


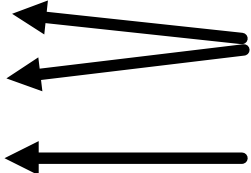
Implementing Value Iteration

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

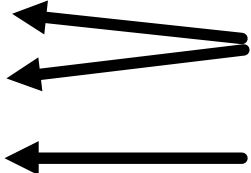
Implementing Value Iteration

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

Implementing Value Iteration

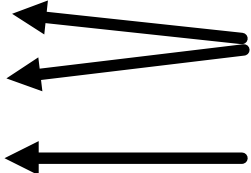
- 1. Make it work
 - 2. Make it right
 - 3. Make it fast
- Problem 4
- Problem 5
- 
- The diagram consists of two arrows originating from the text 'Problem 4'. One arrow points to the first item '1. Make it work' and the other points to the second item '2. Make it right'. A single arrow originates from the text 'Problem 5' and points to the third item '3. Make it fast'.

Implementing Value Iteration

- 1. Make it work
 - 2. Make it right
 - 3. Make it fast
- 
- The diagram shows two arrows originating from the text 'Problem 4' and pointing to the first two items of the list, '1. Make it work' and '2. Make it right'. A single arrow originates from the text 'Problem 5' and points to the third item, '3. Make it fast'.
- 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
- Problem 4
- Problem 5
- 
- The diagram shows two arrows originating from the text 'Problem 4' and pointing to the first two items of the list: '1. Make it work' and '2. Make it right'. A single arrow originates from the text 'Problem 5' and points to the third item: '3. Make it fast'.

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

Find out what is slow (by profiling)!

Bellman Operator

$$U' = B[U]$$

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

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

Naive implementation:

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