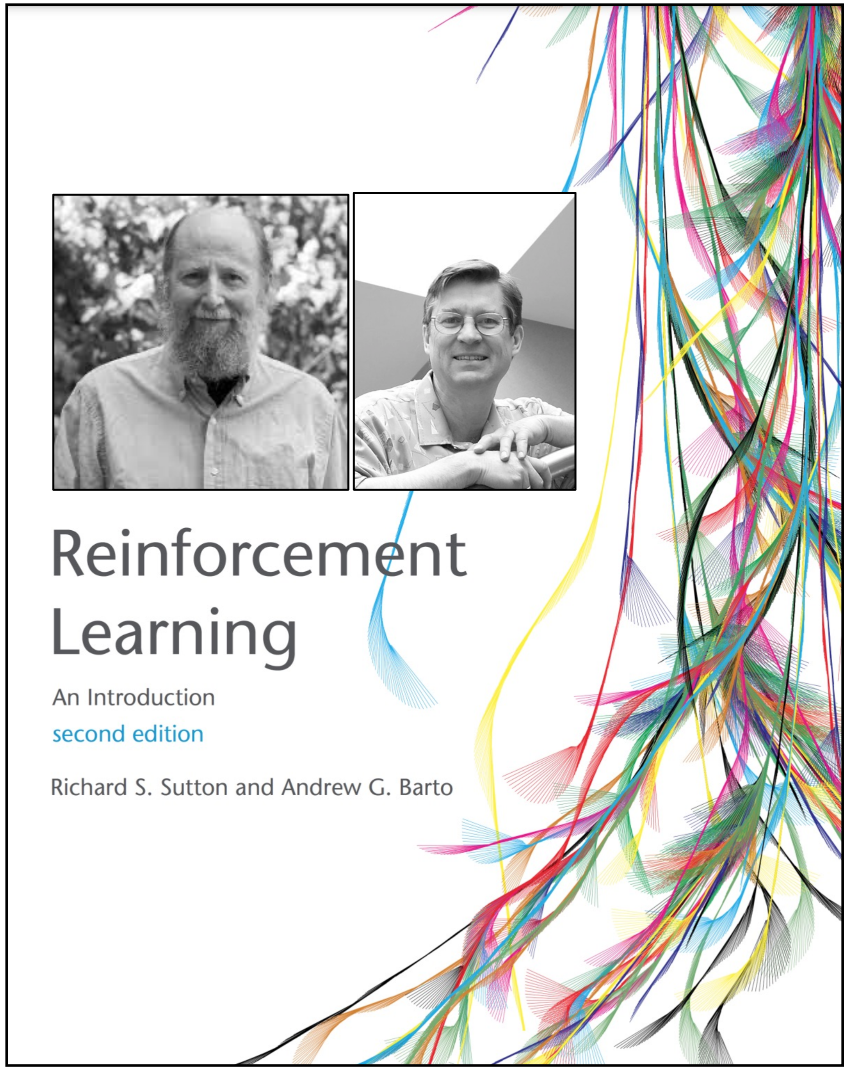
Session 1

Reinforcement Learning

KAUSHIK LAKSHMINARASIMHAN

Bibliography



02/22: Chapters 3-4

- Markov Decision Process
- Model-based Methods (*planning*)

03/01: Chapters 5-8

- Model-Free Methods (*learning*)
- Links to Neuroscience

Group discussion (5 mins)

What is the maximum total from top to bottom? You can only pick one number from each row by moving to one of the adjacent numbers on the row below like so.

```
                            75
                          95 64
                        17 47 82
                      18 35 87 10
                    20 04 82 47 65
                  19 01 23 75 03 34
                88 02 77 73 07 63 67
              99 65 04 28 06 16 70 92
            41 41 26 56 83 40 80 70 33
          41 48 72 33 47 32 37 16 94 29
        53 71 44 65 25 43 91 52 97 51 14
      70 11 33 28 77 73 17 78 39 68 17 57
    91 71 52 38 17 14 91 43 58 50 27 29 48
  63 66 04 68 89 53 67 30 73 16 69 87 40 31
04 62 98 27 23 09 70 98 73 93 38 53 60 04 23
```

Objective of reinforcement learning

To maximize expected *return*:

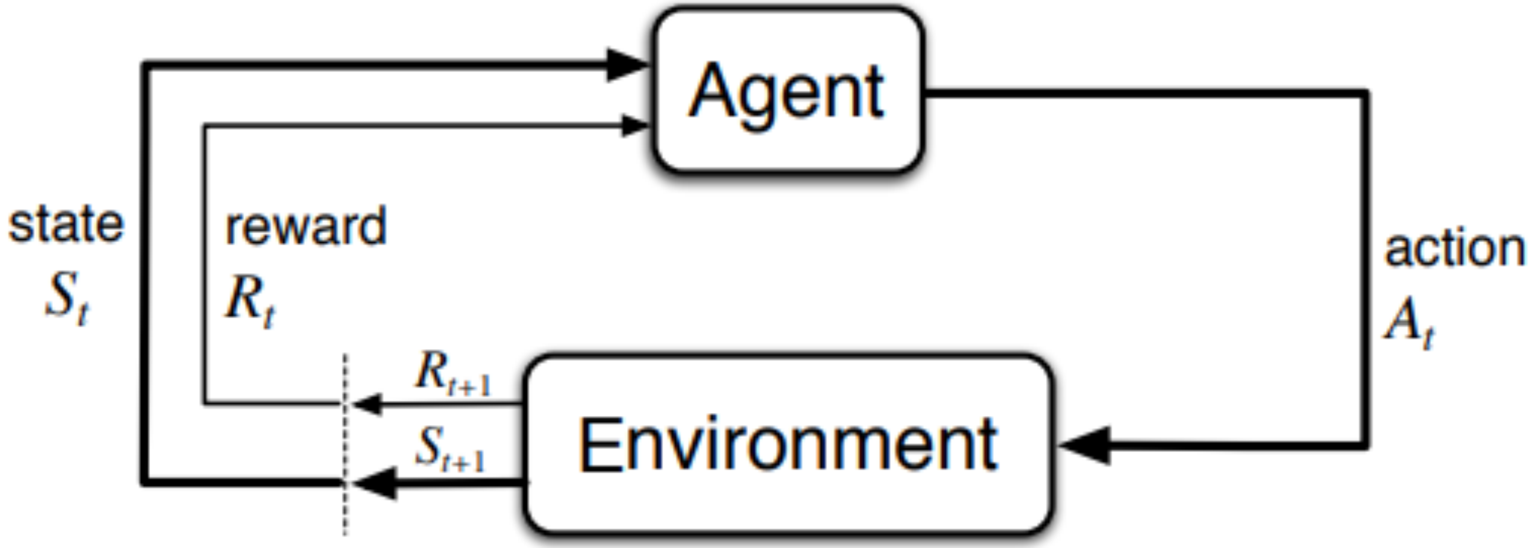
$$\max \quad G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

where $0 \leq \gamma \leq 1$ is a parameter, called the discount factor

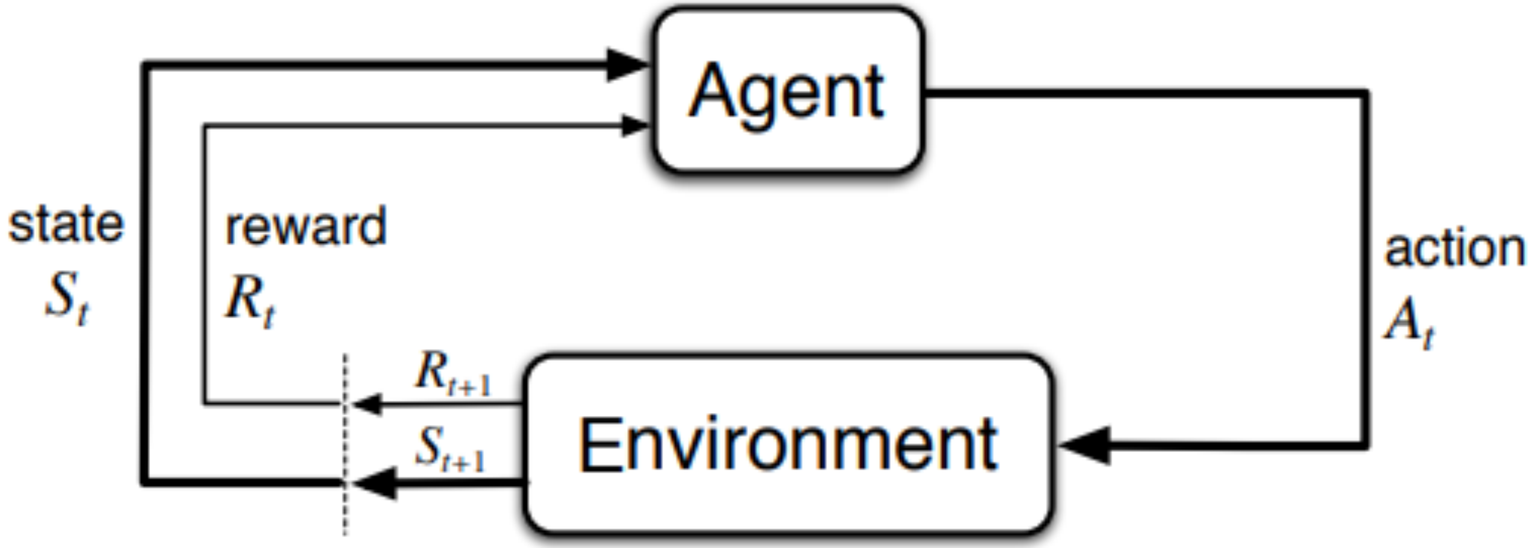
Special case ($\gamma = 1$): maximize expected *cumulative reward*:

$$\max \quad \sum_{k=0}^{\infty} R_{t+k+1}$$

Markov Decision Process



Markov Decision Process



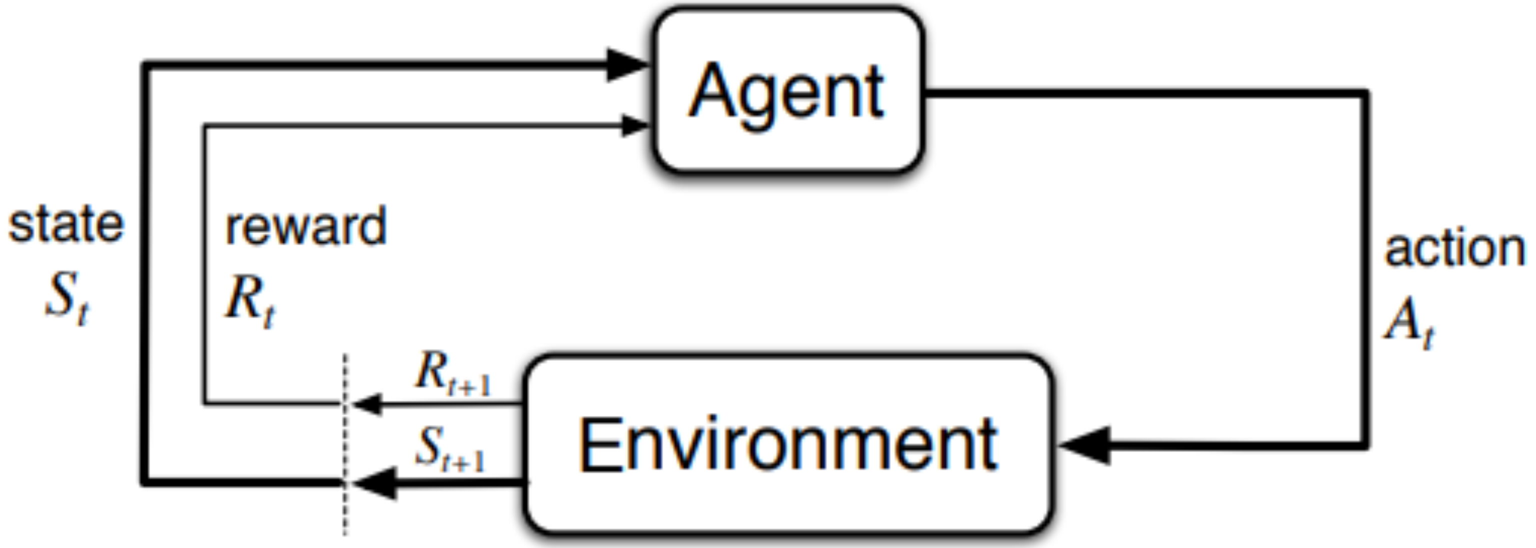
Markov property

$$p(s', r | s, a) \doteq \Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\},$$

for all $s \in \mathcal{S},\ r \in \mathcal{R},\ a \in \mathcal{A}$

$p(s', r | s, a)$ is usually called the *model* of the world or the *dynamics* of the MDP

Markov Decision Process



Markov property

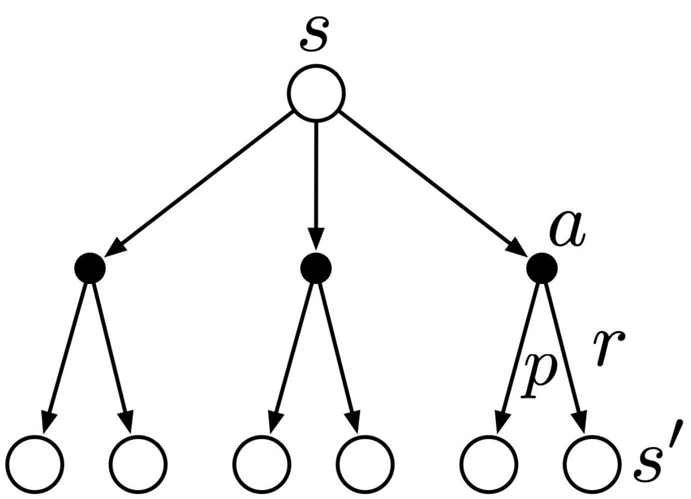
$$p(s', r | s, a) \doteq \Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\},$$

for all $s \in \mathcal{S},\ r \in \mathcal{R},\ a \in \mathcal{A}$

$p(s', r|s, a)$ is usually called the *model* of the world or the *dynamics* of the MDP

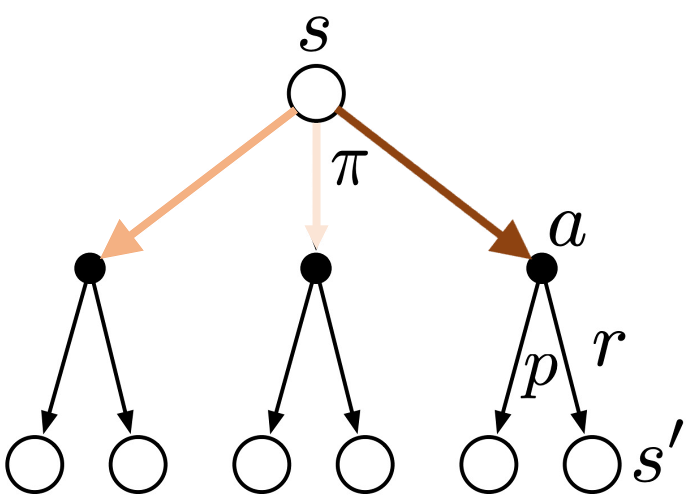
Note: Markov property is a restriction on the state, not on the decision process itself

Markov Decision Process



$p(s', r|s, a)$ is usually called the *model* of the world or the *dynamics* of the MDP

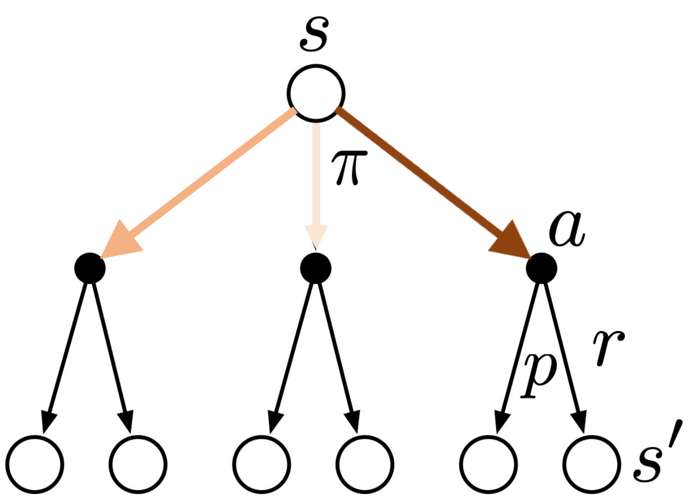
Policy and Value function



$$\pi(a|s) \doteq \Pr\{A_t = a | S_t = s\}$$

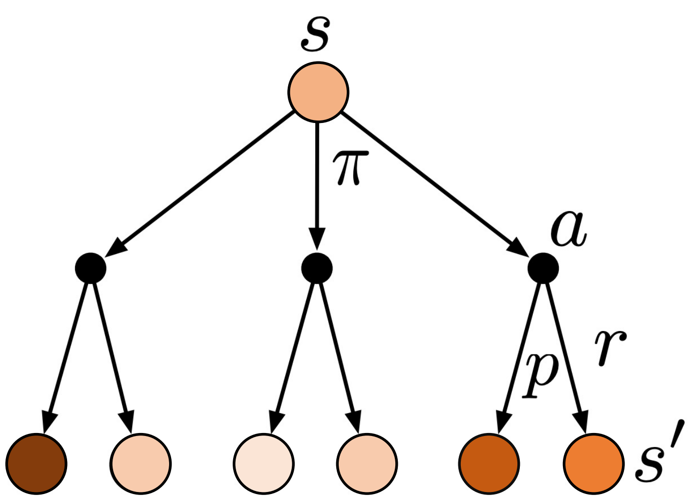
A *policy* is a mapping from states to the probabilities of selecting each possible action

Policy and Value function



$$\pi(a|s) \doteq \Pr\{A_t = a | S_t = s\}$$

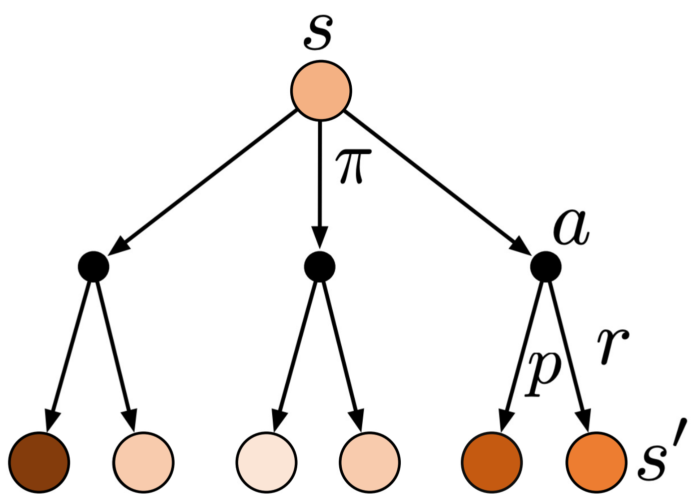
A *policy* is a mapping from states to the probabilities of selecting each possible action



$$v_\pi(s) \doteq \mathbb{E}[G_t | S_t = s]$$

State-value of a state s under a policy π is the expected return when starting from s and following π thereafter

Group discussion (10 mins)

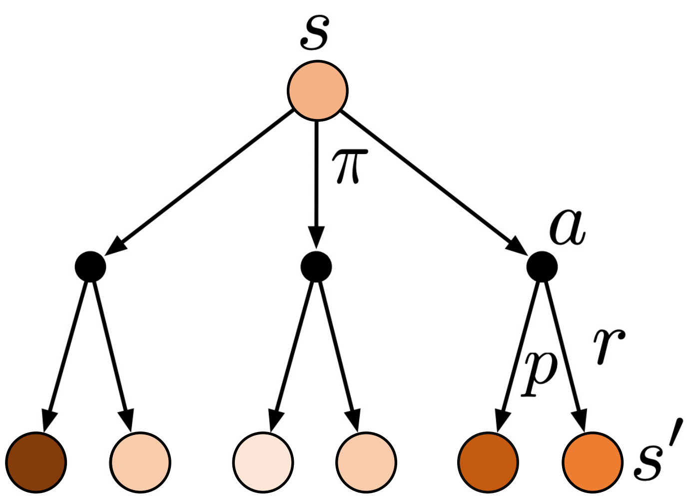


$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$$

Given policy π and dynamics p, derive a recursive relationship between the value of state s, $v_\pi(s)$, and the value at its possible successor state s', $v_\pi(s')$

Hint: $G_t = R_{t+1} + \gamma G_{t+1}$

Bellman Expectation Equation



$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$$

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$$

$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a)\left[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']\right]$$

$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a)[r + \gamma v_\pi(s')]$$

Group discussion (5 mins)

Use the bellman expectation equation to evaluate the top-most node of the triangle on the right under a random walk policy

75

95 64

17 47 82

18 35 87 10

20 04 82 47 65

19 01 23 75 03 34

88 02 77 73 07 63 67

99 65 04 28 06 16 70 92

41 41 26 56 83 40 80 70 33

41 48 72 33 47 32 37 16 94 29

53 71 44 65 25 43 91 52 97 51 14

70 11 33 28 77 73 17 78 39 68 17 57

91 71 52 38 17 14 91 43 58 50 27 29 48

63 66 04 68 89 53 67 30 73 16 69 87 40 31

04 62 98 27 23 09 70 98 73 93 38 53 60 04 23

75

95 64

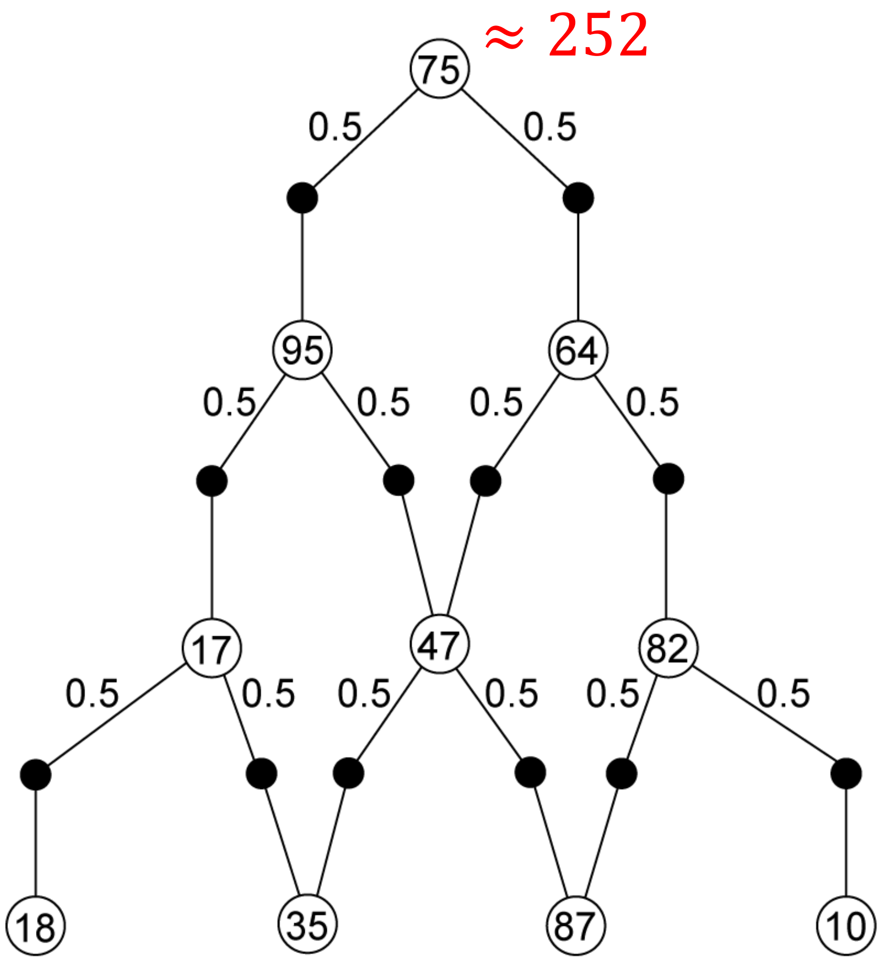
17 47 82

18 35 87 10

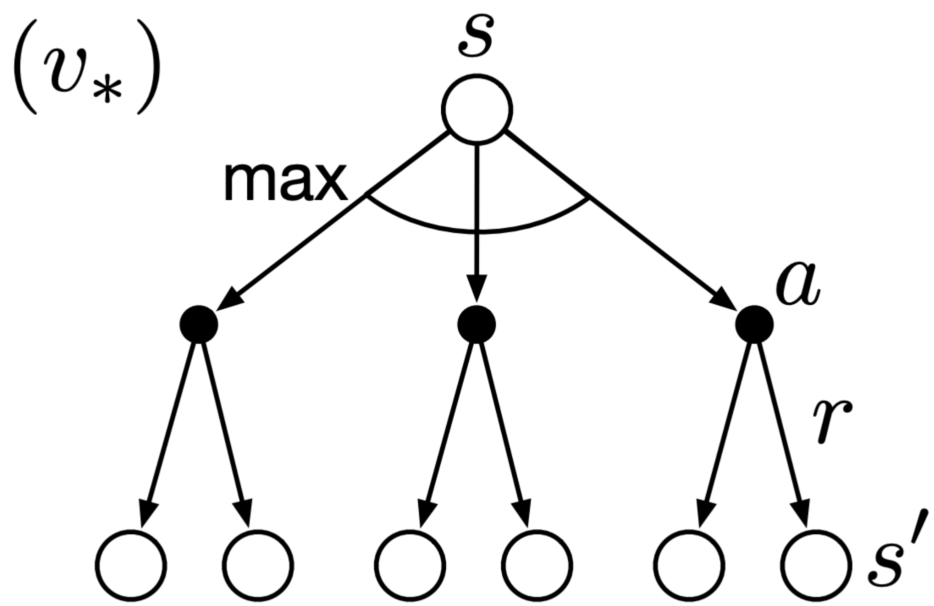
Group discussion

Use the bellman expectation equation to evaluate the top-most node of the triangle on the right under a random walk policy

```
      75
    95  64
  17  47  82
18  35  87  10
```



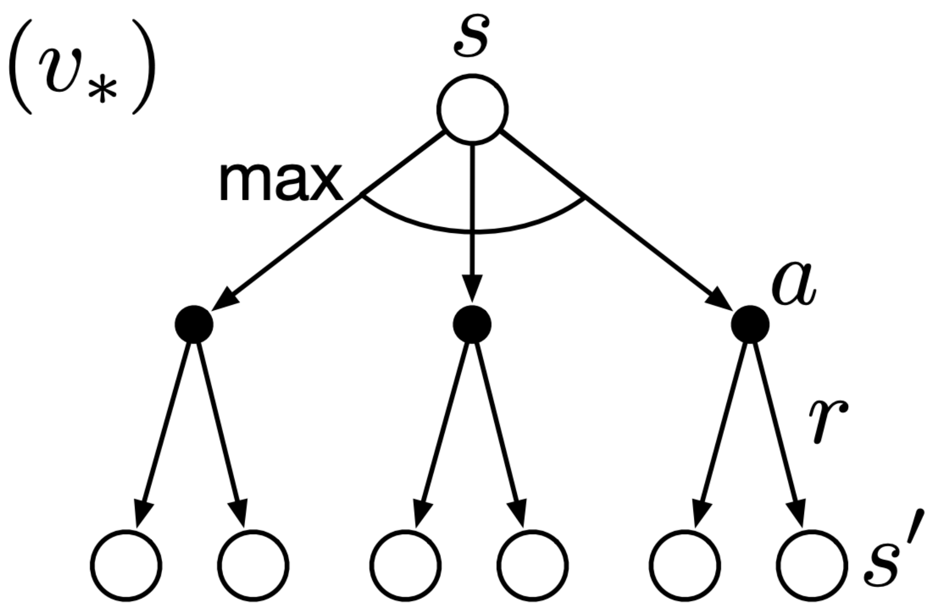
Optimal Value function and Optimal Policy



For any Markov Decision Process

- There exists a deterministic optimal policy π^* that is at least as good as all other policies, $\pi^* \geq \pi, \ \forall \pi$
- All optimal policies achieve the optimal value function, $v_{\pi^*}(s) = v_*(s)$

Optimal Value function and Optimal Policy

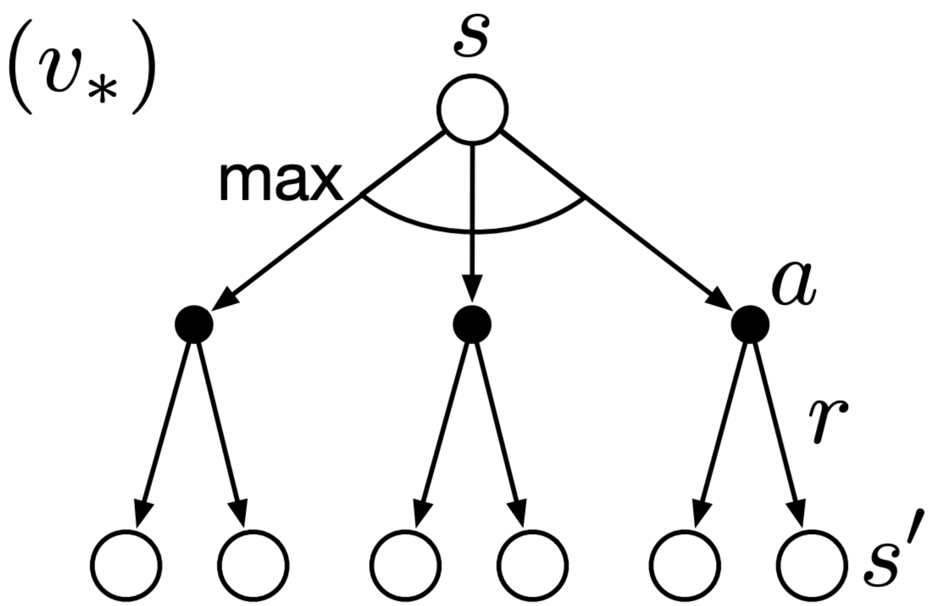


$$v_*(s) = \max_{a} \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')]$$

$$\pi^*(s) = \operatorname*{argmax}_{a} \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')]$$

Any policy that is *greedy* with respect to the optimal value function $v_*(s)$ is an optimal policy

Optimal Value function and Optimal Policy



Bellman Optimality Equation

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')]$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')]$$

Any policy that is *greedy* with respect to the optimal value function $v_*(s)$ is an optimal policy

Bellman Expectation Equation

- Linear, has a closed-form solution
- Used for *policy evaluation* or *prediction*

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

Bellman Optimality Equation

- Non-linear, no closed-form solution
- Used for computing an *optimal policy* or *control*

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')]$$

Bellman Optimality Equation

Solution Methods

- Policy Iteration
- Value Iteration

Model-based algorithms (*planning*)

- Monte Carlo Methods
- Temporal Difference

Model-free algorithms (*learning*)

Bellman Optimality Equation

Solution Methods

- Policy Iteration
- Value Iteration
- Monte Carlo Methods
- Temporal Difference

Policy Iteration

Random initialization:

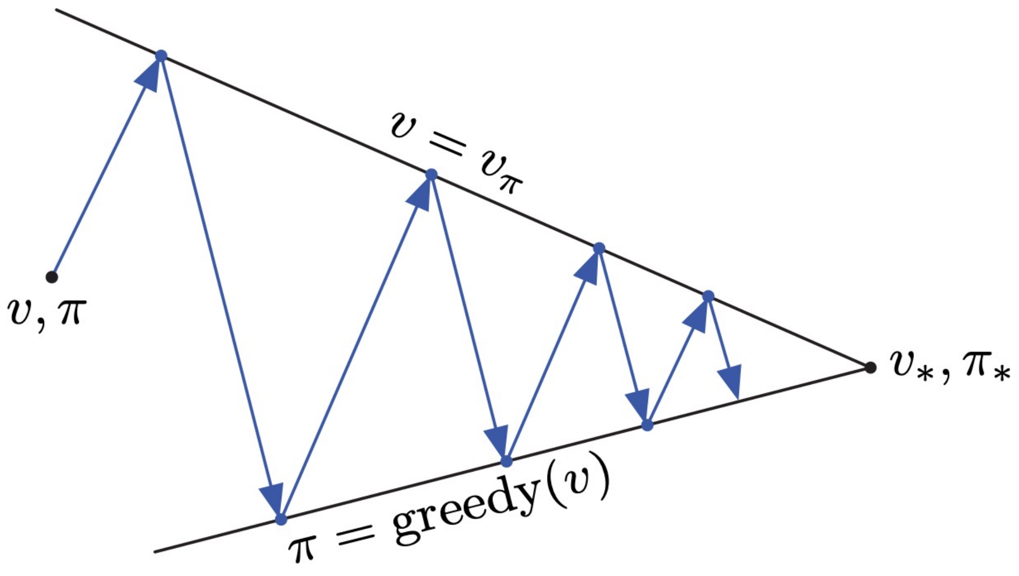
$V(s) \in \mathbb{R}, \pi(s) \in \mathcal{A}(s)$

Loop until $\pi(s)$ is stable:

1. Policy *evaluation*:
   Loop until convergence:
   $$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')] \text{ for each } s \in \mathcal{S}$$

2. Policy *improvement*:
   $$\pi(s) \leftarrow \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')] \text{ for each } s \in \mathcal{S}$$

Policy Iteration



Random initialization:
$V(s) \in \mathbb{R}, \pi(s) \in \mathcal{A}(s)$

Loop until $\pi(s)$ is stable:

1. Policy *evaluation*:
   Loop until convergence:
   $$V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))[r + \gamma V(s')] \text{ for each } s \in \mathcal{S}$$

2. Policy *improvement*:
   $$\pi(s) \leftarrow \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')] \text{ for each } s \in \mathcal{S}$$

Bellman Optimality Equation

Solution Methods

- Policy Iteration
- Value Iteration
- Monte Carlo Methods
- Temporal Difference

Value Iteration

Random initialization:
$V(s) \in \mathbb{R}, \pi(s) \in \mathcal{A}(s)$

Loop until $V(s)$ is stable (combines *evaluation* and *improvement*):

$$V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] \text{ for each } s \in \mathcal{S}$$

$$\pi(s) = \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$$

Group discussion (5 mins)

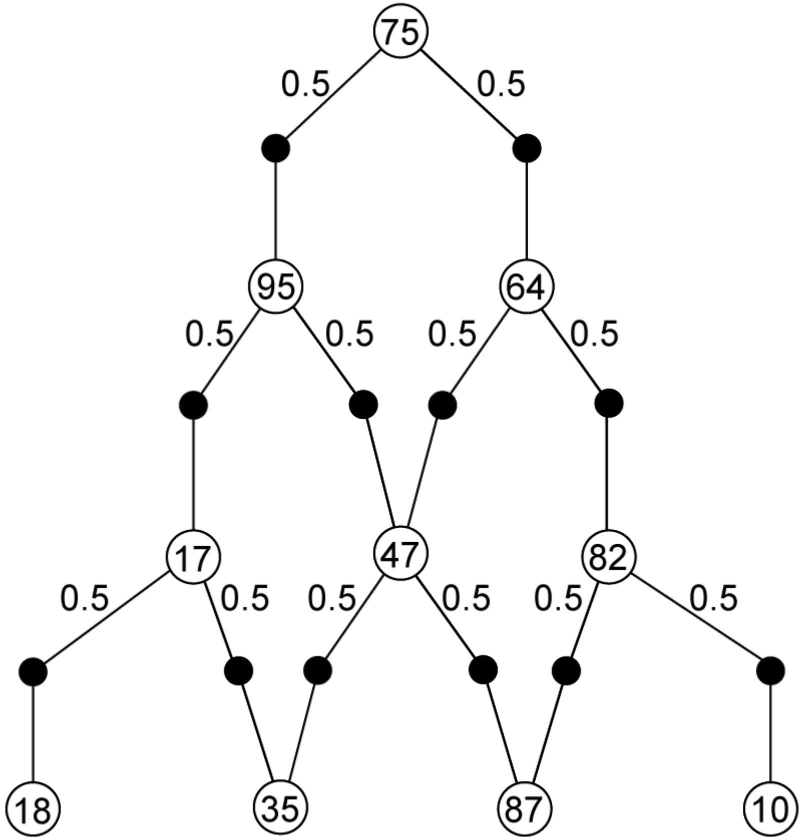
Use *value iteration* to find the optimal value function and the optimal policy for the triangle on the right

```
                              75
                            95 64
                          17 47 82
                        18 35 87 10
                      20 04 82 47 65
                    19 01 23 75 03 34
                  88 02 77 73 07 63 67
                99 65 04 28 06 16 70 92
              41 41 26 56 83 40 80 70 33
            41 48 72 33 47 32 37 16 94 29
          53 71 44 65 25 43 91 52 97 51 14
        70 11 33 28 77 73 17 78 39 68 17 57
      91 71 52 38 17 14 91 43 58 50 27 29 48
    63 66 04 68 89 53 67 30 73 16 69 87 40 31
  04 62 98 27 23 09 70 98 73 93 38 53 60 04 23
```


```
   75
  95 64
 17 47 82
18 35 87 10
```

Group discussion

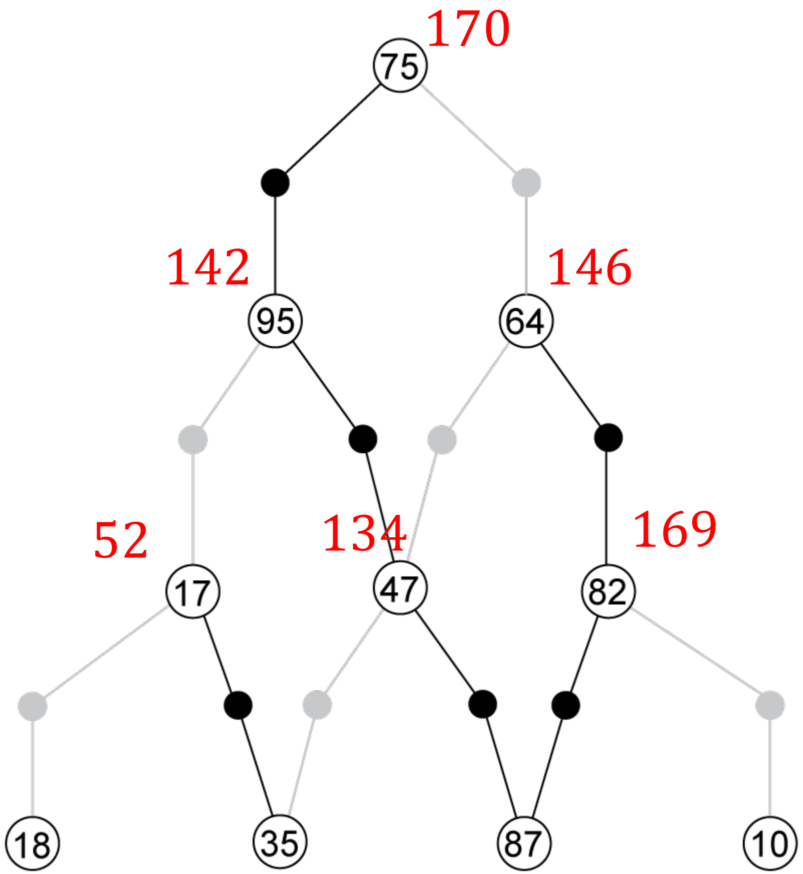
Use *value iteration* to find the optimal value function and the optimal policy for the triangle on the right



Group discussion

Use *value iteration* to find the optimal value function and the optimal policy for the triangle on the right

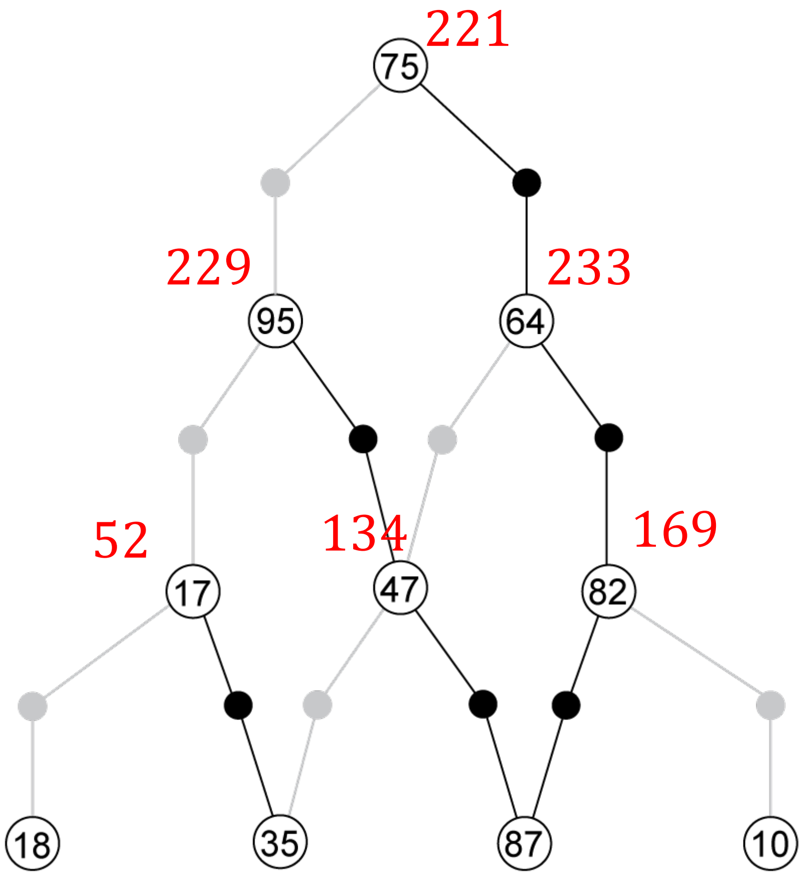
Iteration 1



Group discussion

Use *value iteration* to find the optimal value function and the optimal policy for the triangle on the right

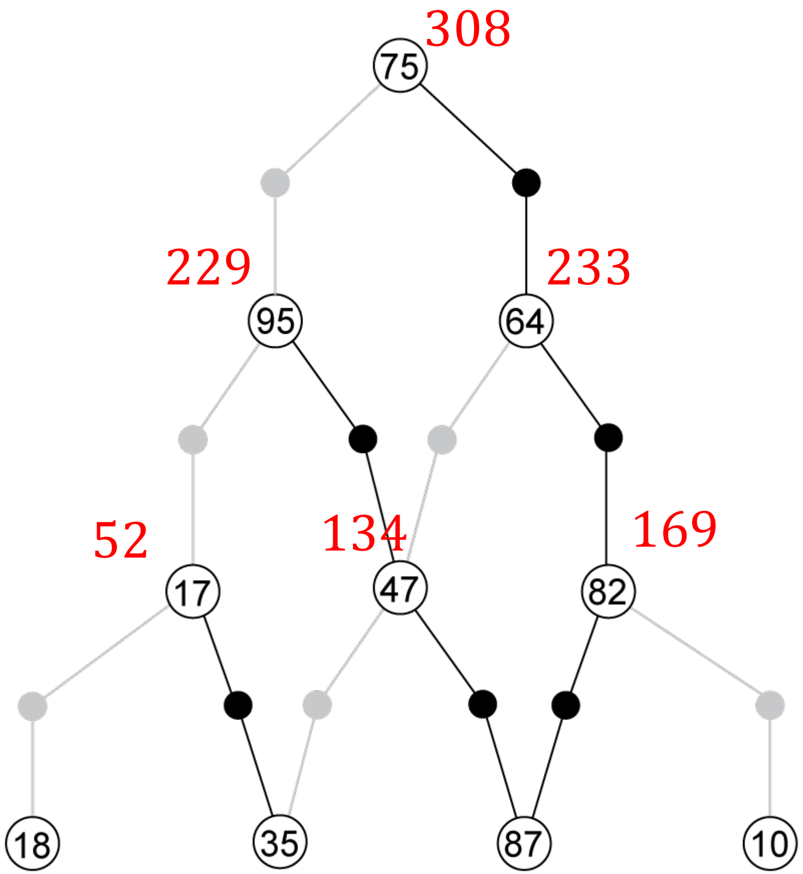
Iteration 2



Group discussion

Use *value iteration* to find the optimal value function and the optimal policy for the triangle on the right

Iteration 3



Try on your own

Write a program that uses *value iteration* to maximize the total from top to bottom

```
                            75
                          95 64
                        17 47 82
                      18 35 87 10
                    20 04 82 47 65
                  19 01 23 75 03 34
                88 02 77 73 07 63 67
              99 65 04 28 06 16 70 92
            41 41 26 56 83 40 80 70 33
          41 48 72 33 47 32 37 16 94 29
        53 71 44 65 25 43 91 52 97 51 14
      70 11 33 28 77 73 17 78 39 68 17 57
    91 71 52 38 17 14 91 43 58 50 27 29 48
  63 66 04 68 89 53 67 30 73 16 69 87 40 31
04 62 98 27 23 09 70 98 73 93 38 53 60 04 23
```