

# Implementing Value Iteration

1. Make it work
2. Make it right
3. Make it fast

# Implementing Value Iteration

1. Make it work
2. Make it right
3. Make it fast



# Implementing Value Iteration

1. Make it work      
  2. Make it right      
  3. Make it fast      
- Problem 4
- Problem 5

# Implementing Value Iteration

1. Make it work
  2. Make it right
  3. Make it fast
- 
- The diagram consists of three horizontal arrows pointing left towards the list items. The first arrow points to 'Problem 4' and ends under the second list item. The second arrow points to 'Problem 5' and ends under the third list item. The third arrow points to the first list item and ends under the first list item.
- Problem 4
- Problem 5

First step for making it fast (in any language, not just julia):

# Implementing Value Iteration

1. Make it work
  2. Make it right
  3. Make it fast
- 
- The diagram consists of two sets of arrows. The first set, labeled 'Problem 4', has two arrows pointing to the second item in the list. The second set, labeled 'Problem 5', has one arrow pointing to the third item in the list.

First step for making it fast (in any language, not just julia):

Find out what is slow (by profiling)!

# Bellman Operator

$$U' = B[U]$$

$$B[U](s) = \max_a \underbrace{\left( R(s, a) + \gamma \sum_{s'} T(s' | s, a) U(s') \right)}_{Q(s, a)}$$

# Bellman Operator

$$U' = B[U]$$

$$B[U](s) = \max_a \underbrace{\left( R(s, a) + \gamma \sum_{s'} T(s' | s, a) U(s') \right)}_{Q(s, a)}$$

$i$  = index of  $s$ ;  $j$  = index of  $s'$

Naive implementation:

$$U'[i] = \max_a \left( R[a][i] + \gamma \sum_j T[a][i, j] U[j] \right)$$

# Bellman Operator

$$U' = B[U]$$

$$B[U](s) = \max_a \underbrace{\left( R(s, a) + \gamma \sum_{s'} T(s' | s, a) U(s') \right)}_{Q(s, a)}$$

$i$  = index of  $s$ ;  $j$  = index of  $s'$

$$y = Mx$$

Naive implementation:

$$y[i] = \sum_j M[i, j]x[i]$$

$$U'[i] = \max_a \left( R[a][i] + \gamma \sum_j T[a][i, j]U[j] \right)$$