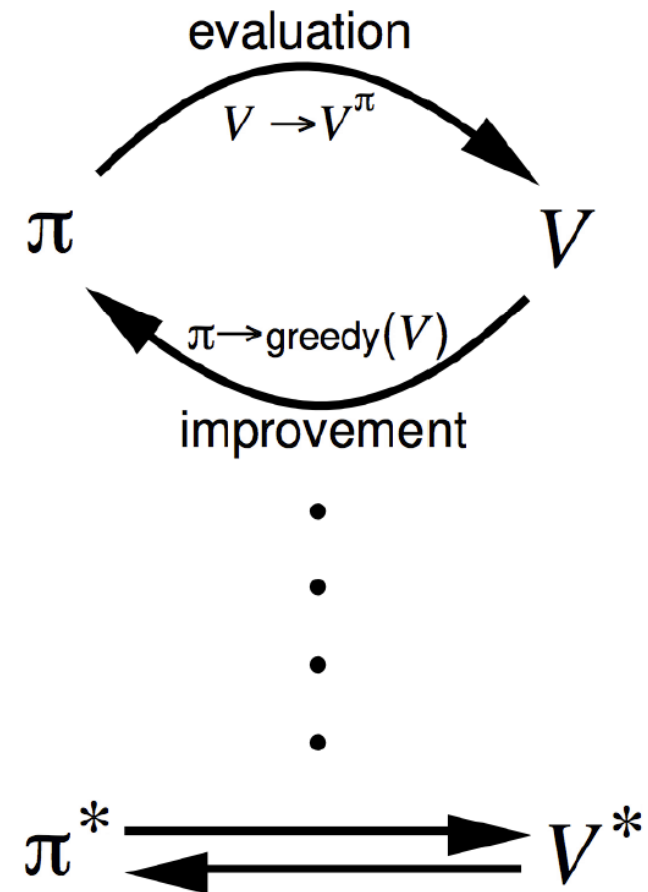


Policy Iteration

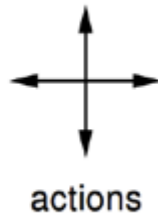


Policy evaluation Estimate v_π
 Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
 Greedy policy improvement



Small Gridworld



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$r = -1$
on all transitions

Policy Evaluation of Random Policy

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Policy Update



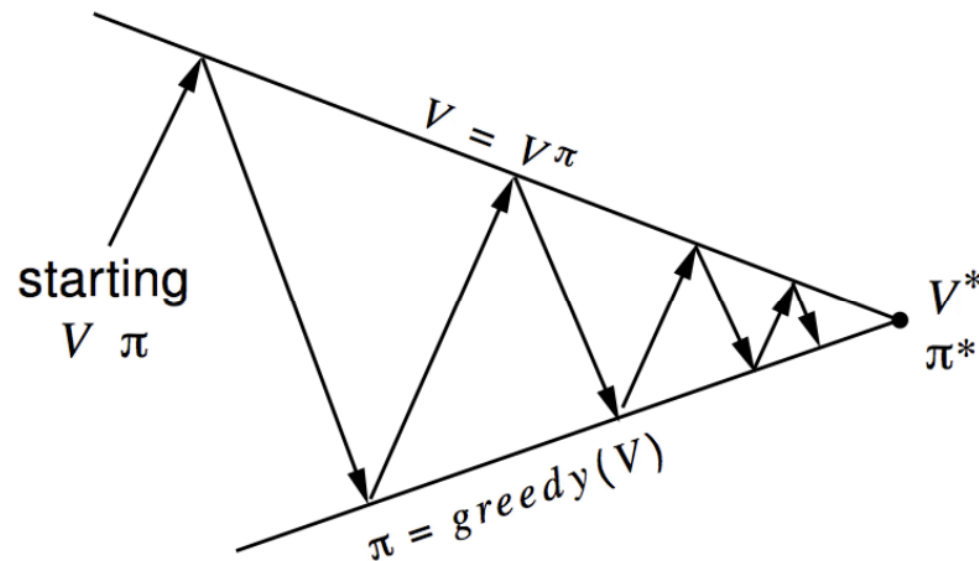
Optimal Policy!

	←	←	↙
↑	↖	↙	↓
↑	↖	↘	↓
↖	→	→	

Modified Policy Iteration

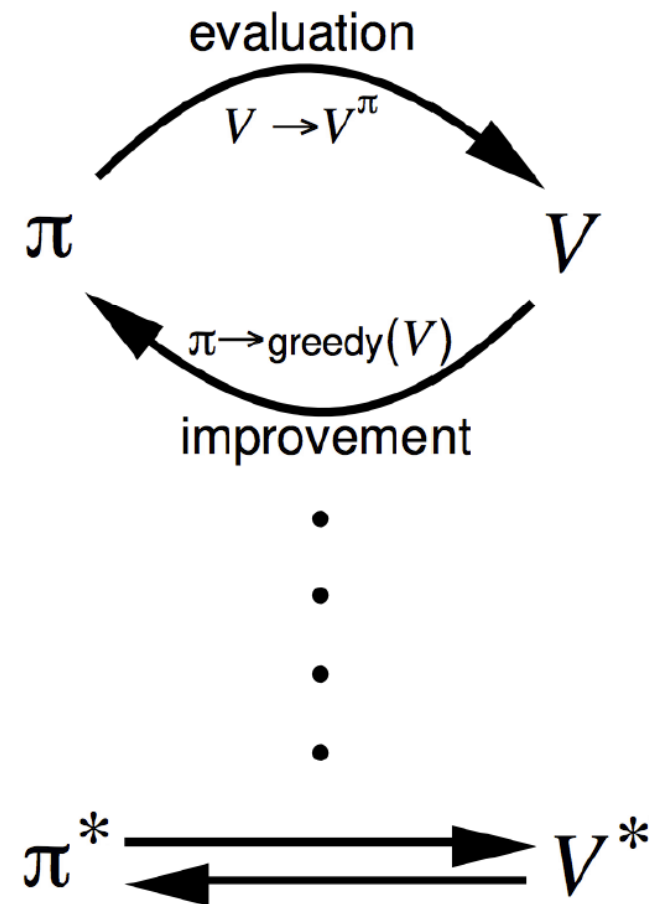
- Does policy evaluation need to converge to v_π ?
- Or should we introduce a stopping condition
 - e.g. ϵ -convergence of value function
- Or simply stop after k iterations of iterative policy evaluation?

Generalised Policy Iteration

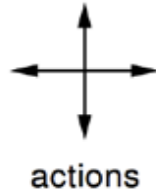


Policy evaluation Estimate v_π
 Any policy evaluation algorithm

Policy improvement Generate $\pi' \geq \pi$
 Any policy improvement algorithm



Small Gridworld



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$r = -1$
on all transitions

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

	←	←	↙
↑	↖	↙	↓
↑	↖	↘	↓
↙	→	→	

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

	←	←	↙
↑	↖	↙	↓
↑	↖	↘	↓
↙	→	→	

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

	←	←	↙
↑	↖	↙	↓
↑	↖	↘	↓
↙	→	→	

optimal
policy

Value Iteration for Optimal Policies

Bellman Optimality Equation

- 1 The optimal value functions are reached by the Bellman optimality equations:

$$v^*(s) = \max_a q^*(s, a)$$

$$q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) v^*(s')$$

thus

$$v^*(s) = \max_a R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) v^*(s')$$

$$q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) \max_{a'} q^*(s', a')$$

Value Iteration by turning the Bellman Optimality Equation as update rule

- 1 If we know the solution to subproblem $v^*(s')$, which is optimal.
- 2 Then the solution for the optimal $v^*(s)$ can be found by iteration over the following Bellman Optimality backup rule,

$$v(s) \leftarrow \max_{a \in \mathcal{A}} \left(R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) v(s') \right)$$

- 3 The idea of value iteration is to apply these updates iteratively

Algorithm of Value Iteration

- ① Objective: find the optimal policy π
- ② Solution: iteration on the Bellman optimality backup
- ③ Value Iteration algorithm:

- ① initialize $k = 1$ and $v_0(s) = 0$ for all states s
- ② For $k = 1 : H$
 - ① for each state s

$$q_{k+1}(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) v_k(s')$$

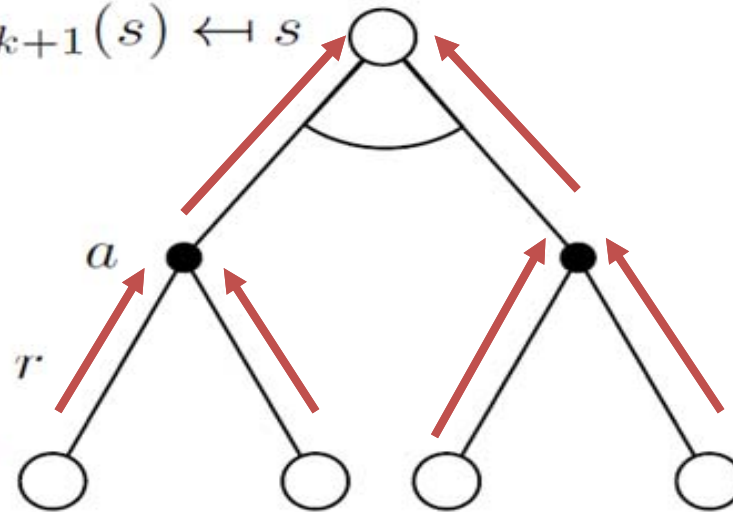
$$v_{k+1}(s) = \max_a q_{k+1}(s, a)$$

- ② $k \leftarrow k + 1$
- ③ To retrieve the optimal policy after the value iteration:

$$\pi(s) = \arg \max_a R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) v_{k+1}(s')$$

Or equivalently, $\pi(s) = \arg \max_a q_{k+1}(s, a)$

Iteration k+1: $v_{k+1}(s) \leftarrow s$



Iteration k: $v_k(s') \leftarrow s'$

$$v_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right)$$

Example: Shortest Path

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

After the optimal values are reached, we run policy extraction to retrieve the optimal policy.

Difference between Policy Iteration and Value Iteration

- ① Policy iteration includes: **policy evaluation** + **policy improvement**, and the two are repeated iteratively until policy converges.
- ② Value iteration includes: **finding optimal value function** + **one policy extraction**. There is no repeat of the two because once the value function is optimal, then the policy out of it should also be optimal (i.e. converged).

Decision Making in Markov Decision Process (MDP)

- ① Prediction (evaluate a given policy):
 - ① Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and policy π or MRP $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$
 - ② Output: value function v^π
- ② Control (search the optimal policy):
 - ① Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$
 - ② Output: optimal value function v^* and optimal policy π^*

Summary for Prediction and Control in MDP

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

Demo of policy iteration and value iteration



- 1 Policy iteration: Iteration of policy evaluation and policy improvement(update)
- 2 Value iteration

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html































































































GridWorld: Dynamic Programming Demo

Policy Evaluation (one sweep)

Policy Update

Toggle Value Iteration

Reset

0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 
0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 
0.00 					0.00 				0.00 
0.00 	0.00 	0.00 	0.00 		0.00 	0.00 	0.00 	0.00 	0.00 
0.00 	0.00 	0.00 	0.00 		0.00 	0.00 	0.00 	0.00 	0.00 
0.00 	0.00 	0.00 	0.00 		0.00 	0.00 	0.00 	0.00 	0.00 
0.00 	0.00 	0.00 	0.00 		0.00 	0.00 	0.00 	0.00 	0.00 
0.00 	0.00 	0.00 	0.00 		0.00 	0.00 	0.00 	0.00 	0.00 
0.00 	0.00 	0.00 	0.00 		0.00 	0.00 	0.00 	0.00 	0.00 
0.00 	0.00 	0.00 	0.00 		0.00 	0.00 	0.00 	0.00 	0.00 
0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 	0.00 

Assignments

- *P61, Exercise 3.6*
- *P64, Exercise 3.7*

- *提交邮箱: gjsf_2020@126.com*
- *邮件+文件命名: 学号+姓名*
- *提交期限: 下周四23:59*