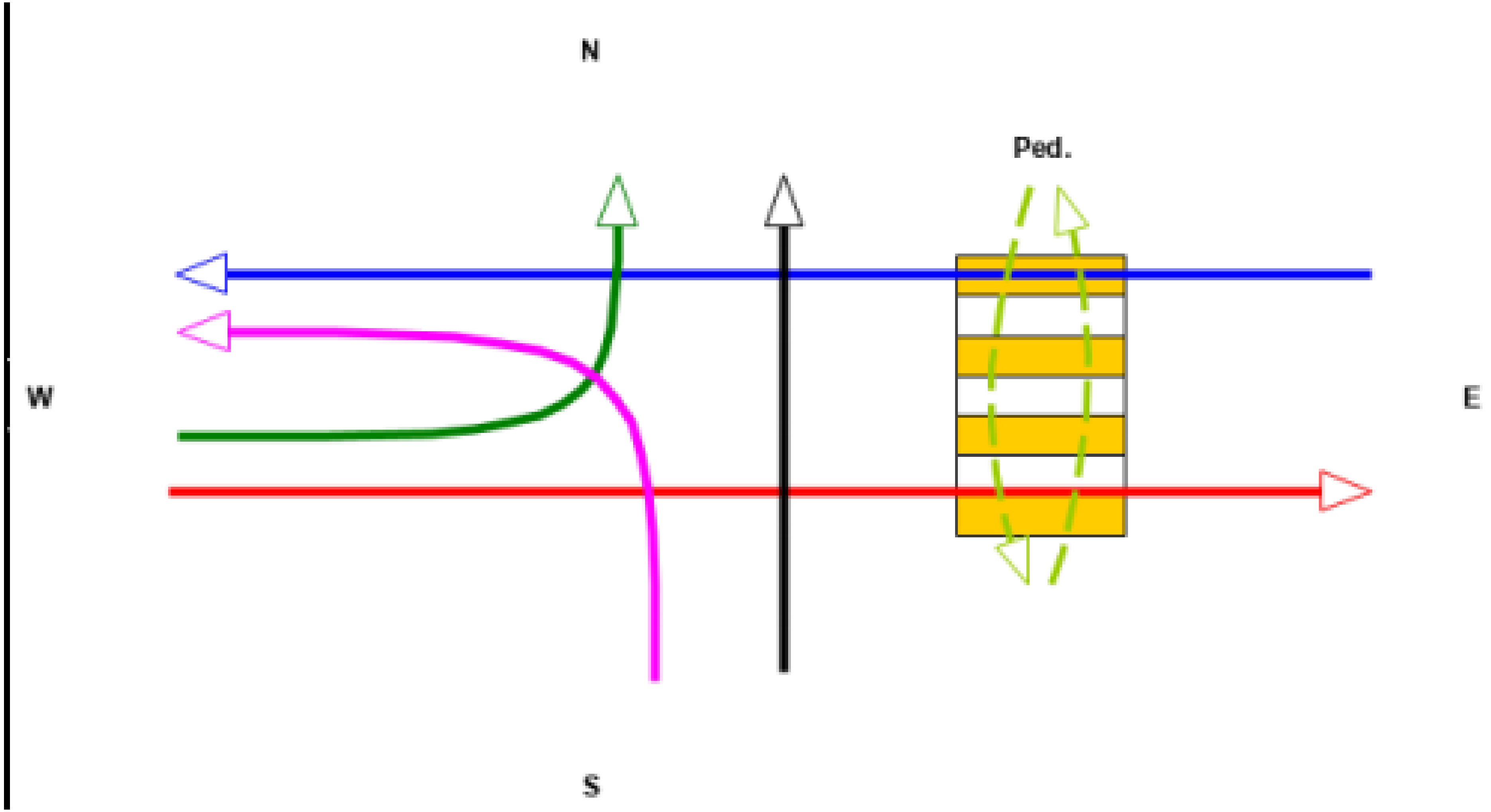
Creating and testing a traffic light model

Gilman Maxim

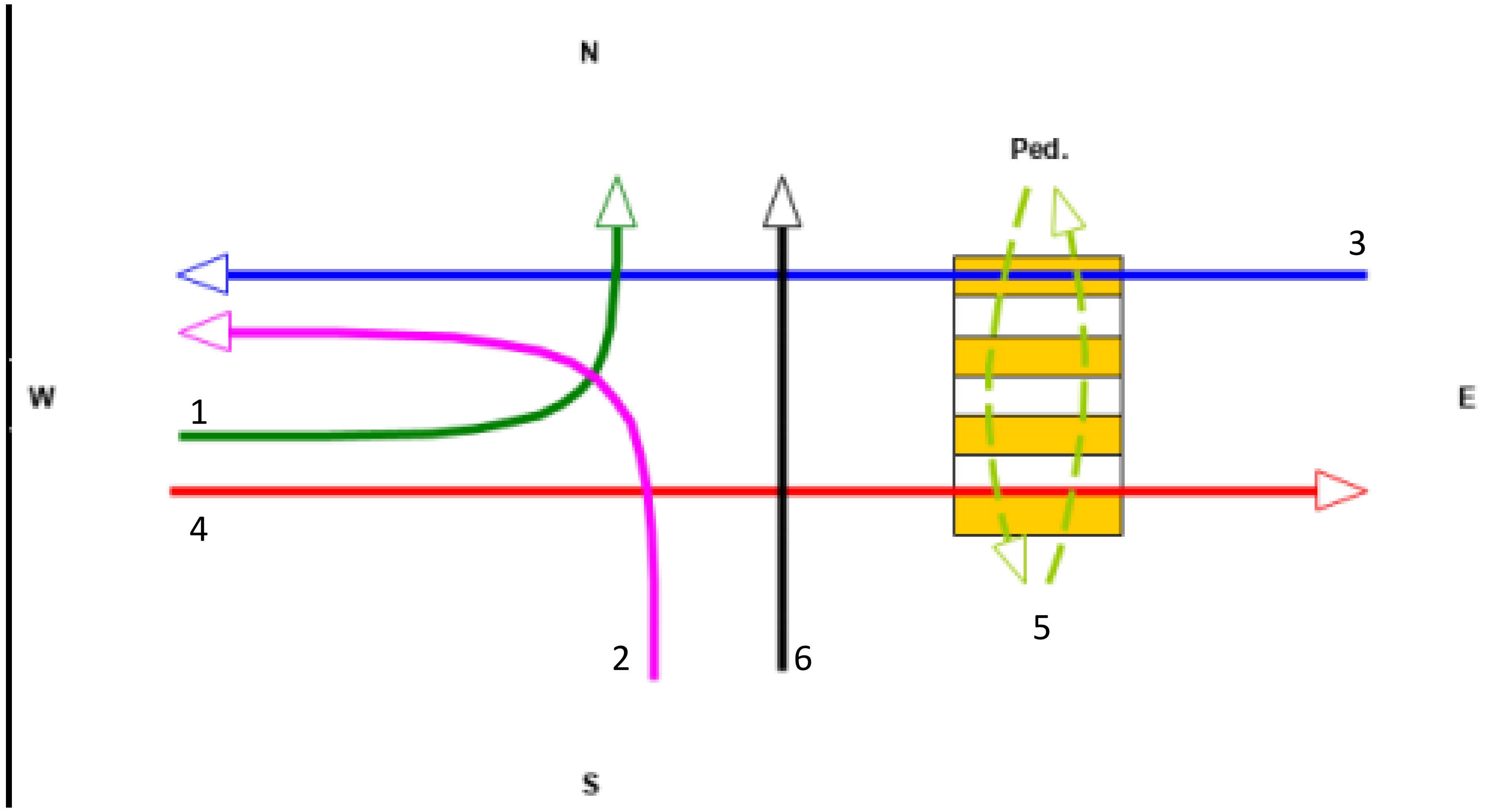
Task

11



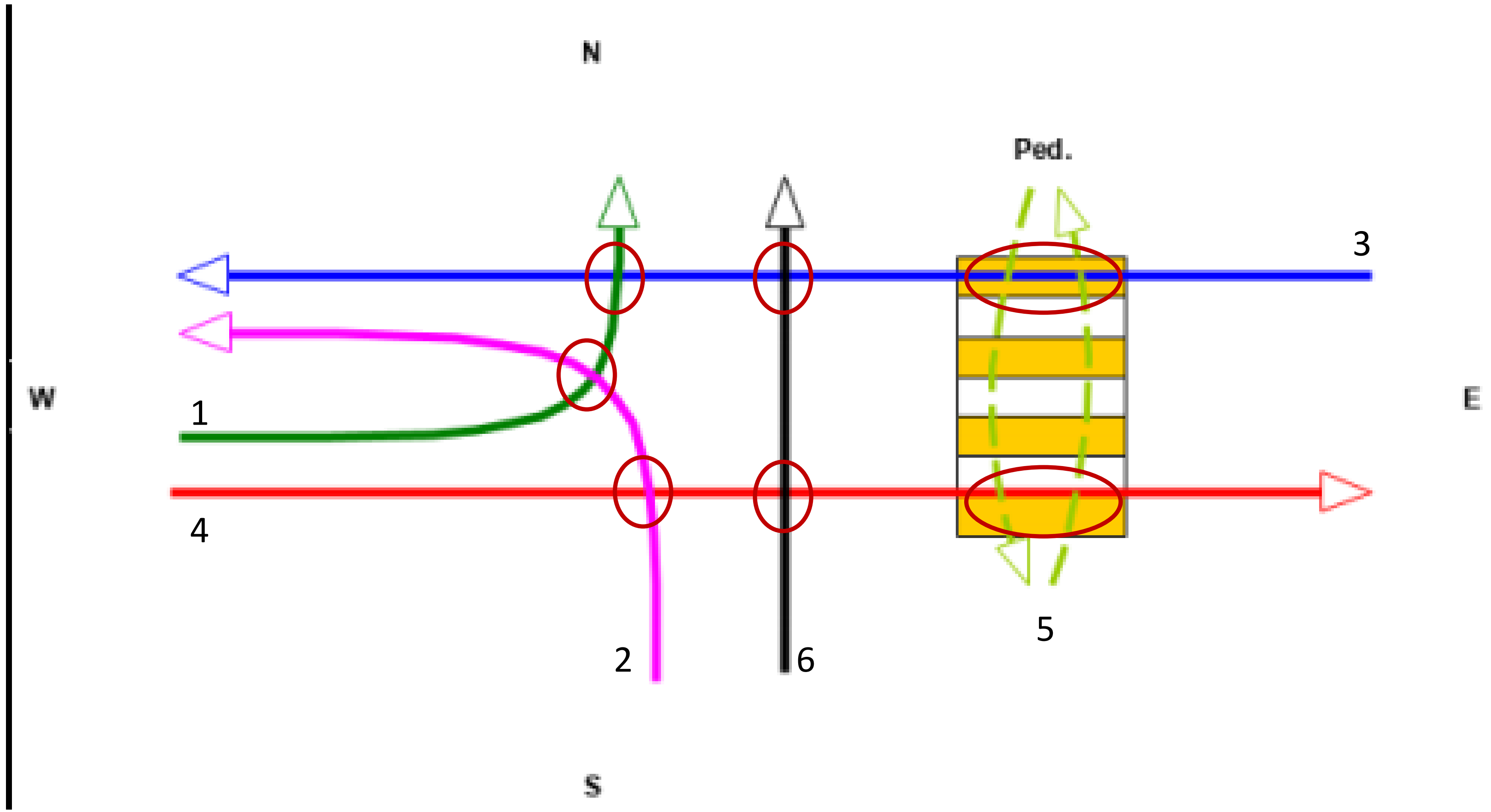
Idea

11



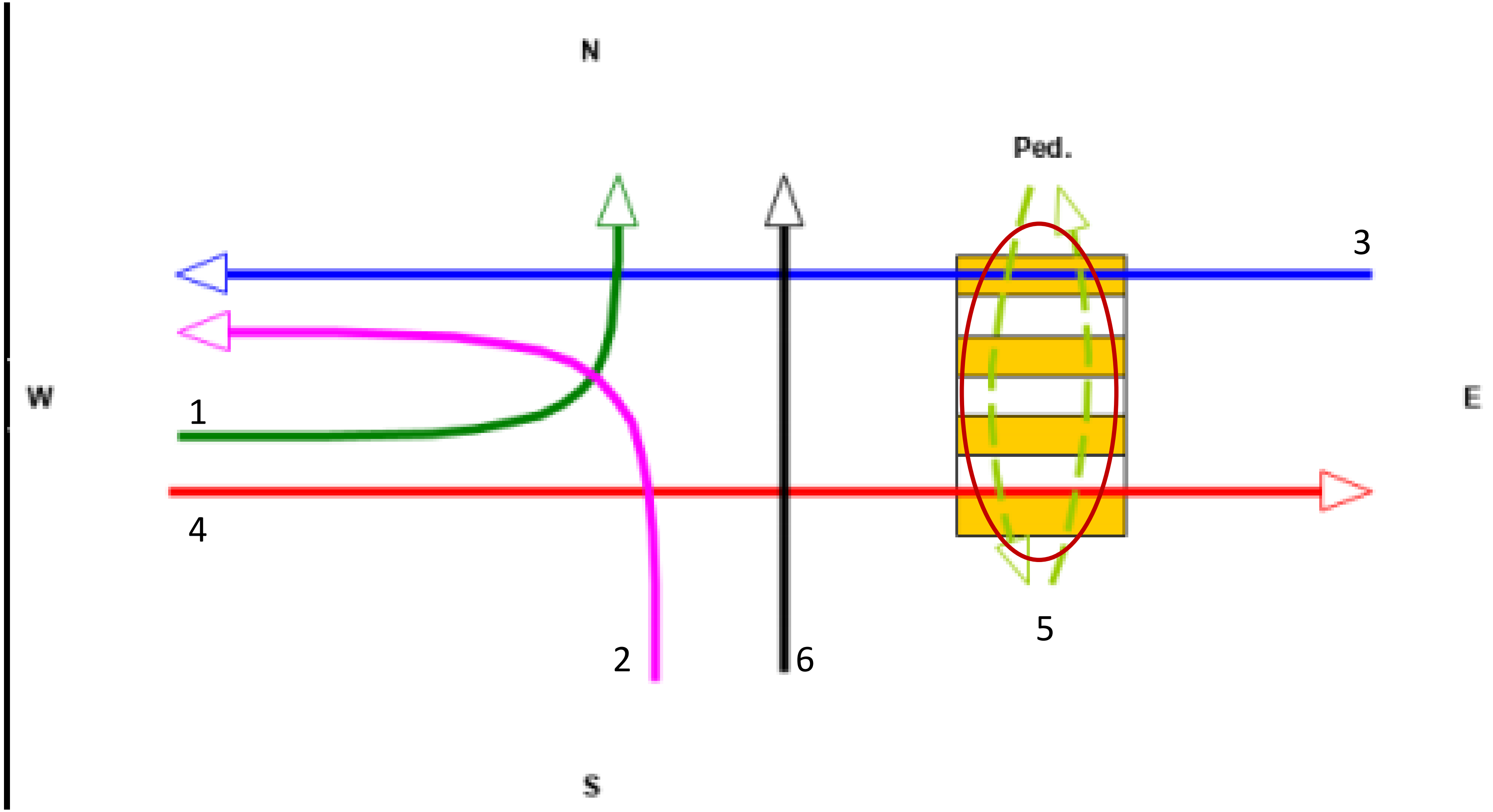
Idea

11



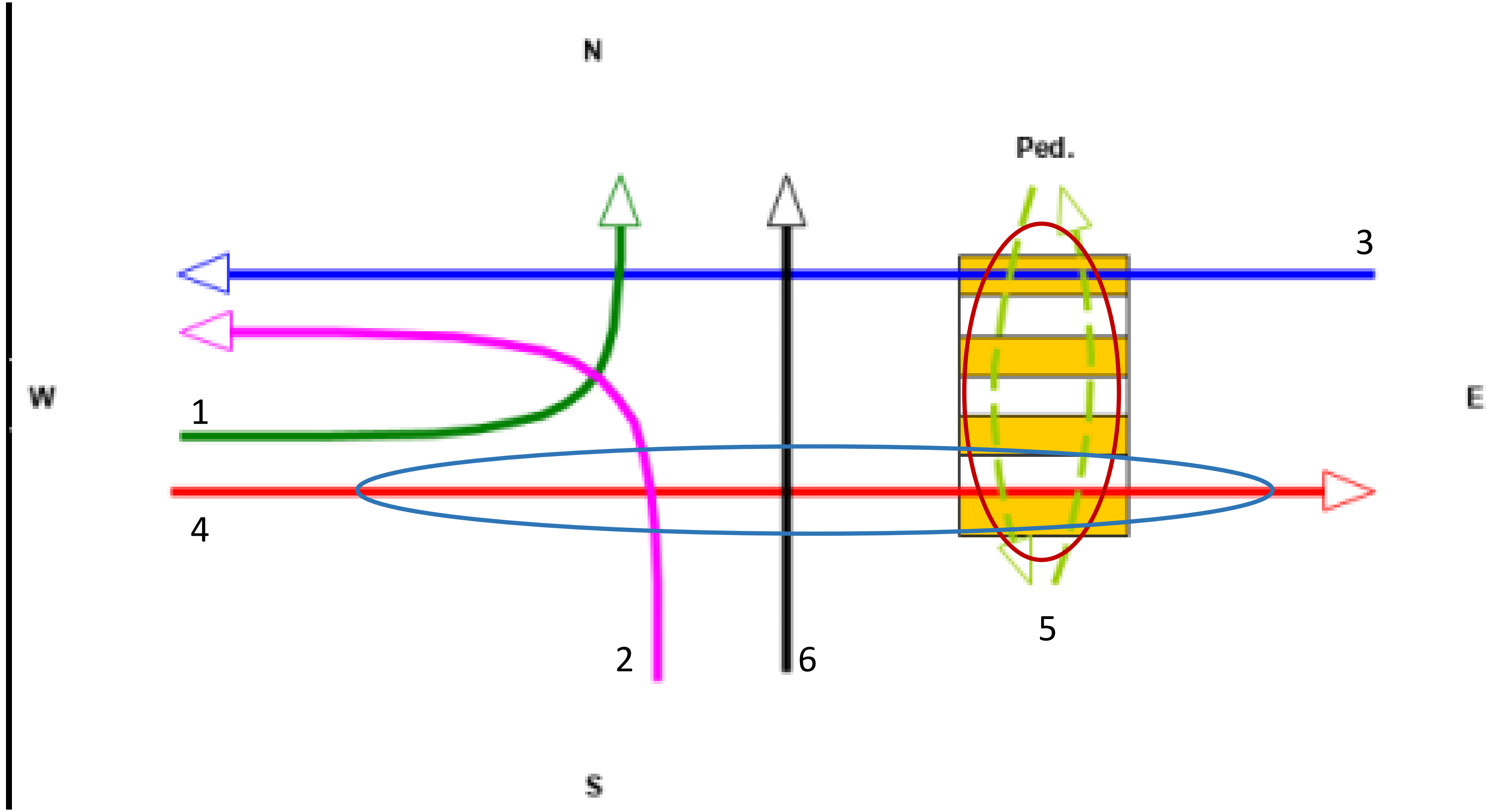
Idea

11



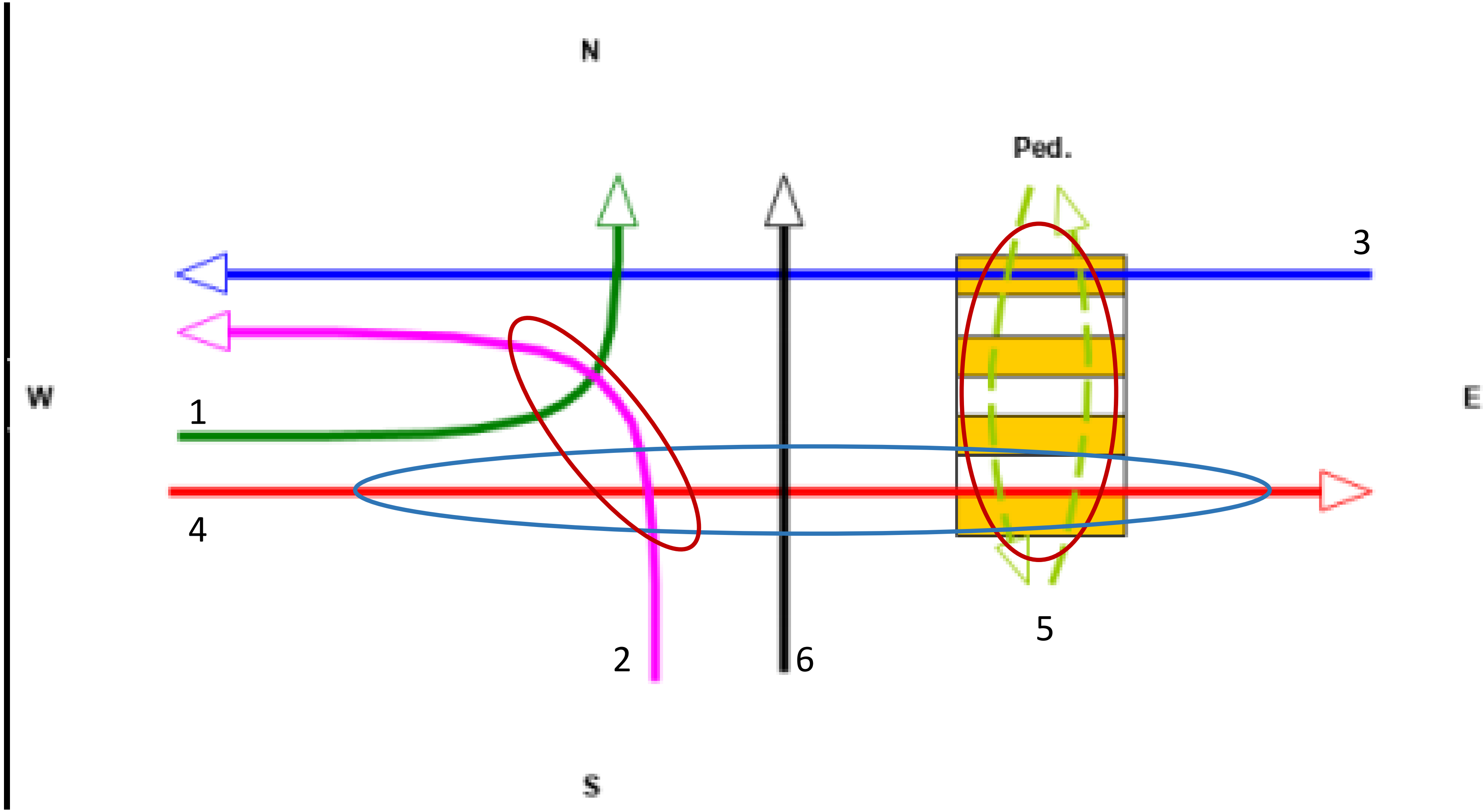
Idea

11

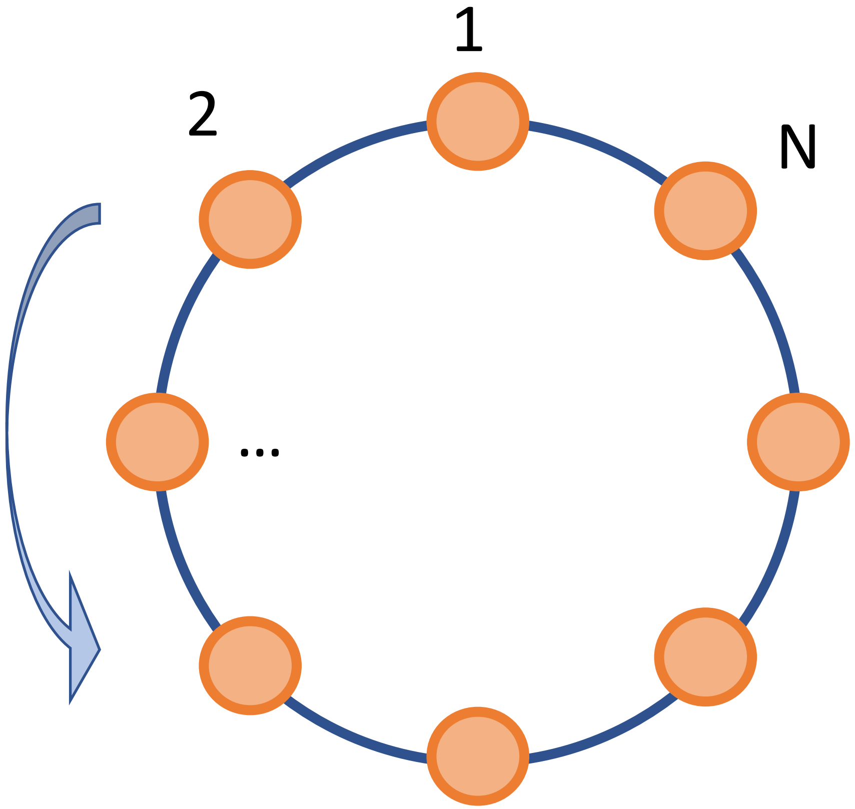


Idea

11



Algorithm



Global

- Current turn

Each

- My №
- Neighbour №
- Competitors №s

Architecture

- N traffic lights

```
proctype TrafficLight();
```

- N message channels for traffic

```
chan TL1 = [M] of {byte};
```

- N statuses – red /green

```
bool statuses [6];
```

- N +1 requests

```
short requests [7];
```

Inside the TL

- IF
  (It is my turn && there is some cars for me) ->
    - IF
      (I didn’t ask yet) ->
      I asks to set color as green (but don’t do it yet);
      requests [n] = n;
      Set next turn
    - ELSE ->
        - IF (there is no rivals) -> I win; Set green light ; all cars gone
        - ELSE ->

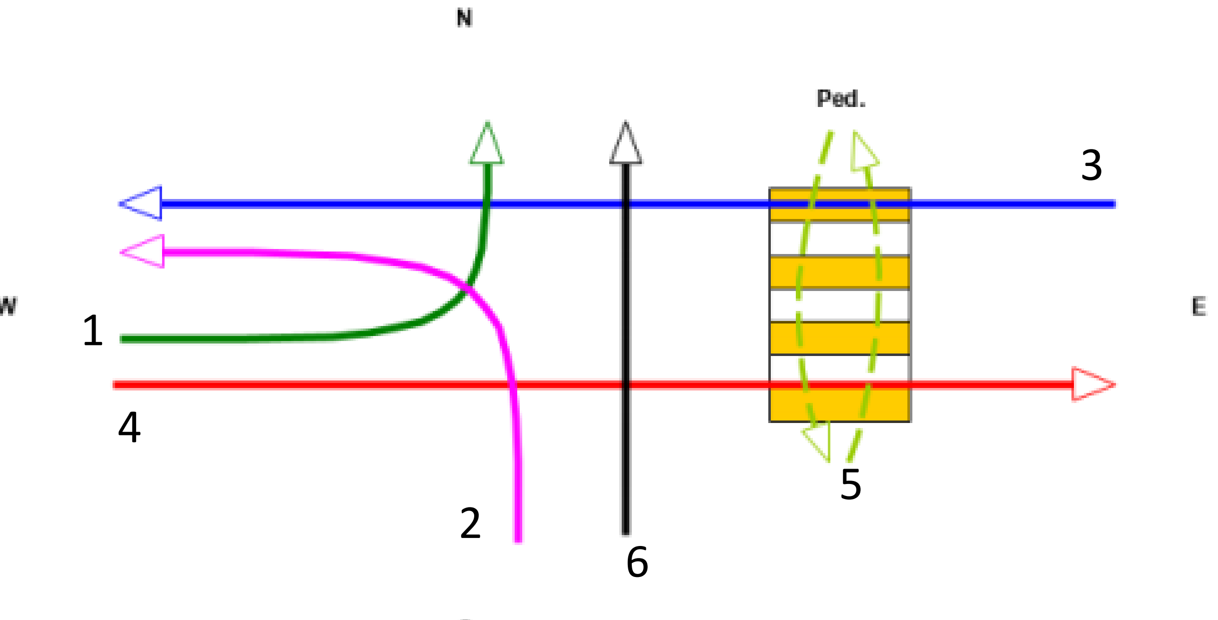
          Select MAX from all competitors;
            - If (it is me) -> I win; Set green light; all cars gone
            - Else ->
              Let winner go;
              For all TLs that fighted: requests[i] + n;

When TL set green and return after the circle -> return red light

Proctypes

```
proctype TrafficLight
(byte number; byte nextNum; byte fProblem; byte sProblem;
          byte tProblem; chan tlChan)
```

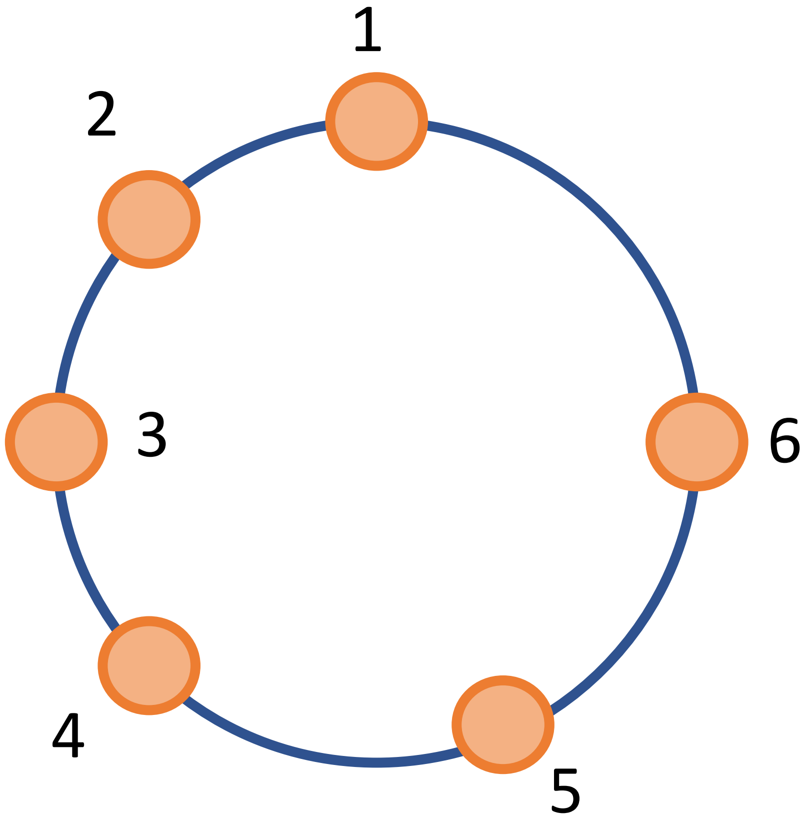
```
run  TrafficLight (1, 2, 2, 3, 0, TL1);
```



Example

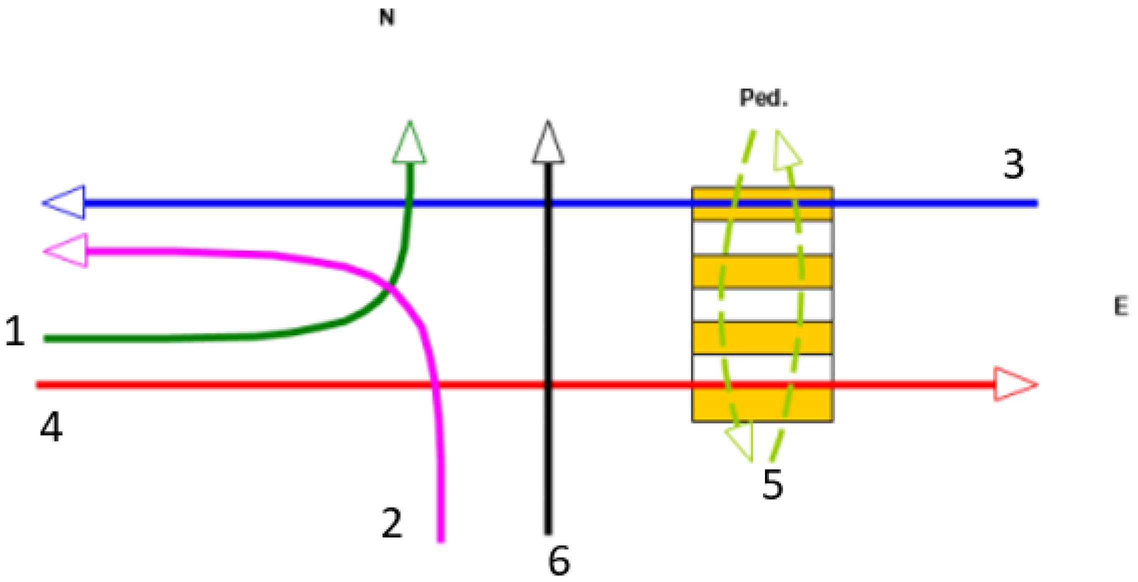
All wants to run

Current turn: **1**



Requests:

[0 , 0 , 0 , 0 , 0 , 0]



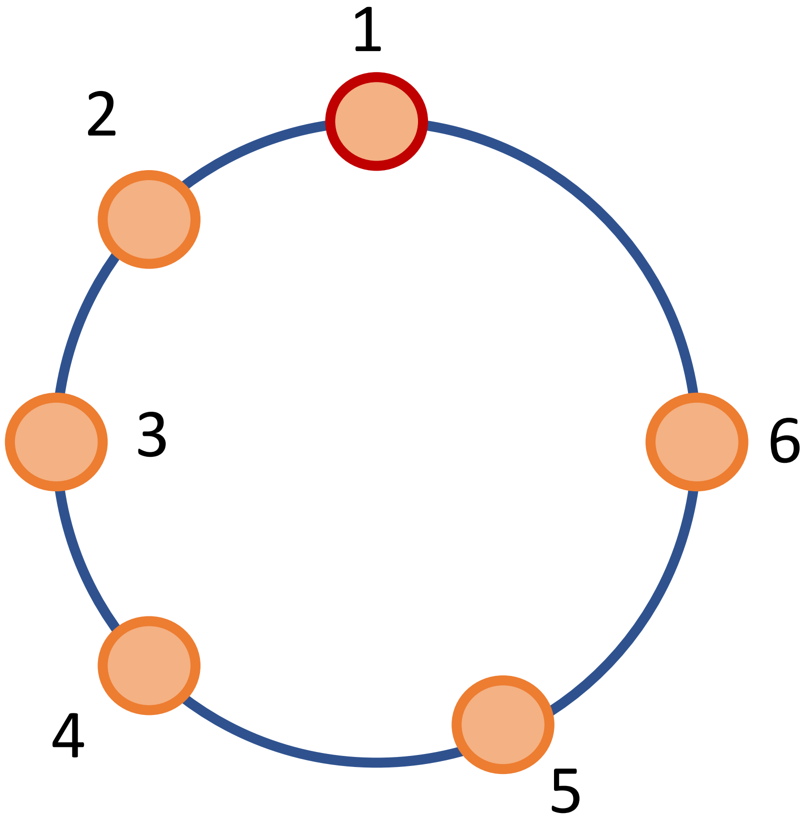
Example

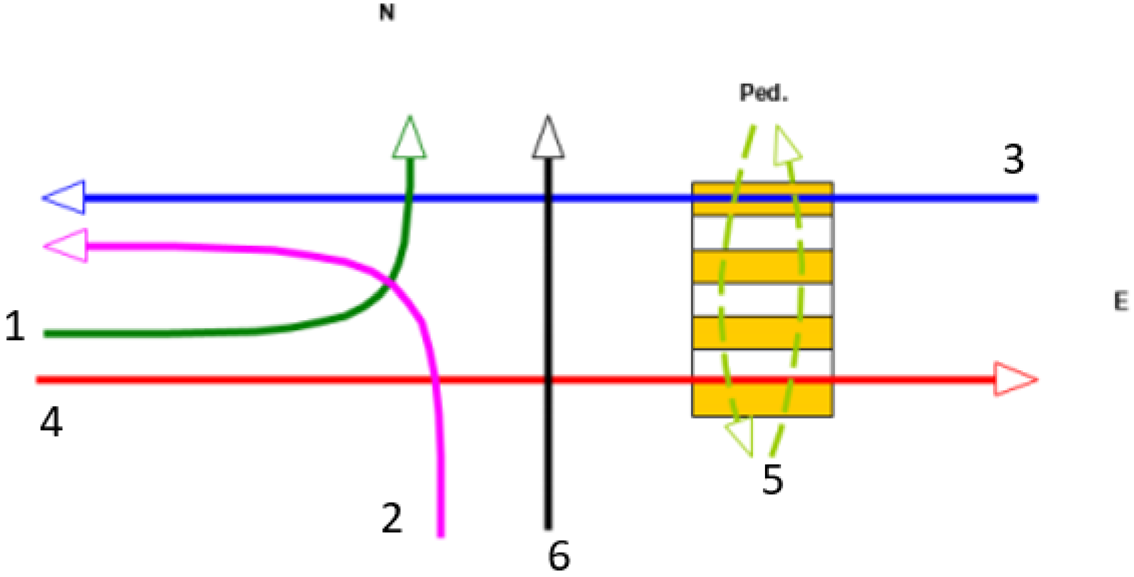
All wants to run

Current turn: **1**

Requests:

[0 , 0 , 0 , 0 , 0 , 0]

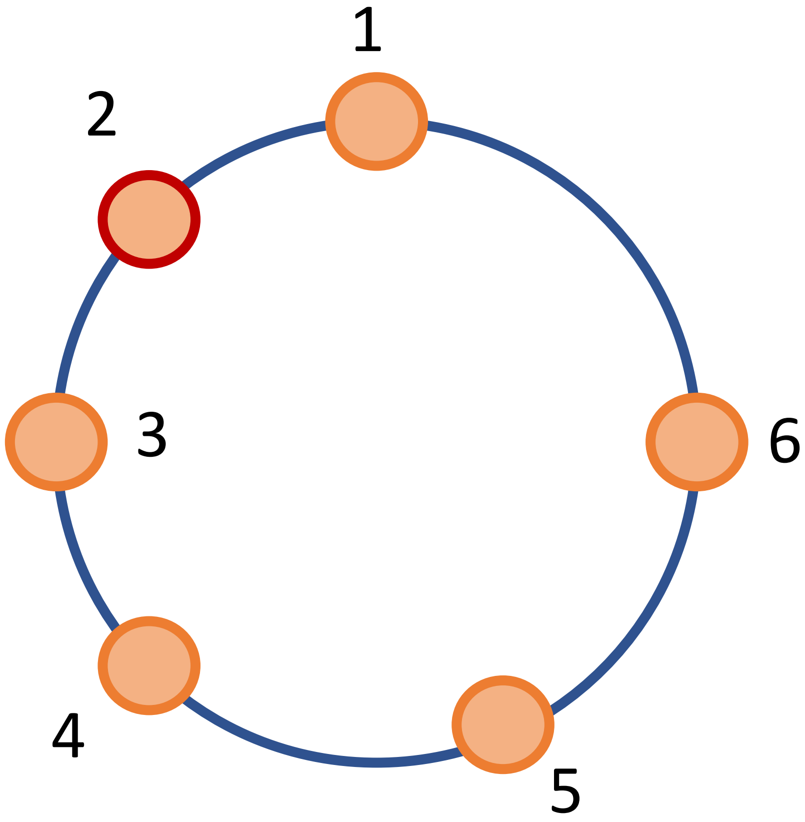




Example

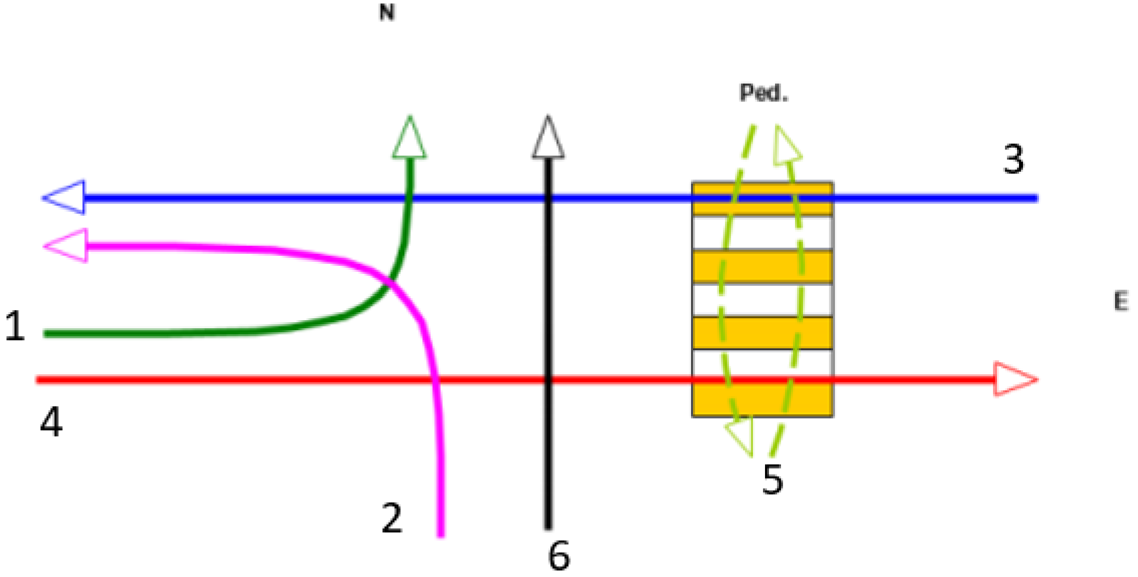
All wants to run

Current turn: **2**



Requests:

[1 , 0 , 0 , 0 , 0 , 0]



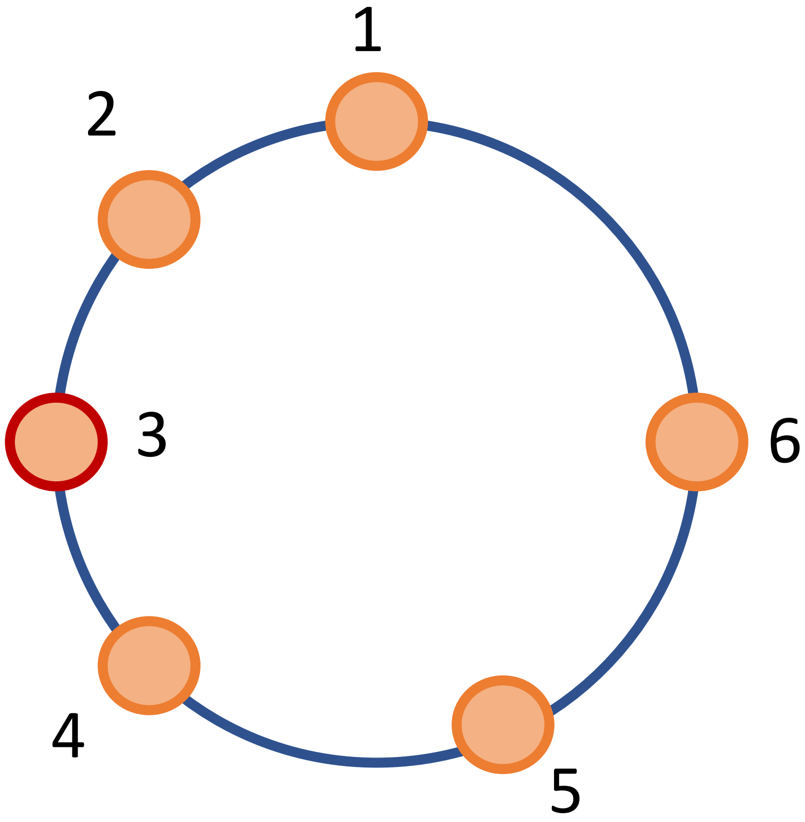
Example

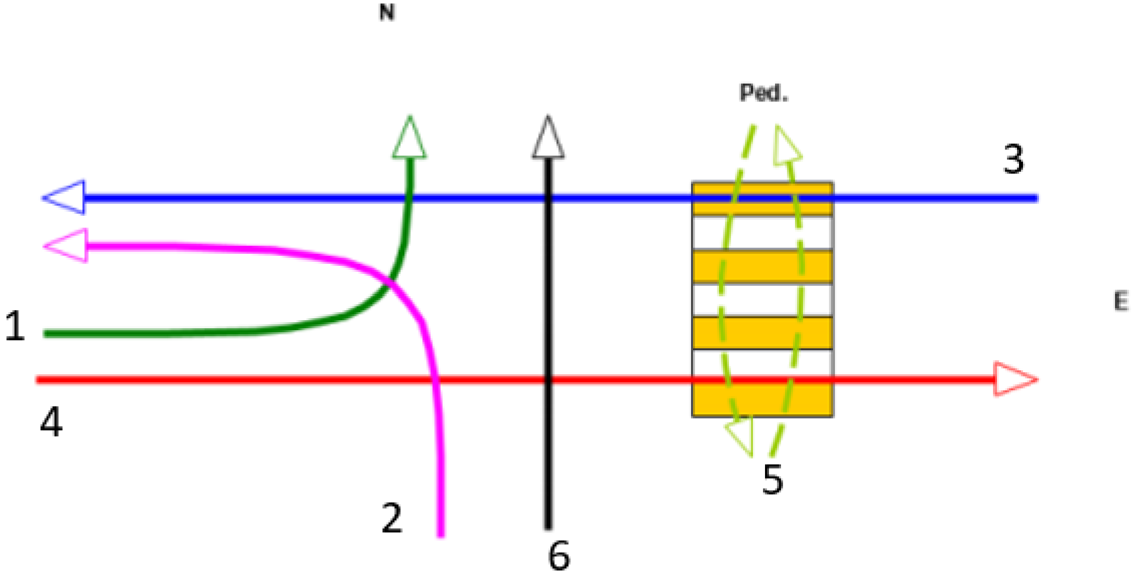
All wants to run

Current turn: **3**

Requests:

[1 , 2 , 0 , 0 , 0 , 0]

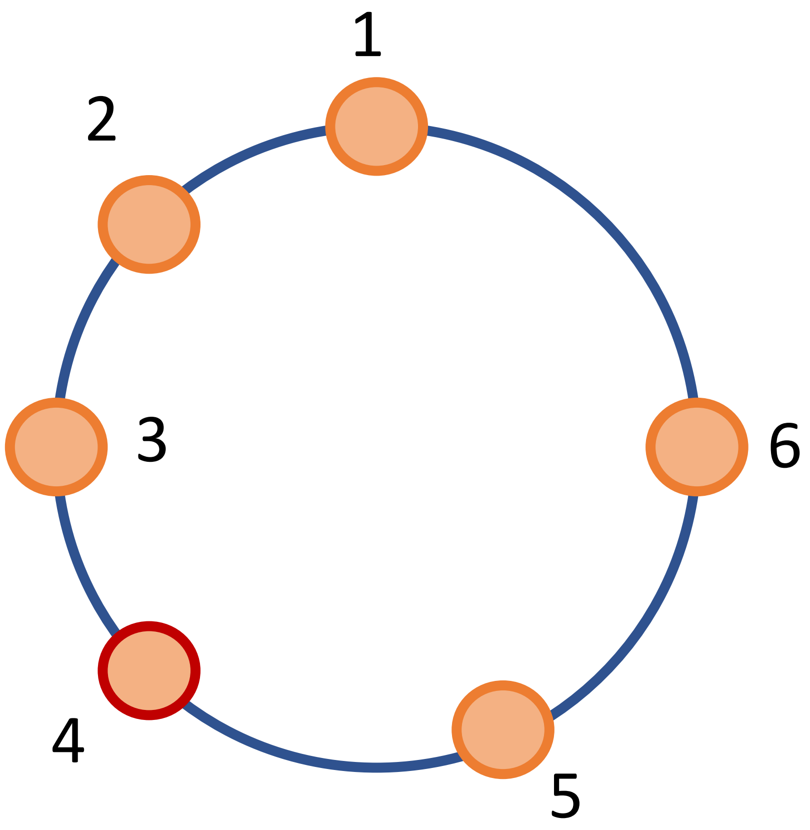




Example

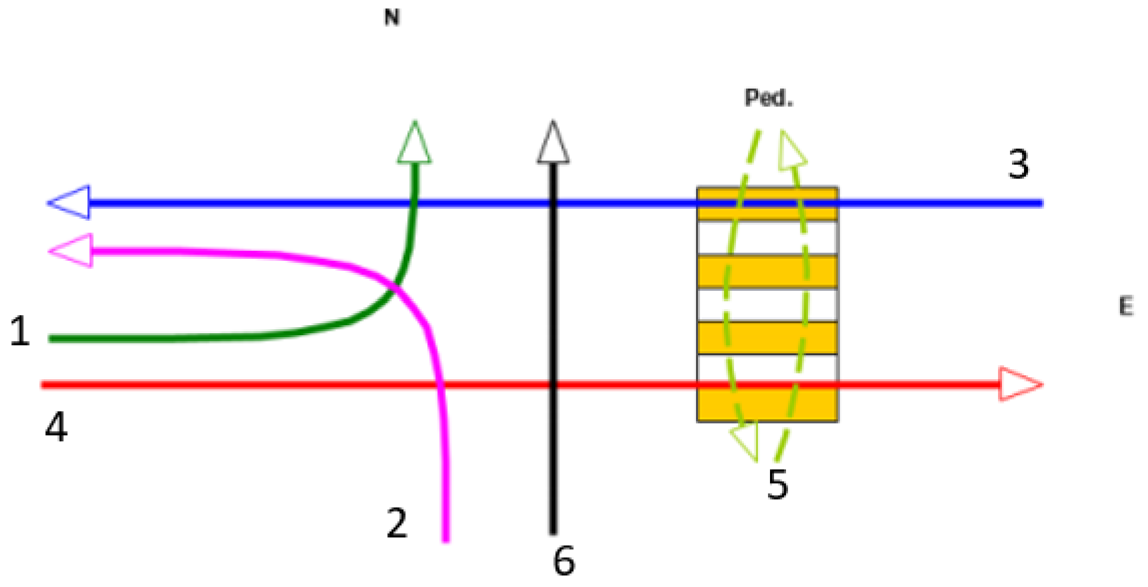
All wants to run

Current turn: **4**



Requests:

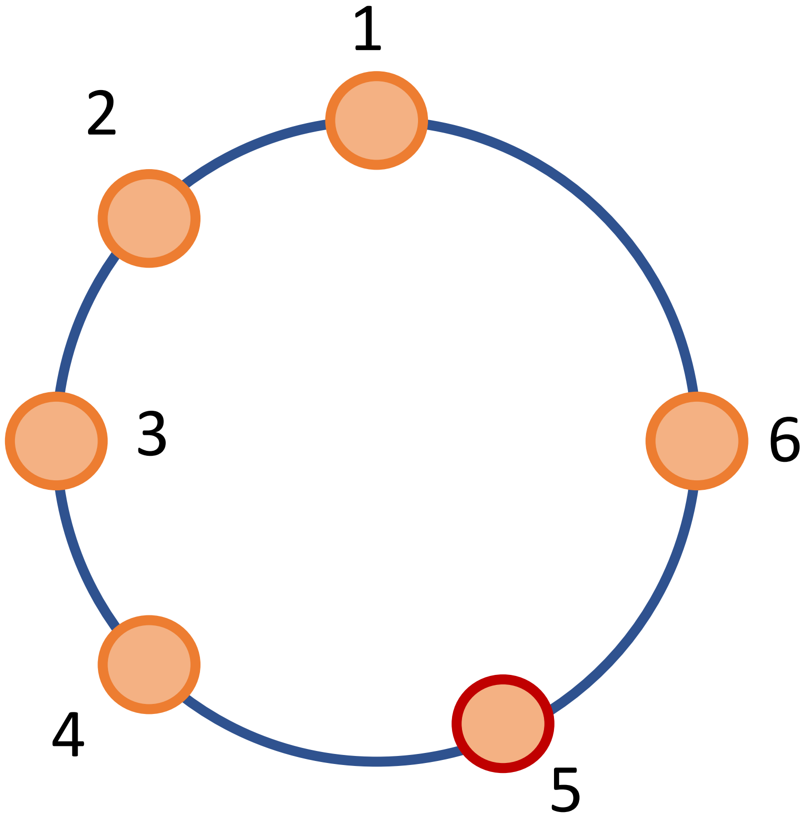
[1 , 2 , 3 , 0 , 0 , 0]



Example

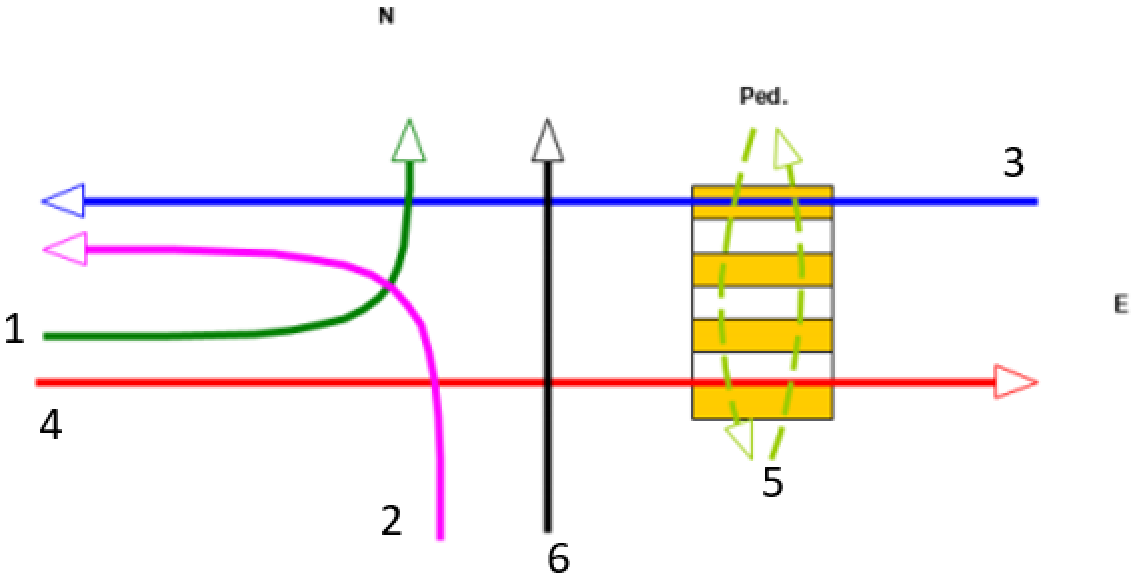
All wants to run

Current turn: **5**



Requests:

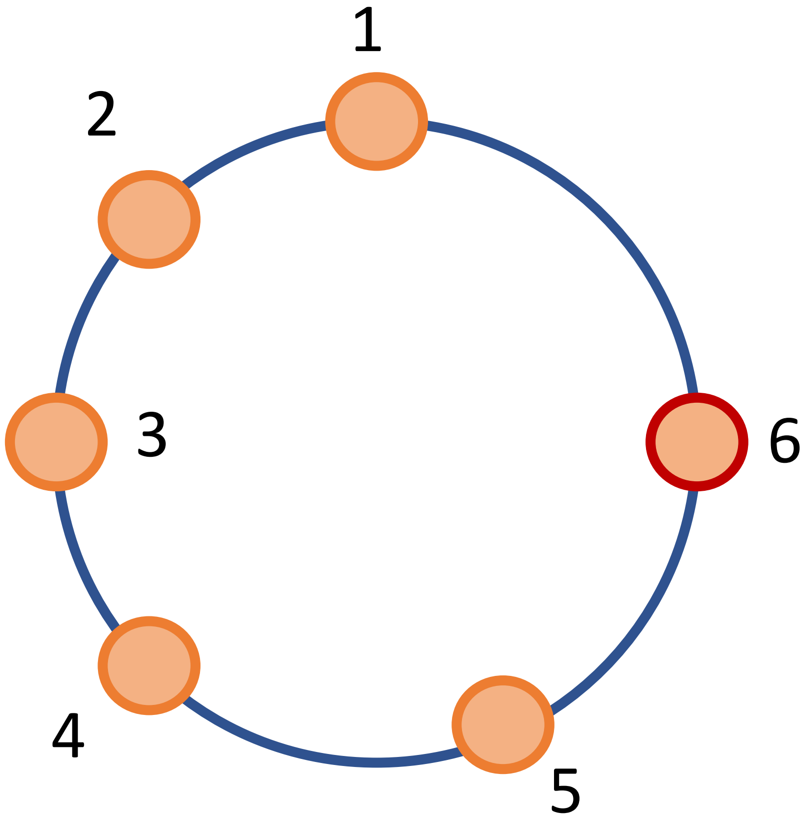
[1 , 2 , 3 , 4 , 0 , 0]



Example

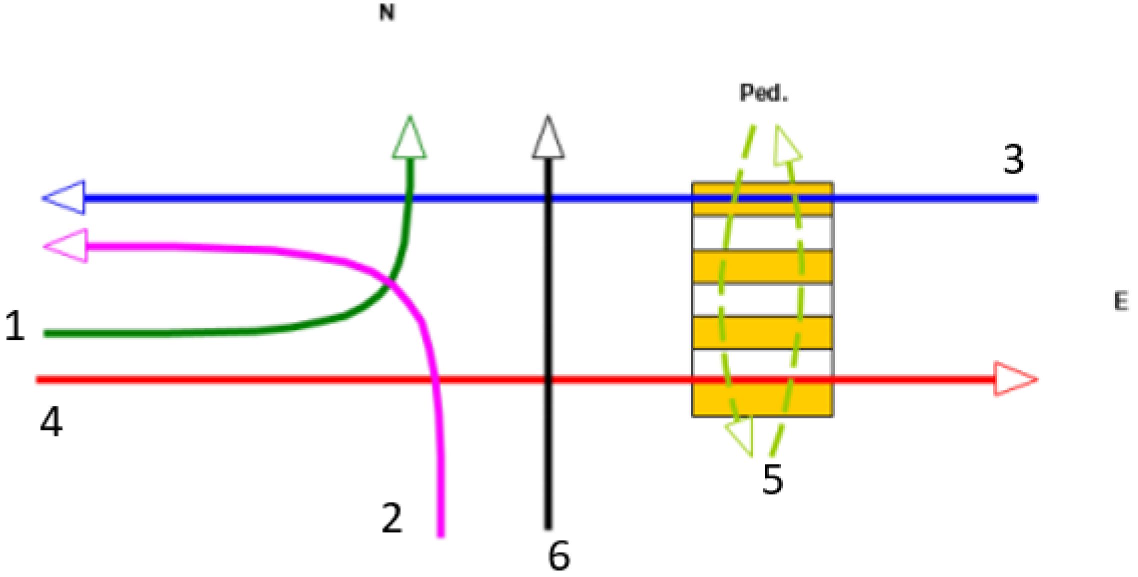
All wants to run

Current turn: **6**



Requests:

[1 , 2 , 3 , 4 , 5 , 0]



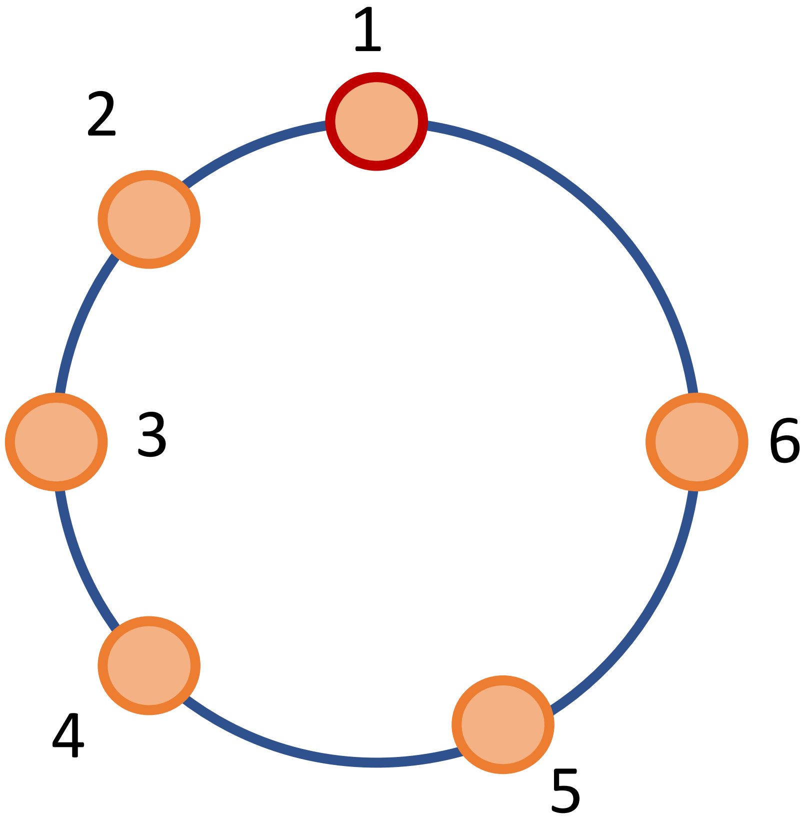
Example

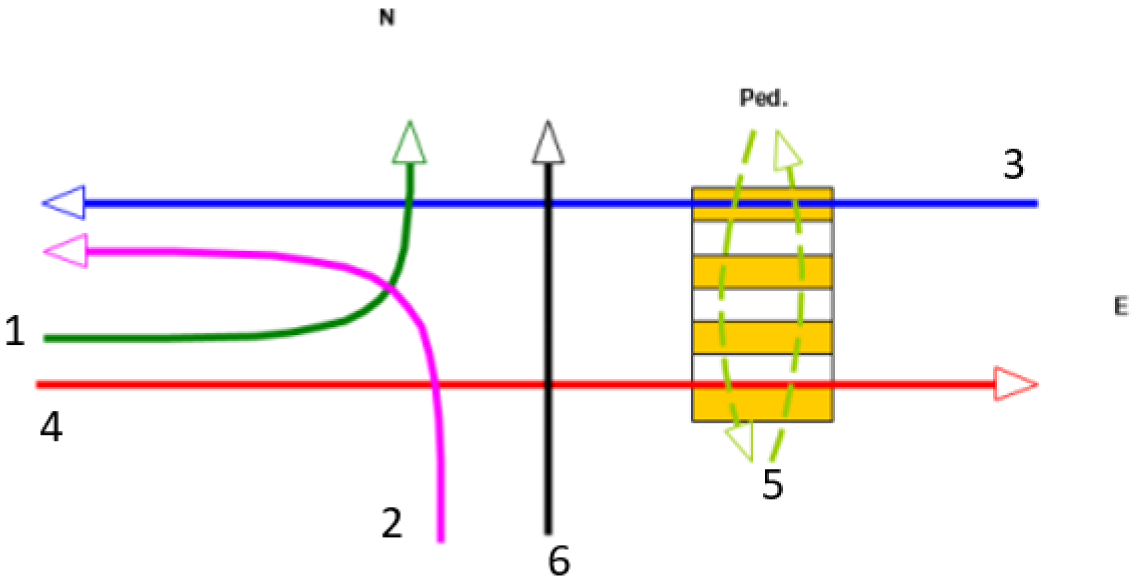
All wants to run

Current turn: **1**

Requests:

[1 , **2** , **3** , 4 , 5 , 6]





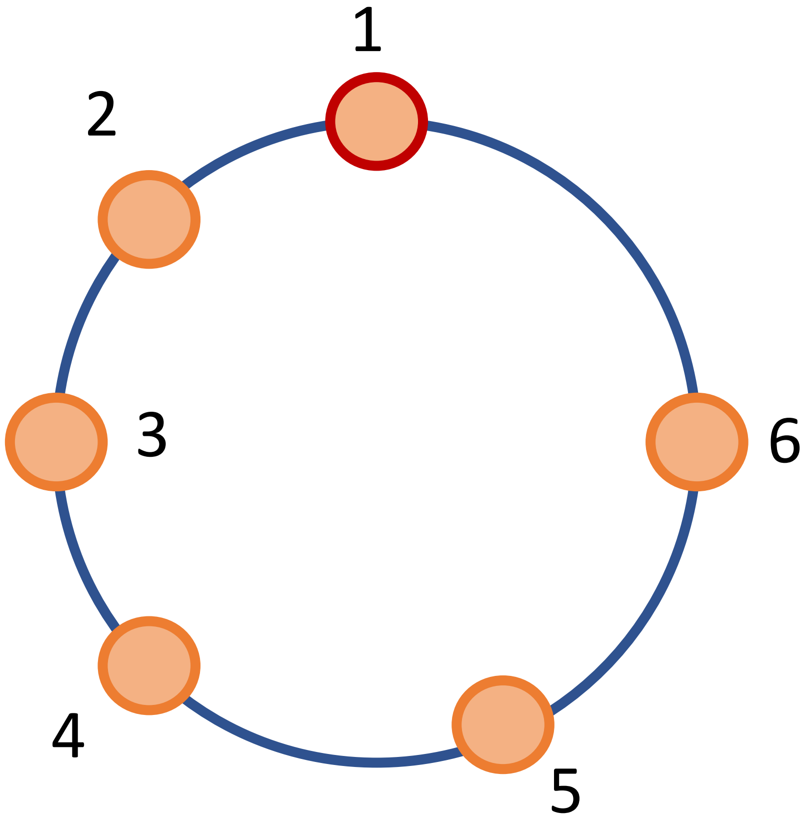
Example

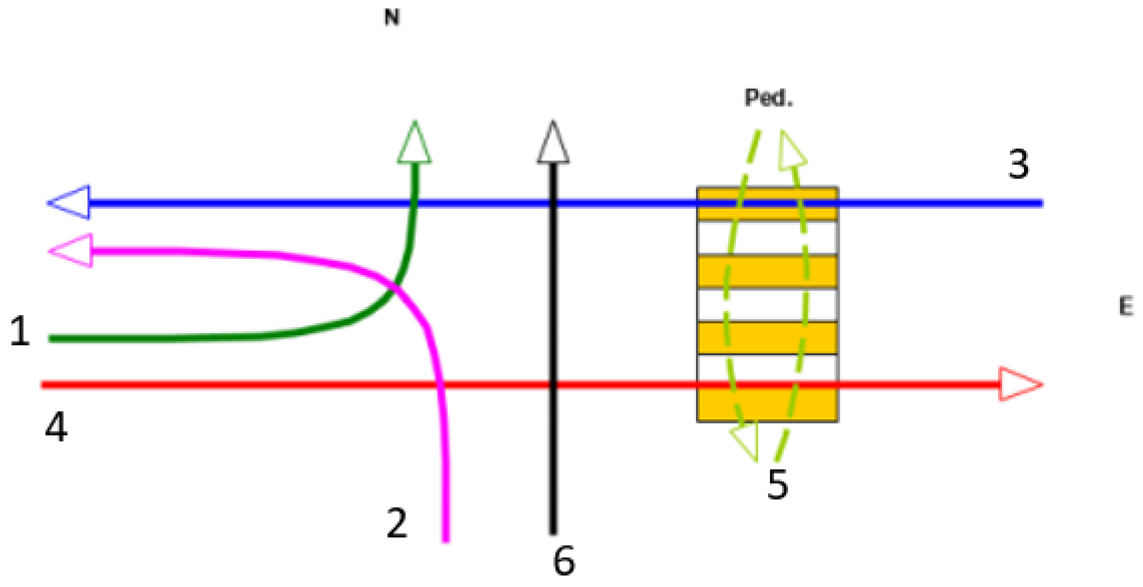
All wants to run

Current turn: **1**

Requests:

[7 , **8** , **9** , 4 , 5 , 6]

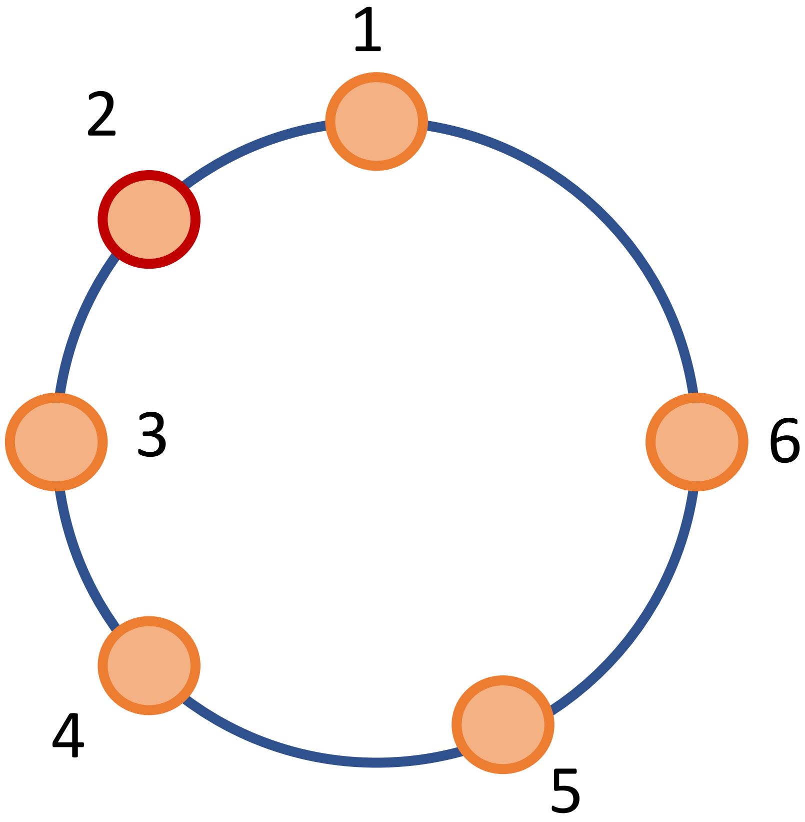




Example

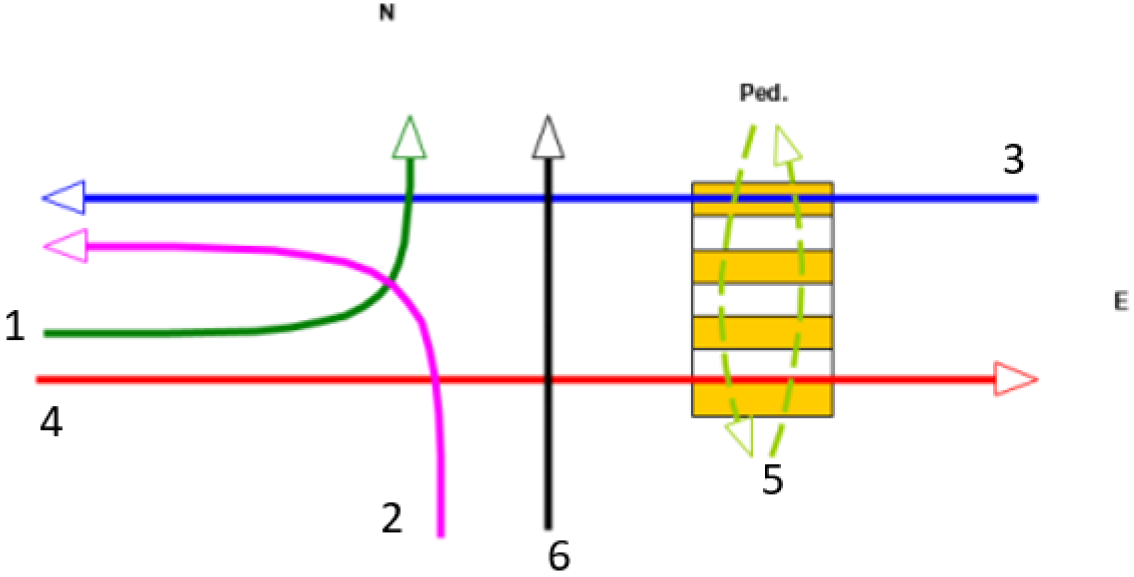
All wants to run

Current turn: **2**



Requests:

[**7** , 8 , 9 , **4** , 5 , 6]



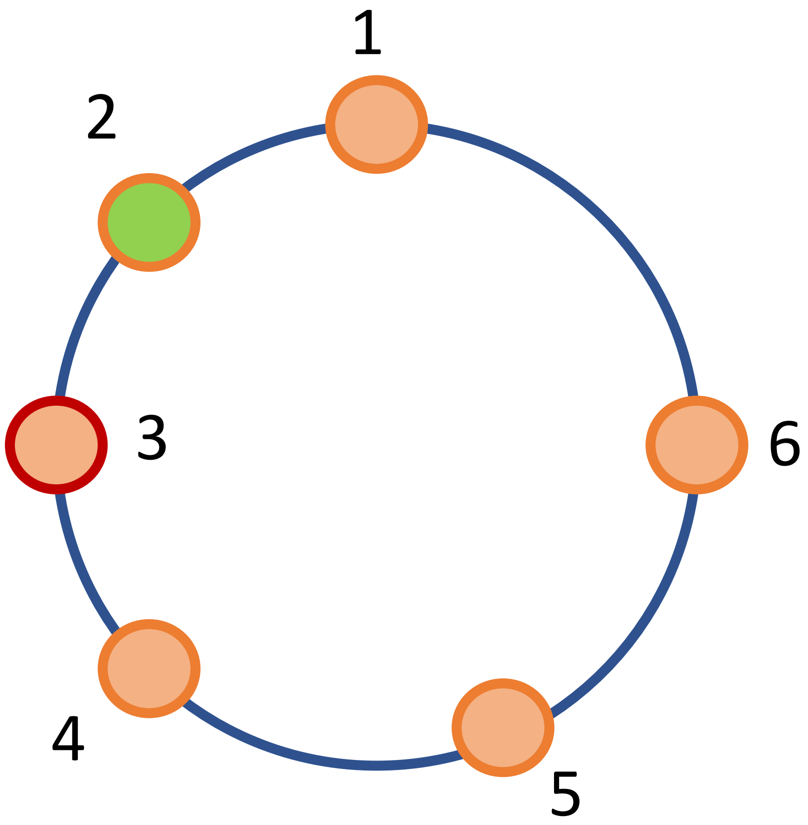
Example

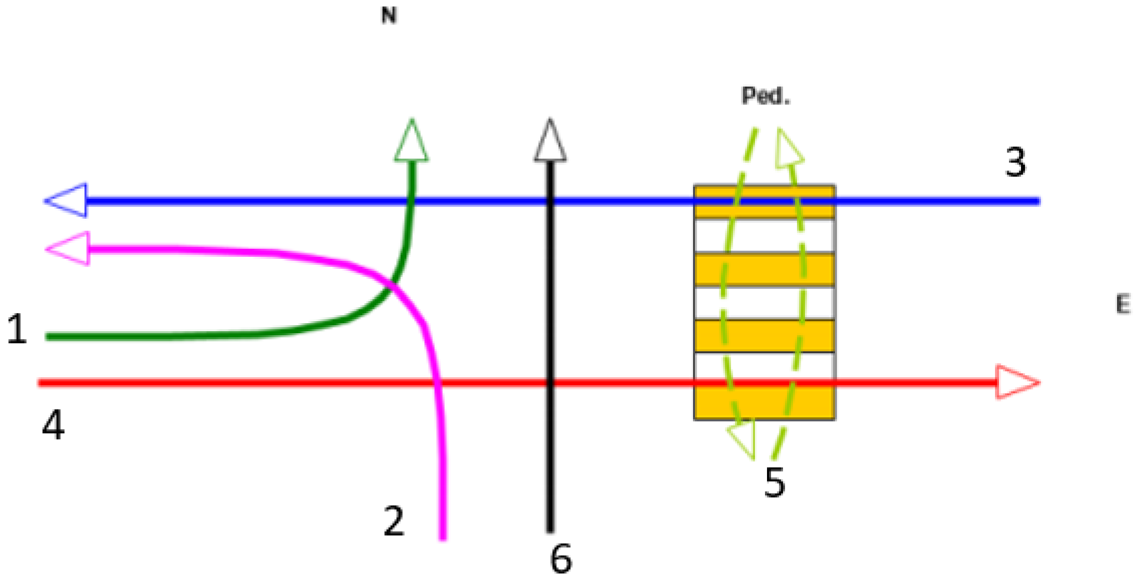
All wants to run

Current turn: **3**

Requests:

[**7** , ∞ , 9 , 4 , **5** , **6**]





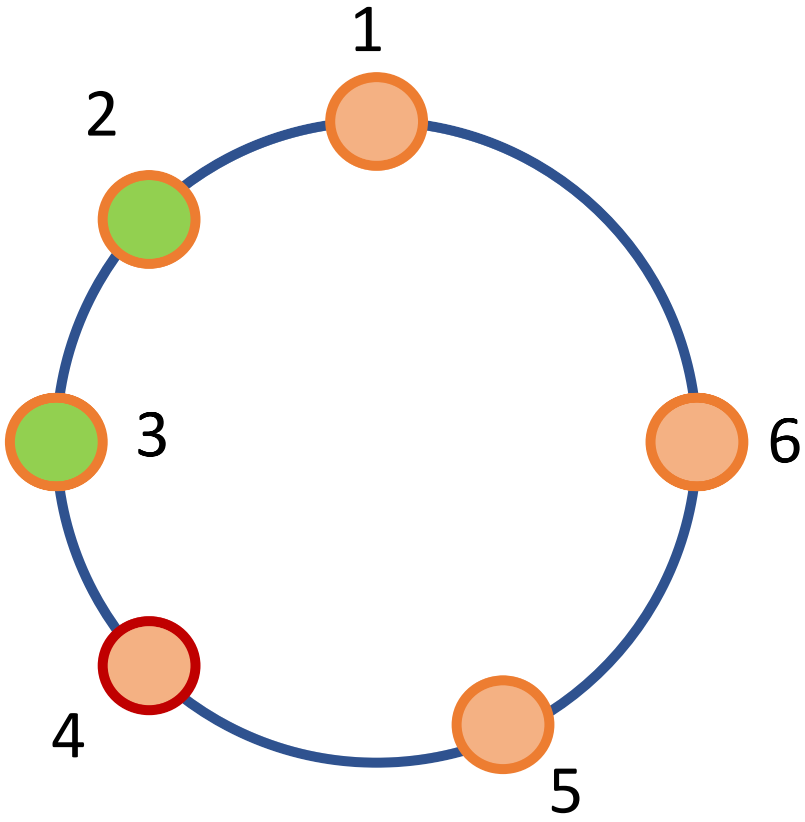
Example

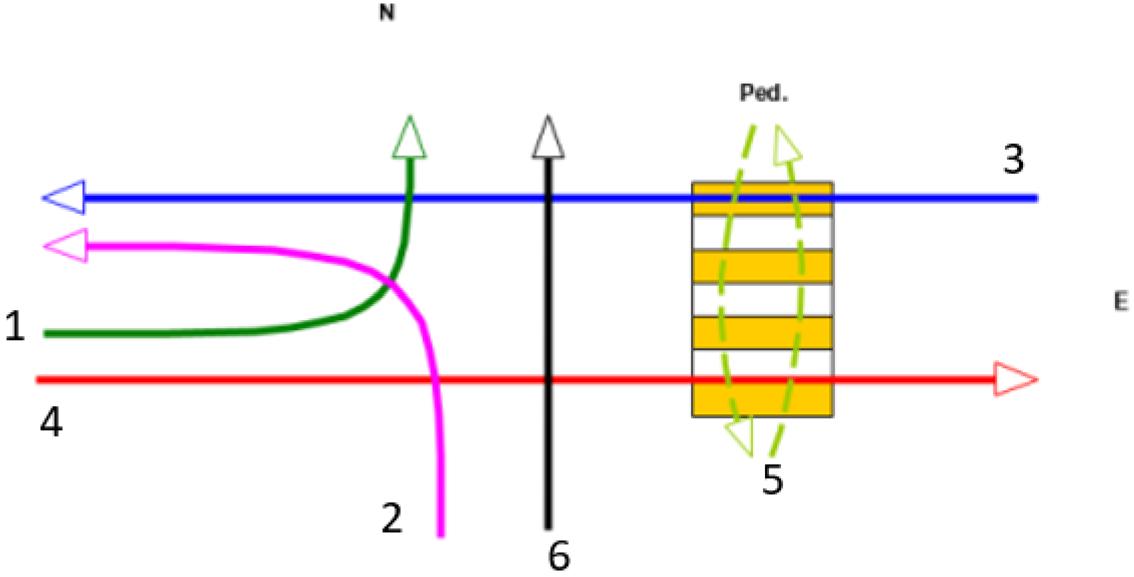
All wants to run

Current turn: **4**

Requests:

[7 , ∞ , ∞ , <u>4</u> , **5** , **6**]





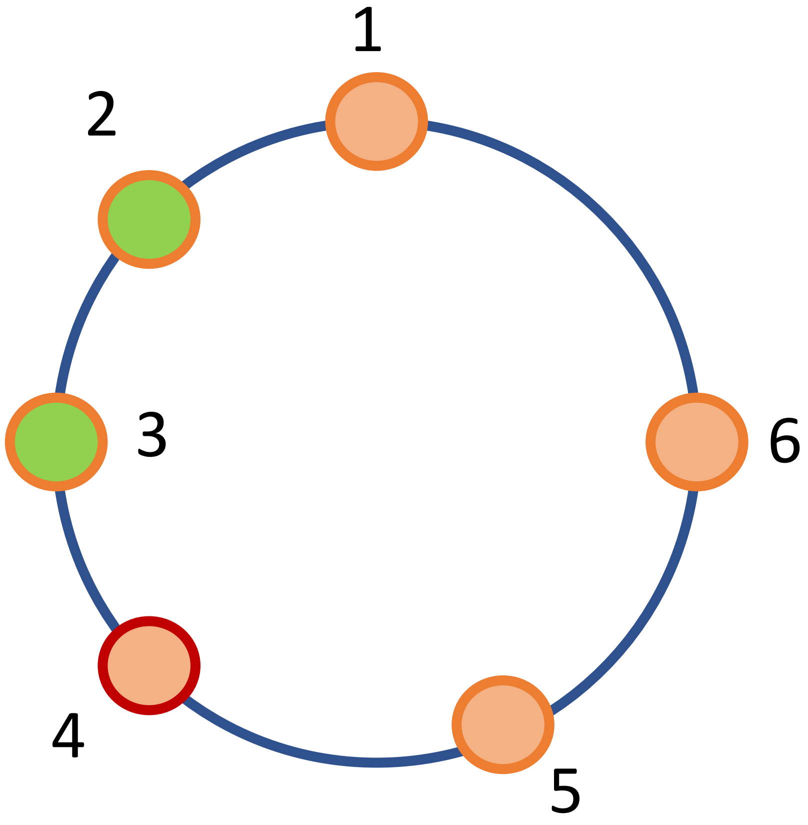
Example

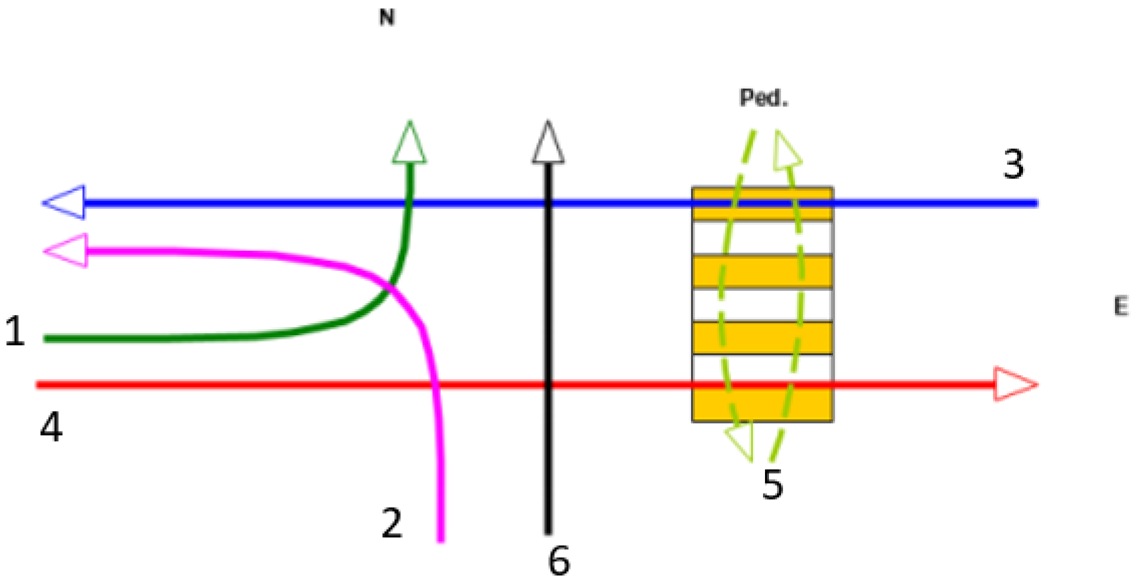
All wants to run

Current turn: **4**

Requests:

[7 , ∞ , ∞ , 10 , **11** , **12**]

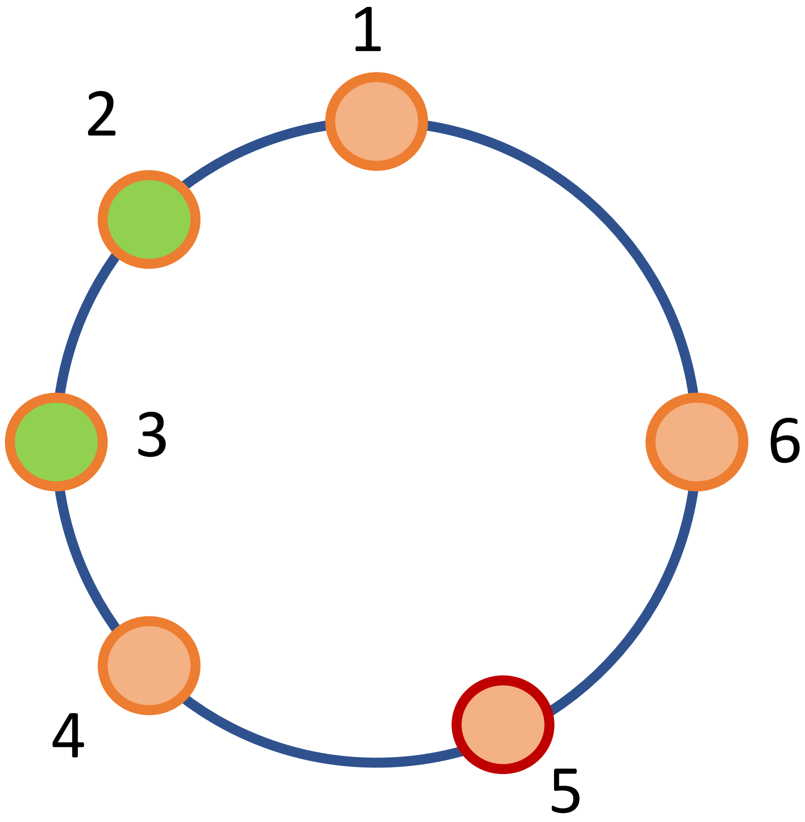




Example

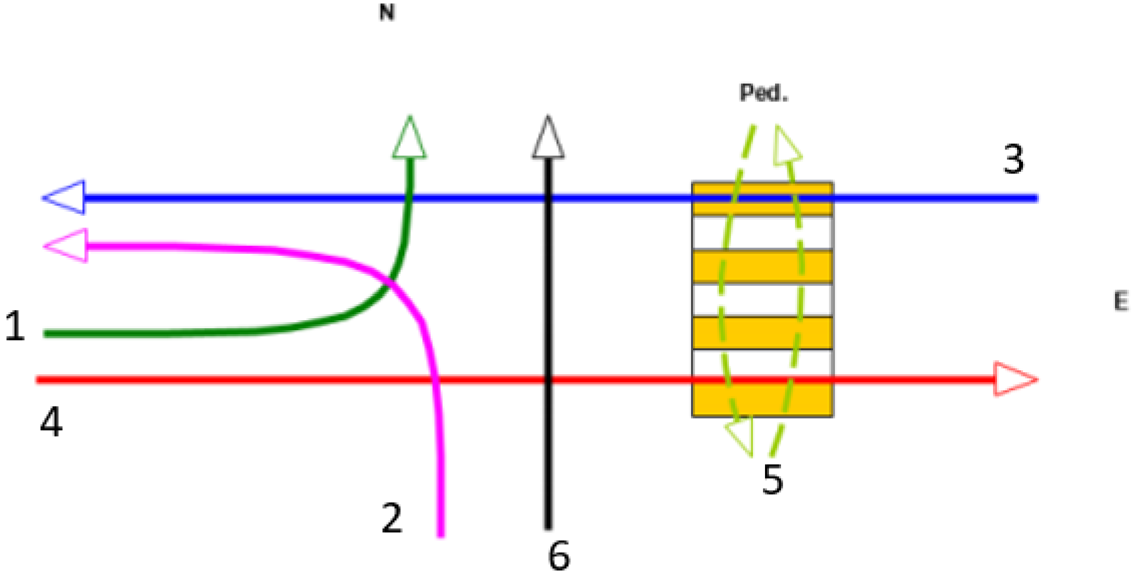
All wants to run

Current turn: **5**



Requests:

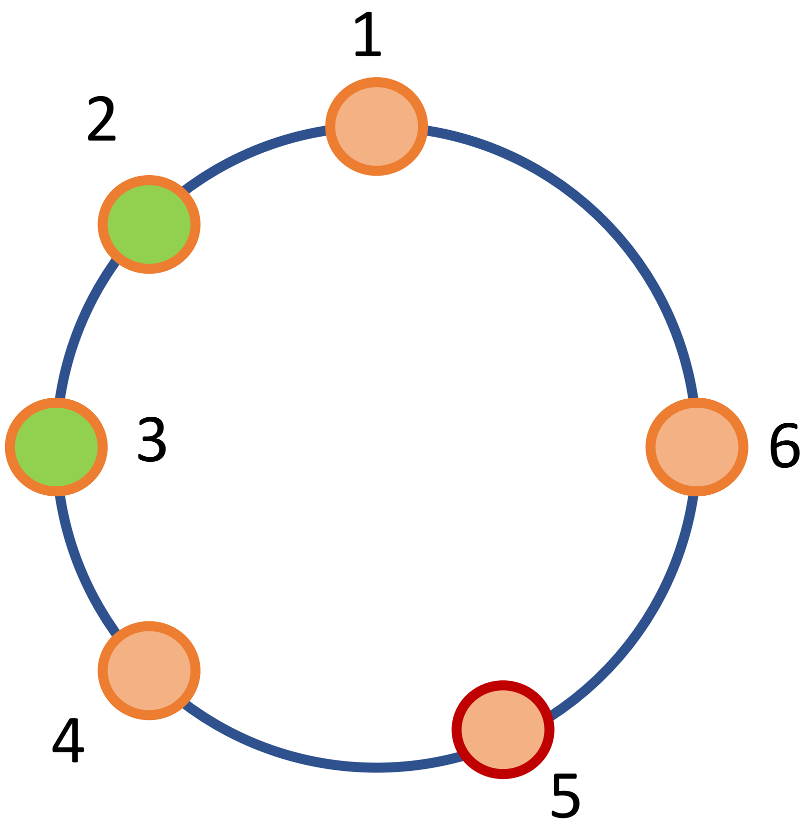
[7 , ∞ , ∞ , **10** , 11 , 12]



Example

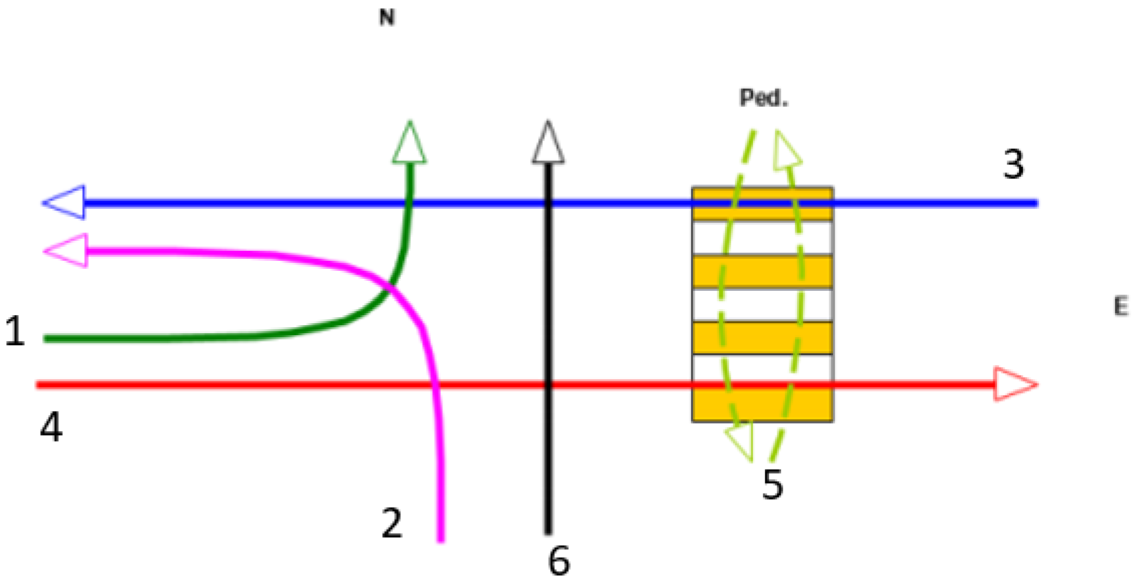
All wants to run

Current turn: **5**



Requests:

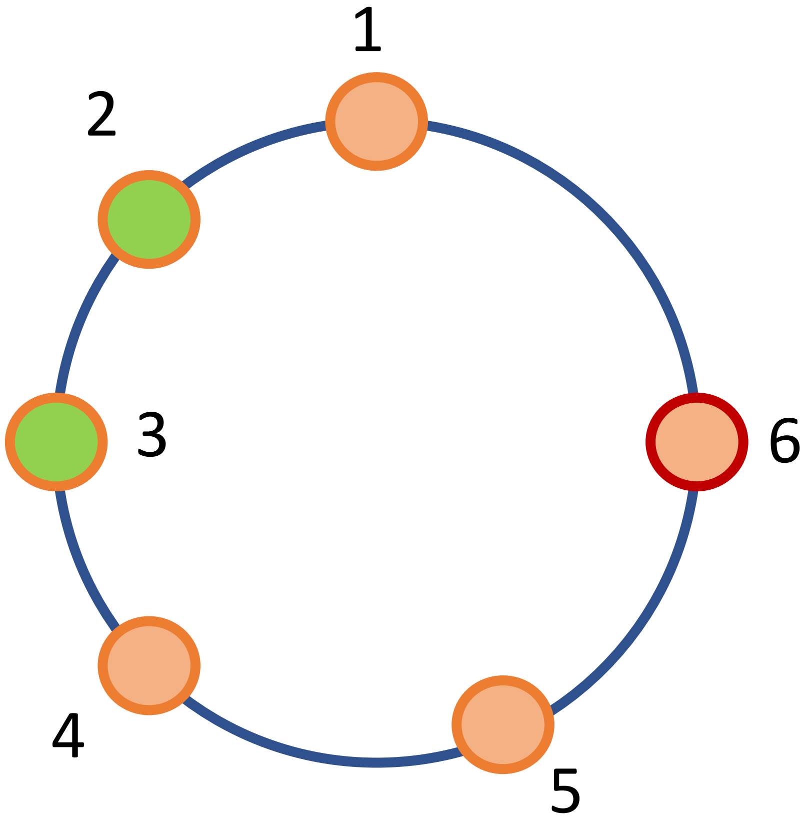
[7 , ∞ , ∞ , **16** , 17 , 12]



Example

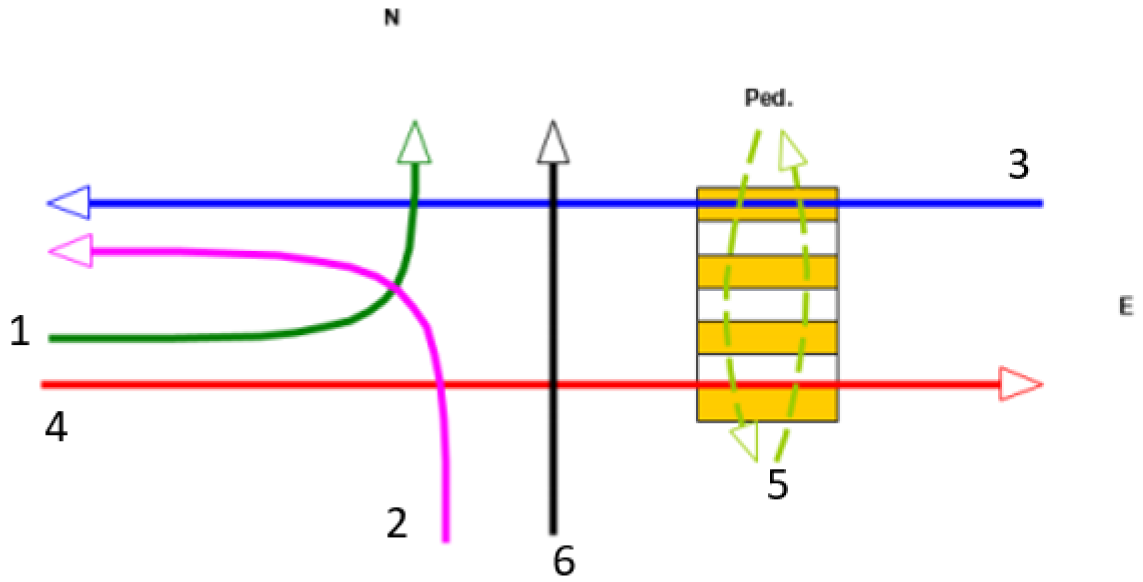
All wants to run

Current turn: **6**



Requests:

[7 , ∞ , ∞ , **16** , 17 , 12]



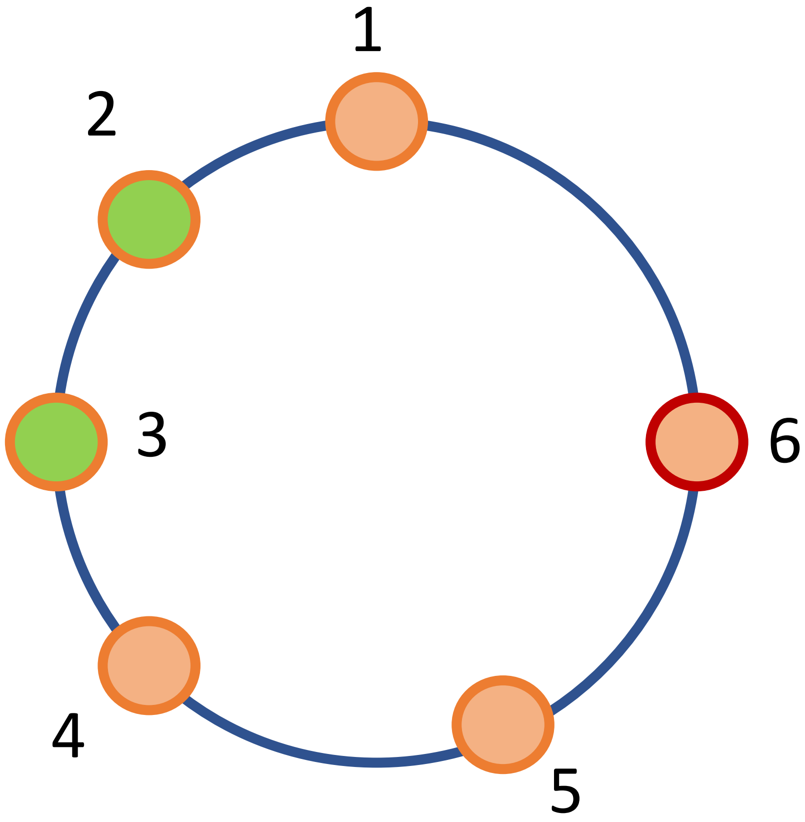
Example

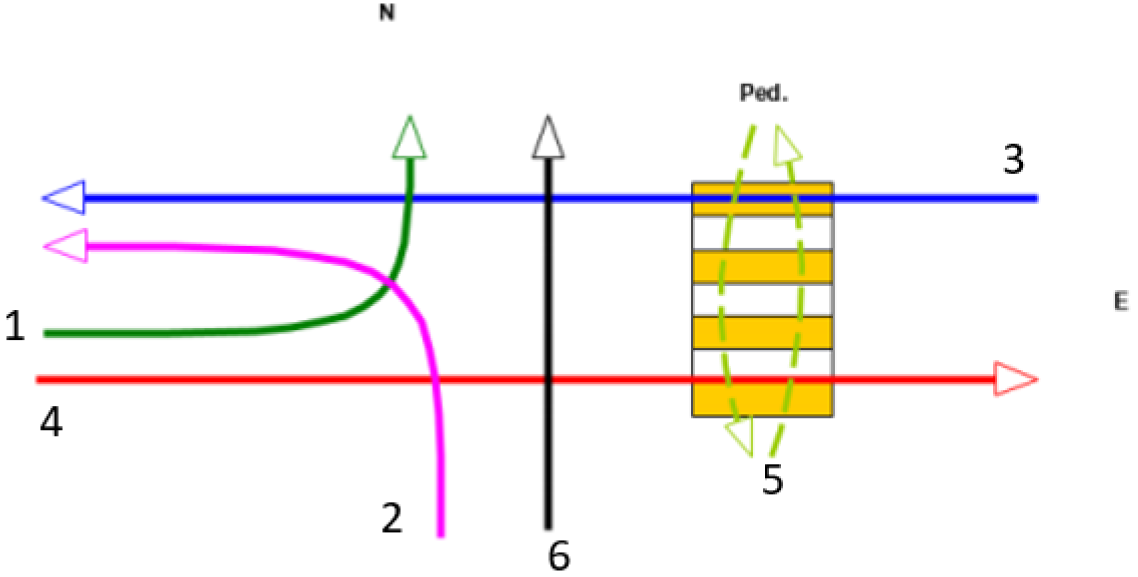
All wants to run

Current turn: **6**

Requests:

[7 , ∞ , ∞ , **22** , 17 , <u>18</u>]





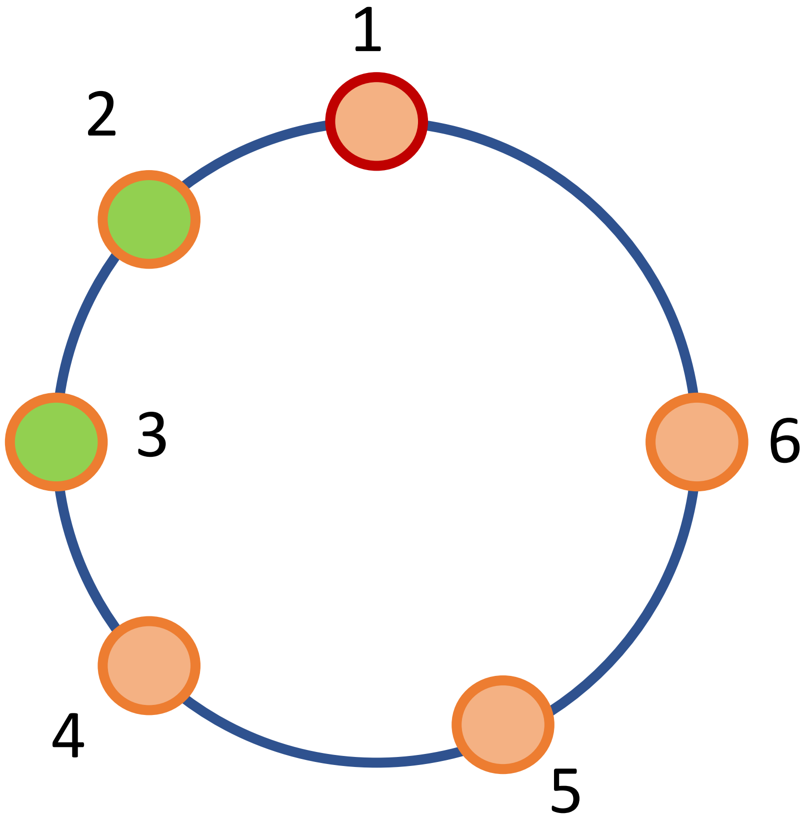
Example

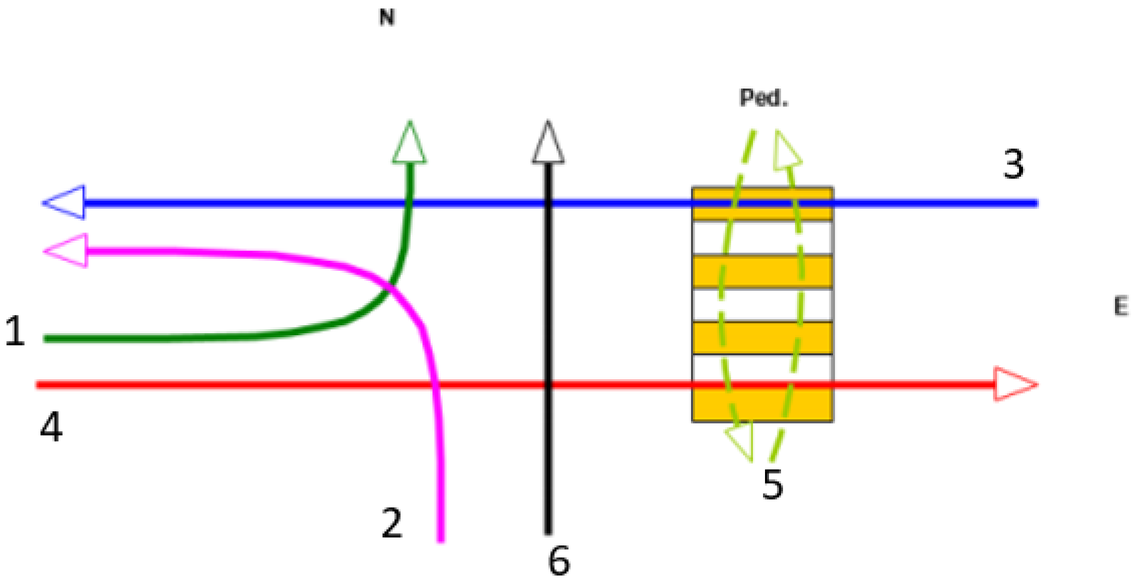
All wants to run

Current turn: **1**

Requests:

[7 , ∞ , ∞ , 22 , 17 , 18]





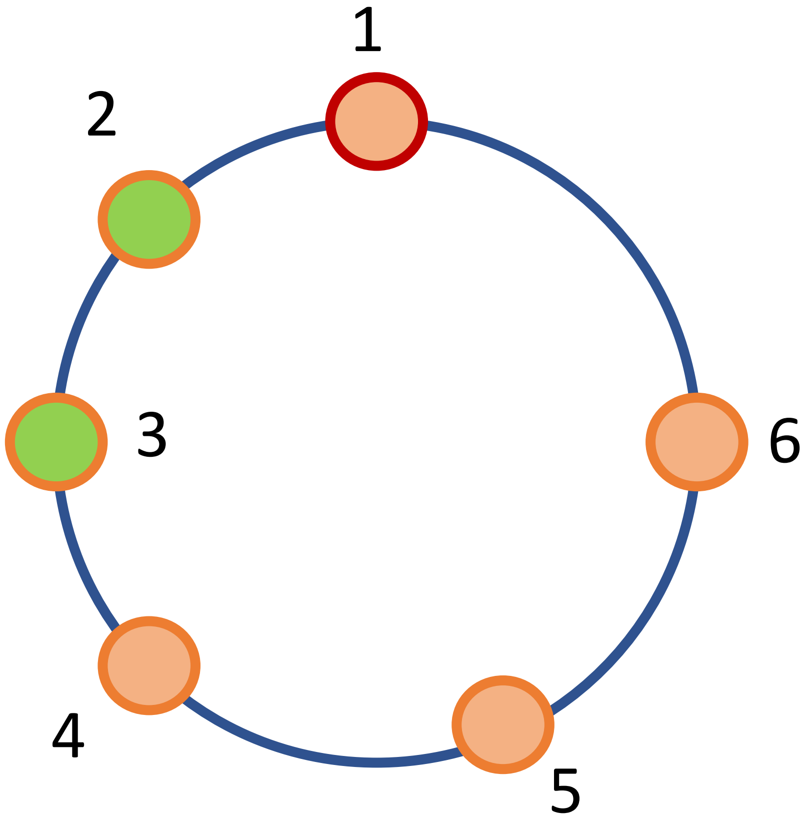
Example

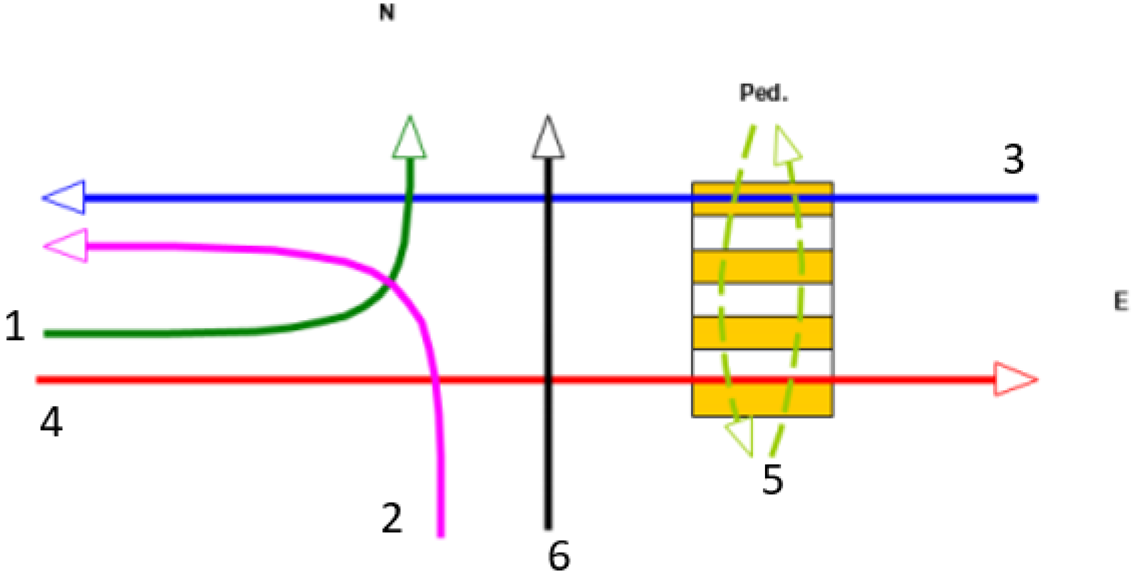
All wants to run

Current turn: **1**

Requests:

[<u>13</u> , ∞ , ∞ , 22 , 17 , 18]





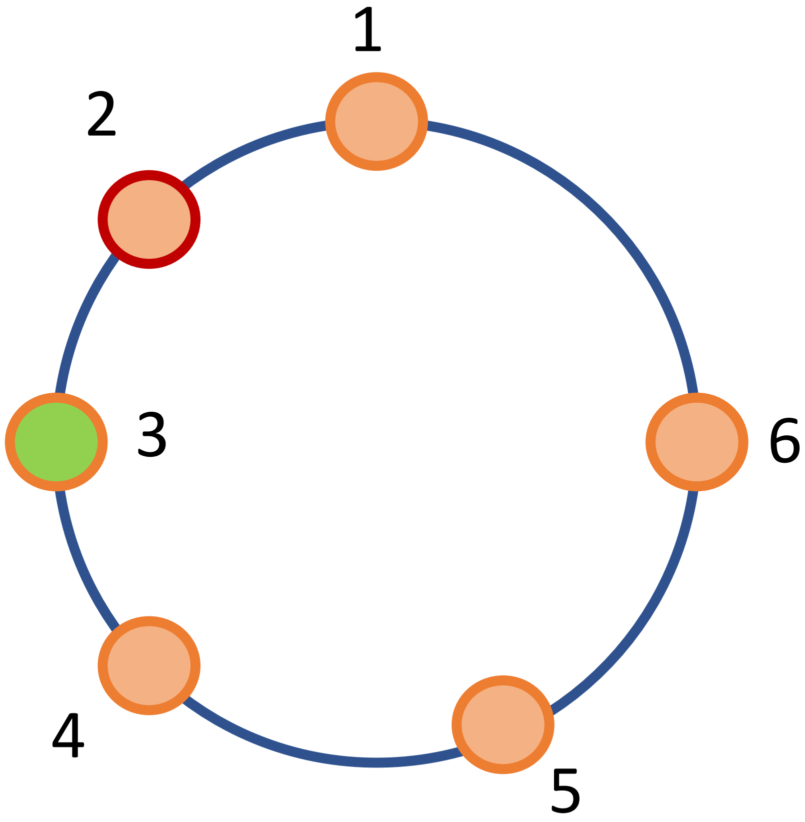
Example

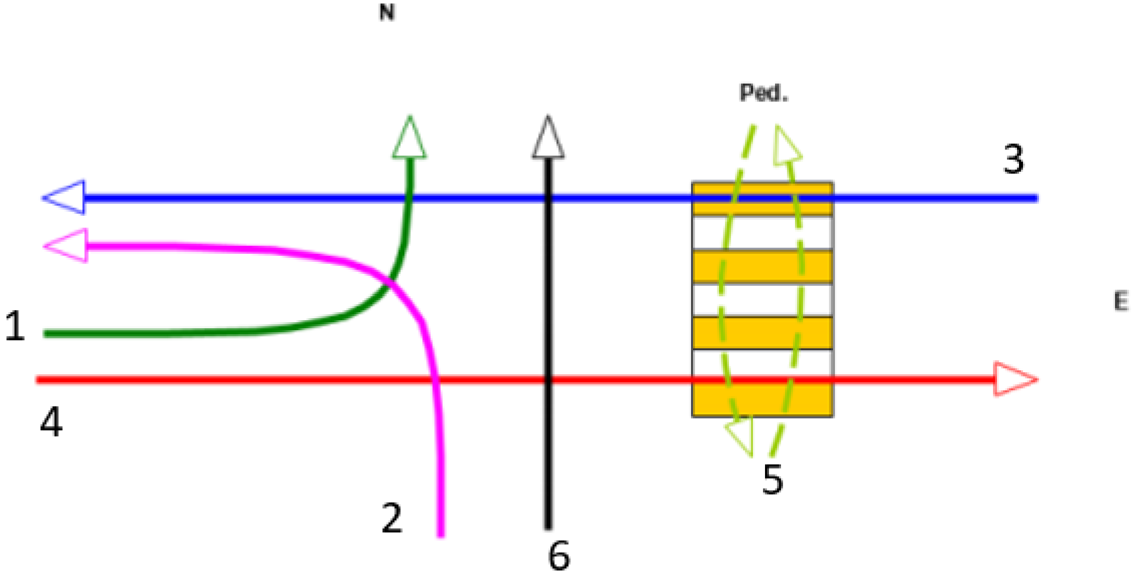
All wants to run

Current turn: **2**

Requests:

[13 , 0 , ∞ , 22 , 17 , 18]

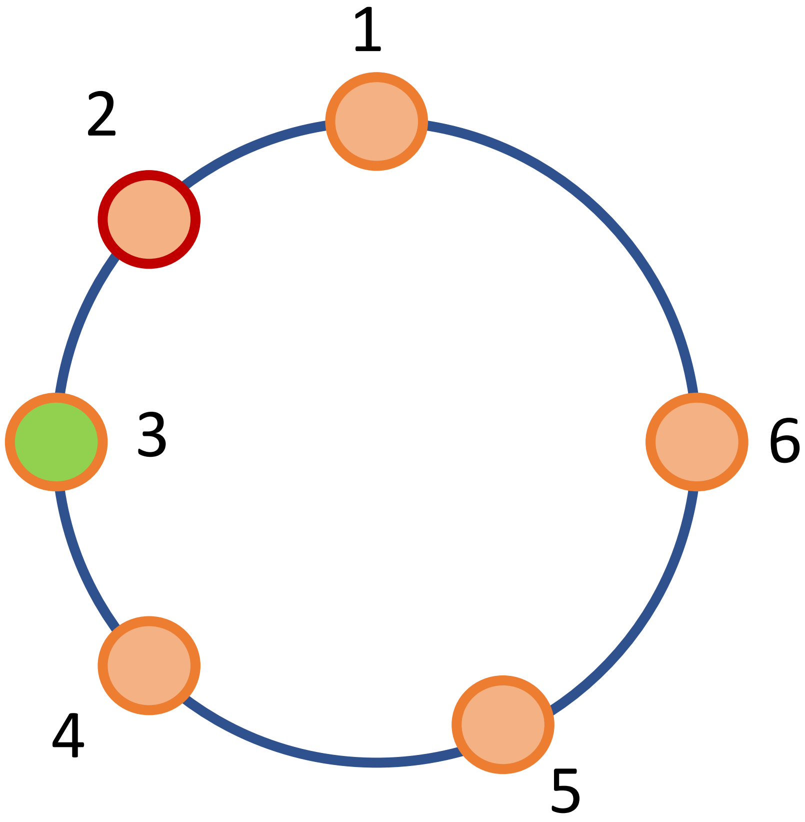




Example

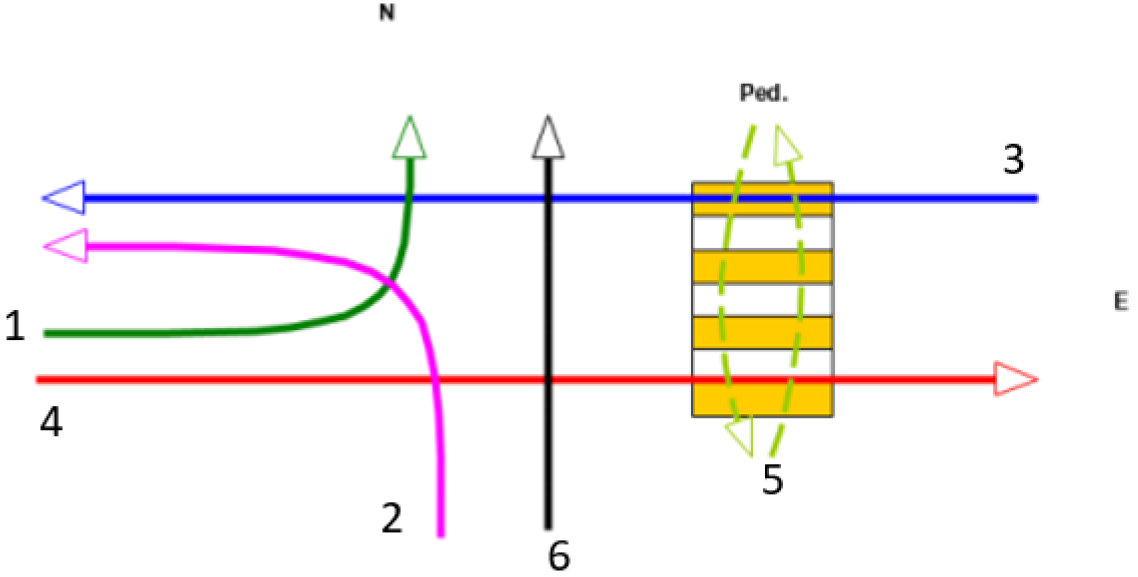
All wants to run

Current turn: **2**



Requests:

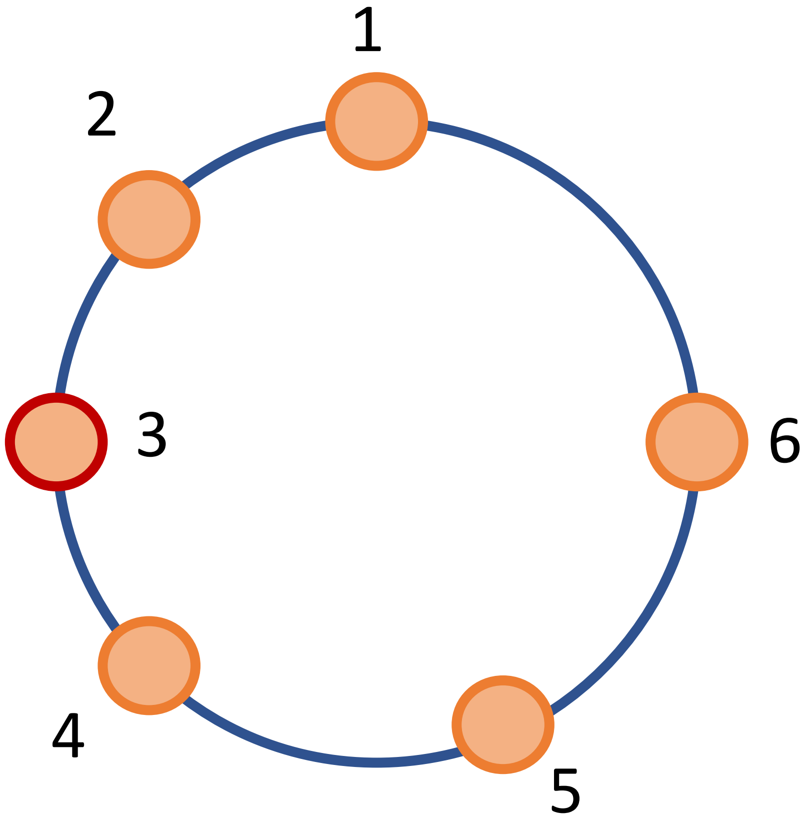
[13 , 2 , ∞ , 22 , 17 , 18]



Example

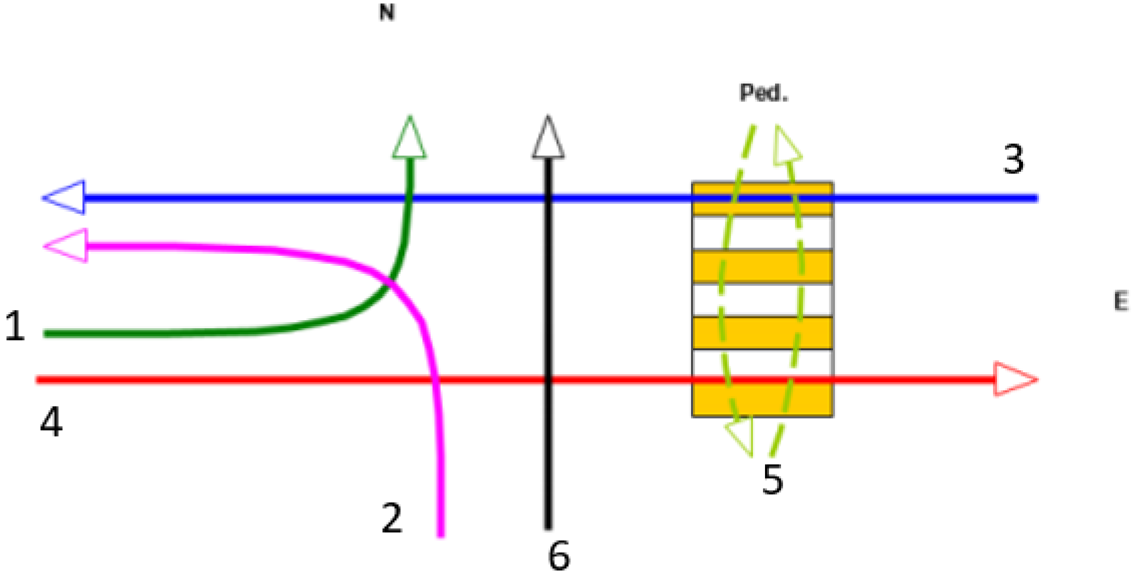
All wants to run

Current turn: **3**



Requests:

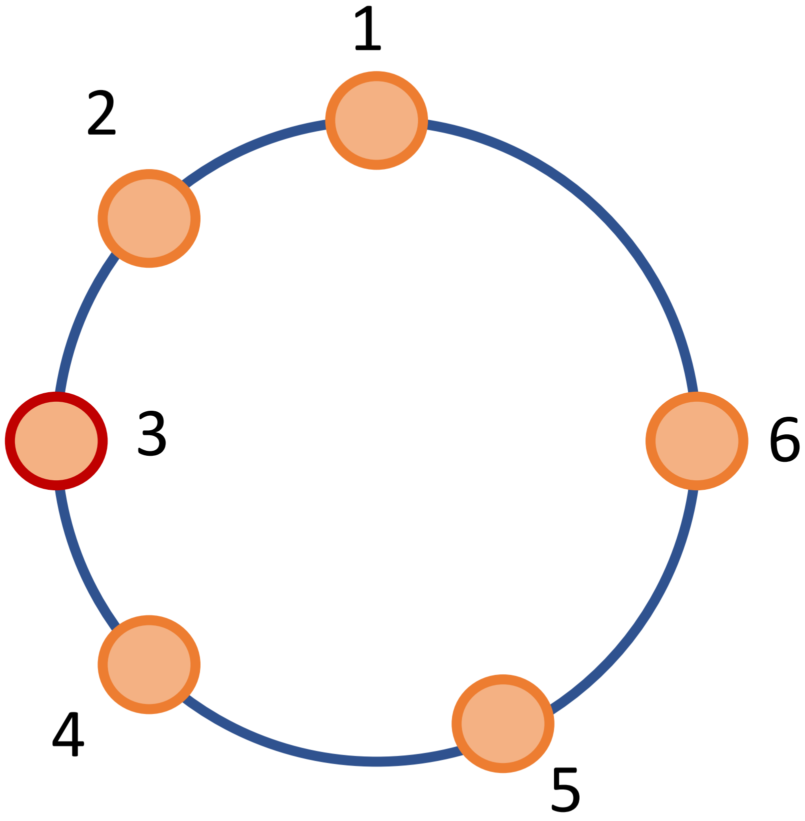
[13 , 2 , 0 , 22 , 17 , 18]



Example

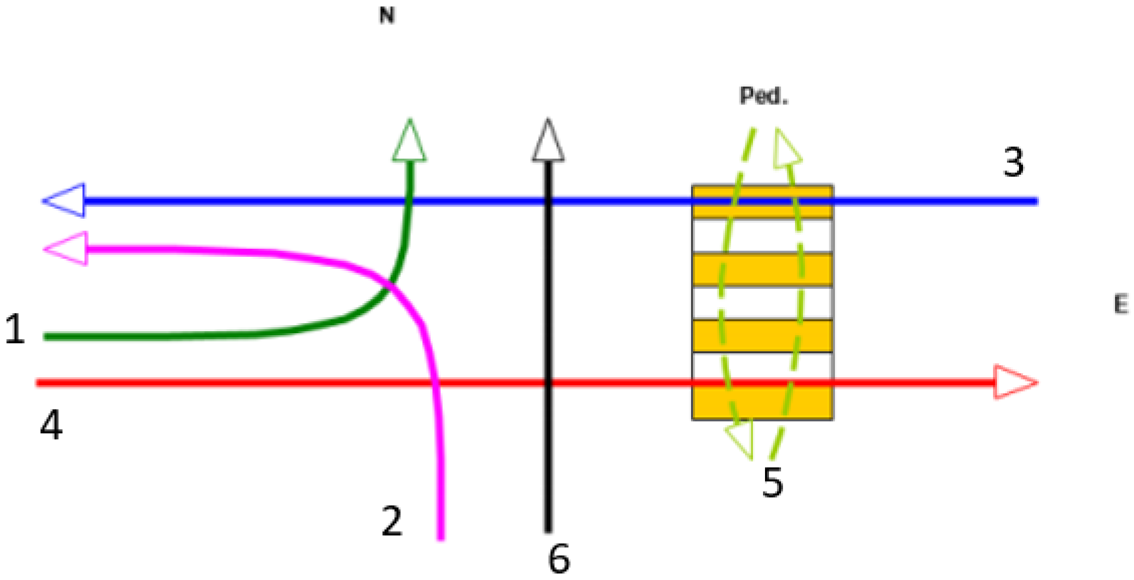
All wants to run

Current turn: **3**



Requests:

[13 , 2 , 3 , 22 , 17 , 18]



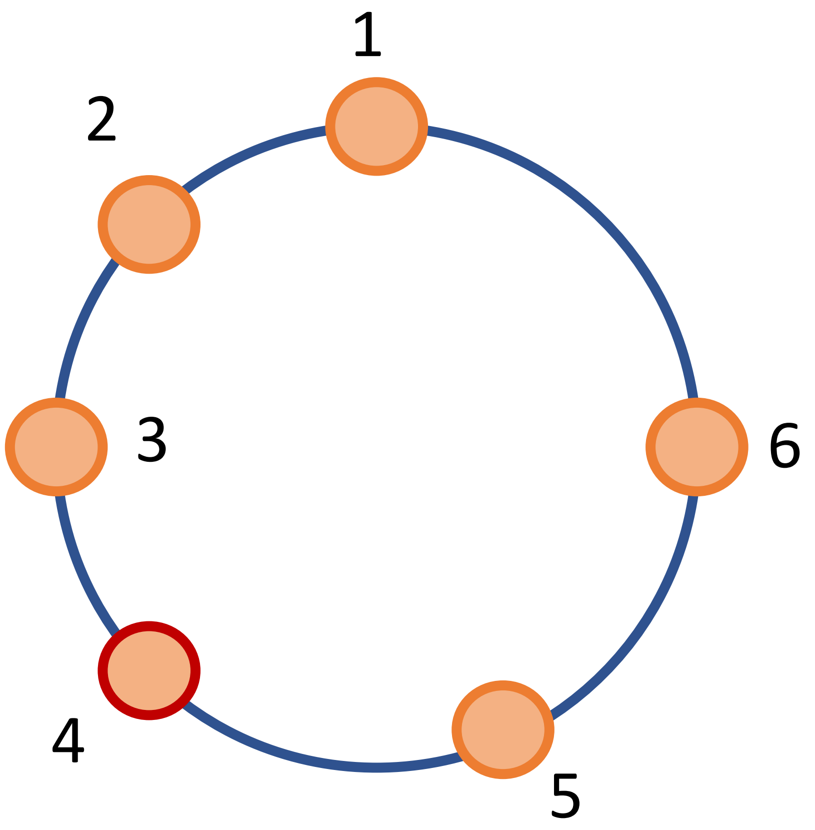
Example

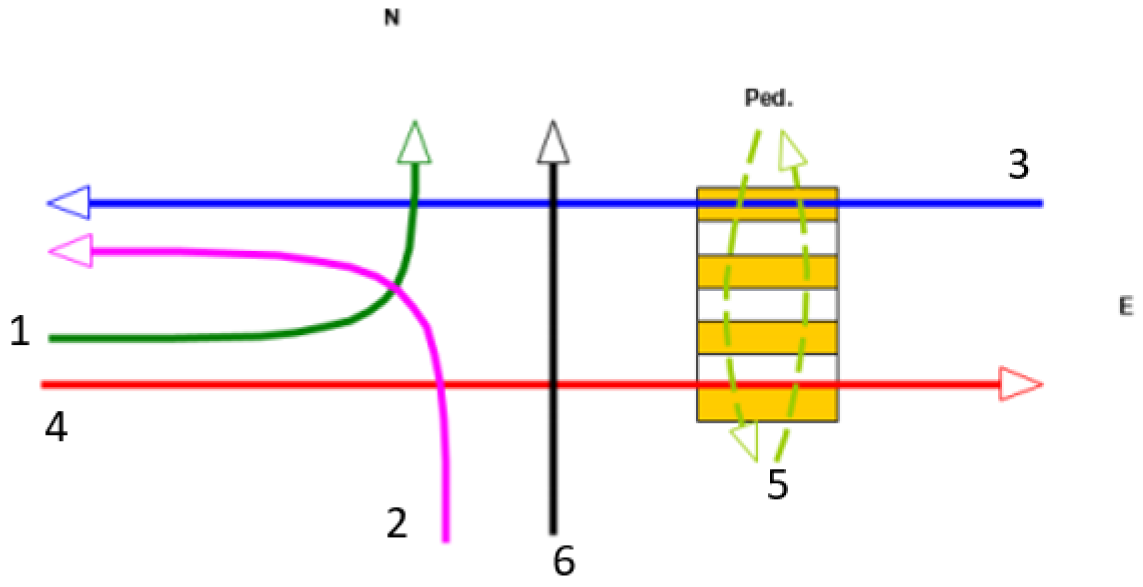
All wants to run

Current turn: **4**

Requests:

[13 , **2** , 3 , 22 , **17** , **18**]

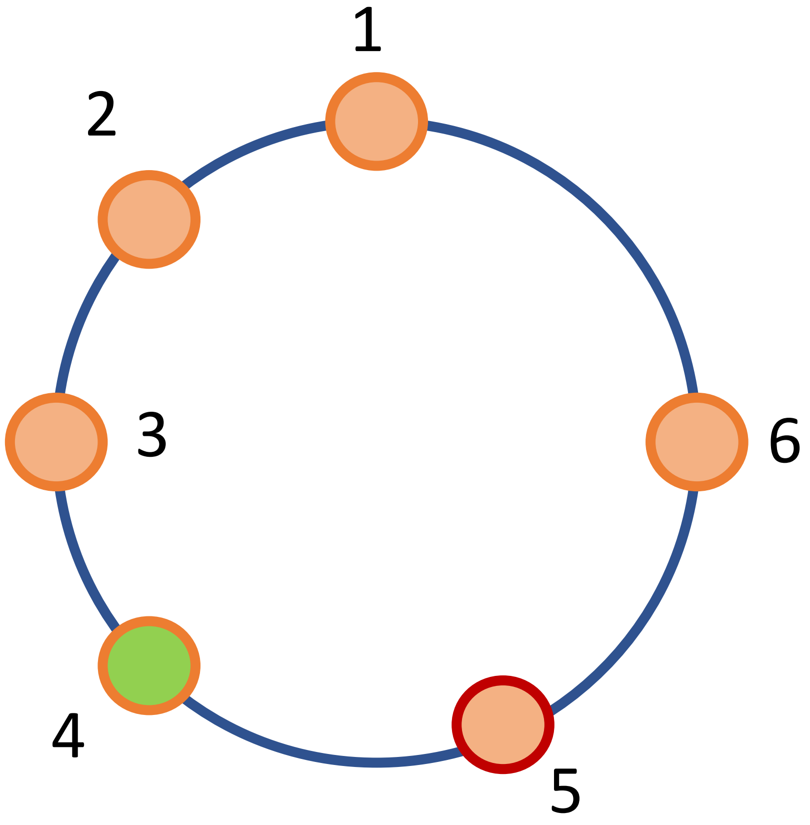




Example

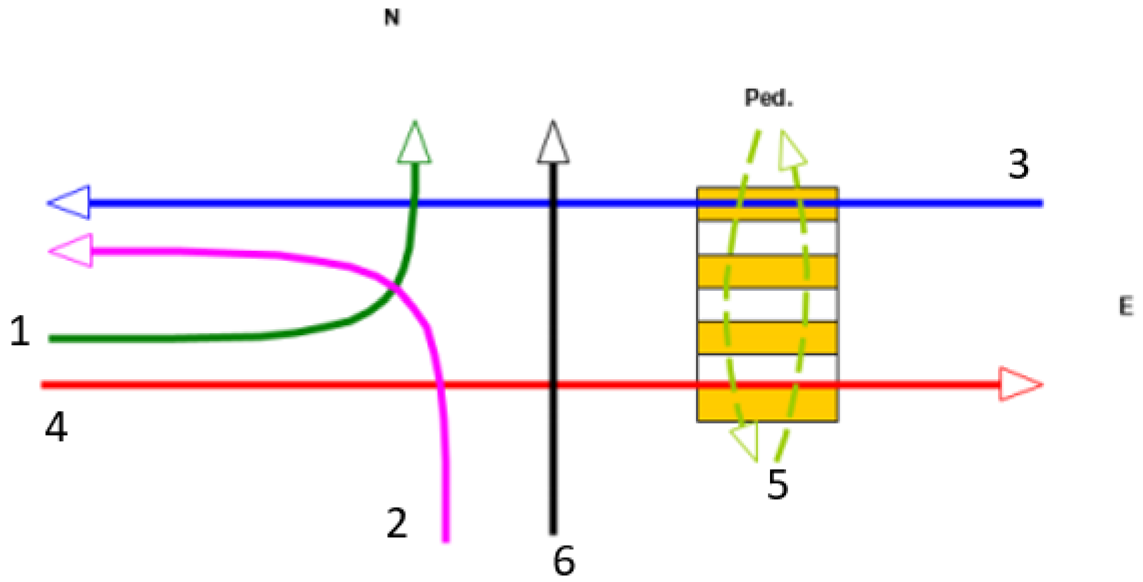
All wants to run

Current turn: **5**



Requests:

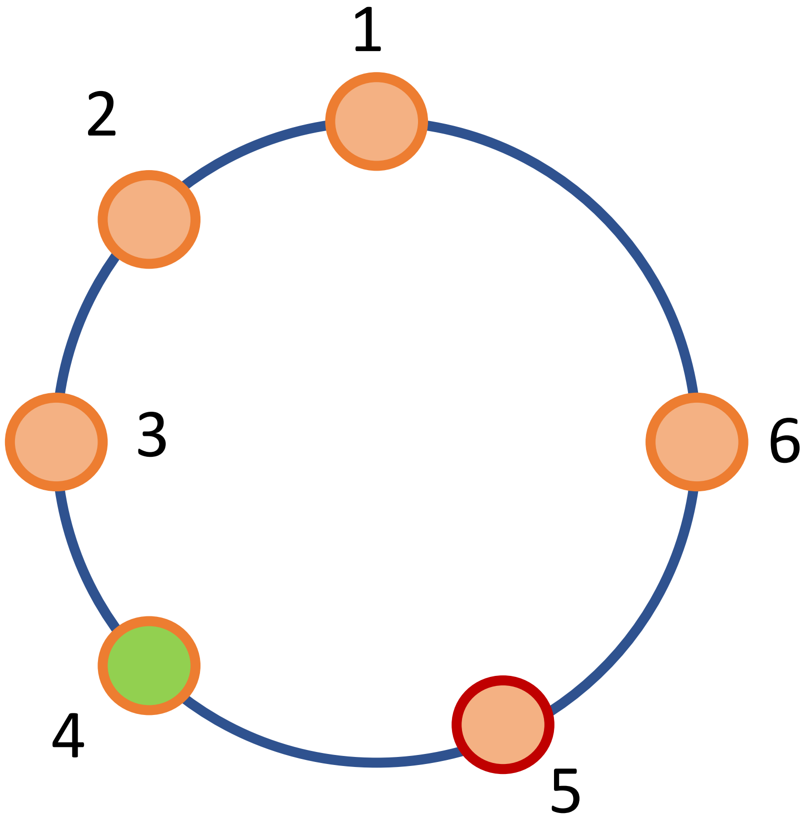
[13 , 2 , **3** , ∞ , <u>17</u> , 18]



Example

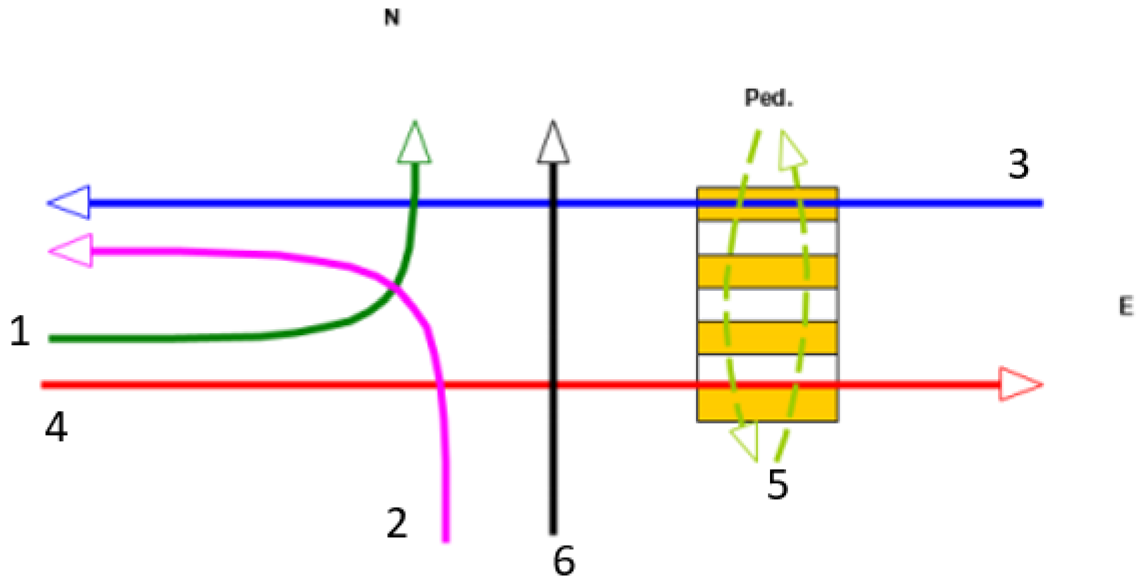
All wants to run

Current turn: **5**



Requests:

[13 , 2 , **9** , ∞ , 23 , 18]



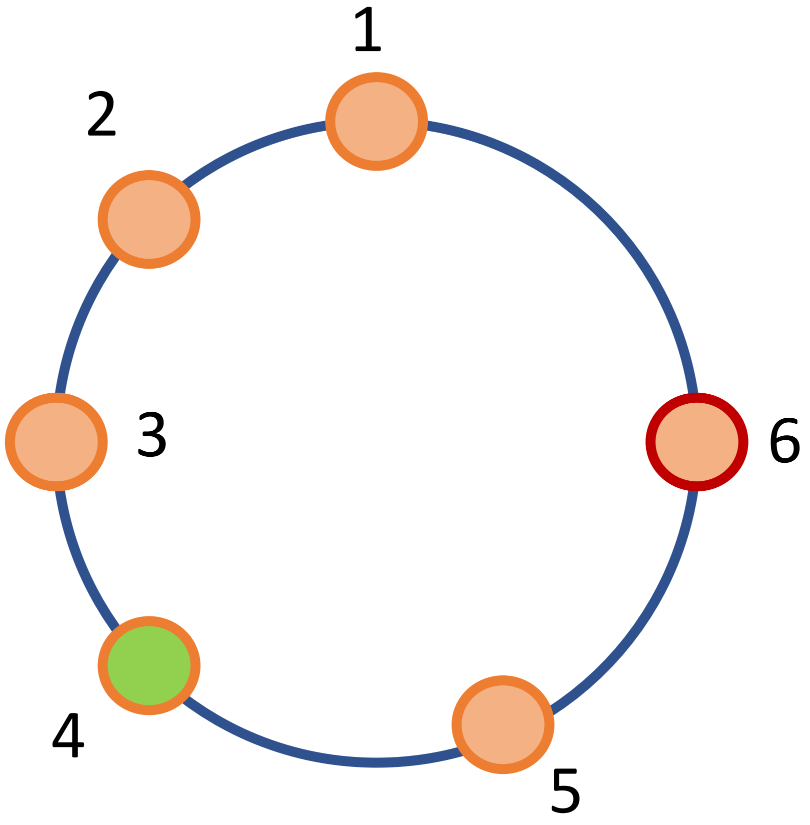
Example

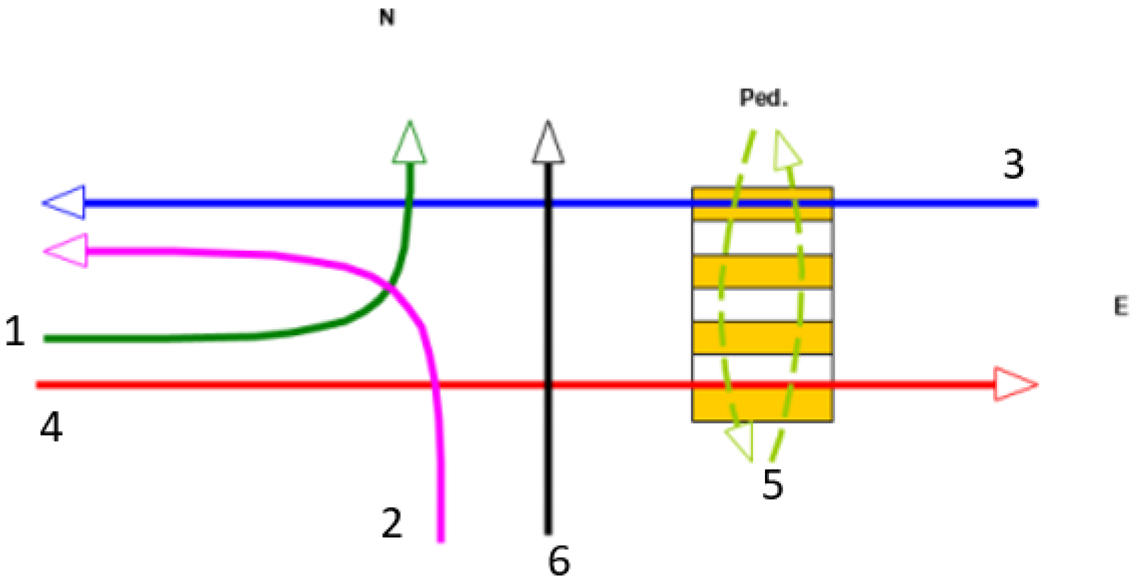
All wants to run

Current turn: **6**

Requests:

[13 , 2 , **9** , ∞ , **23** , 18]





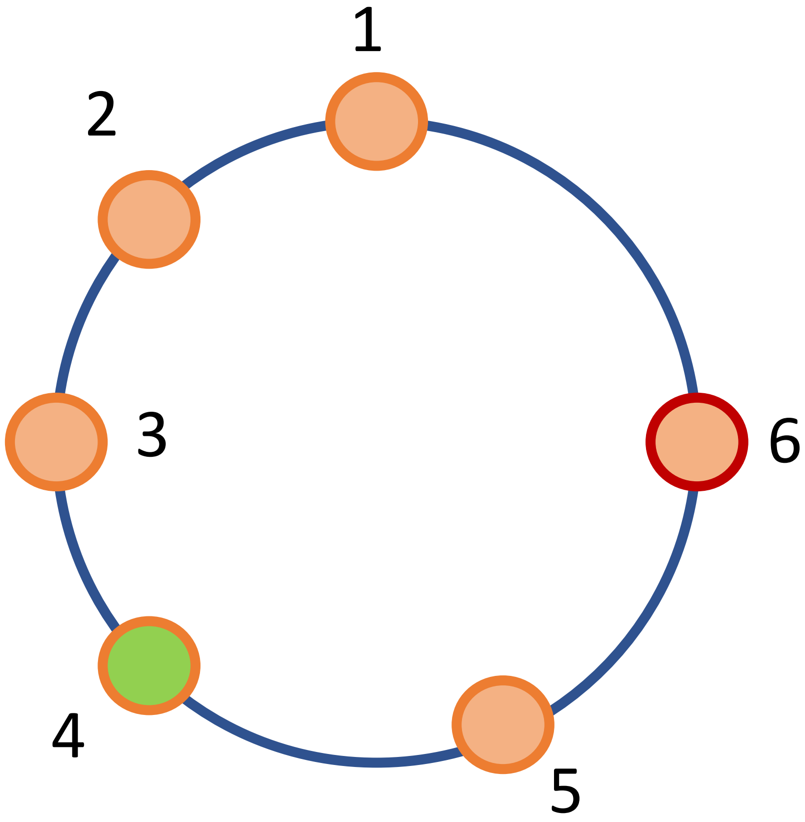
Example

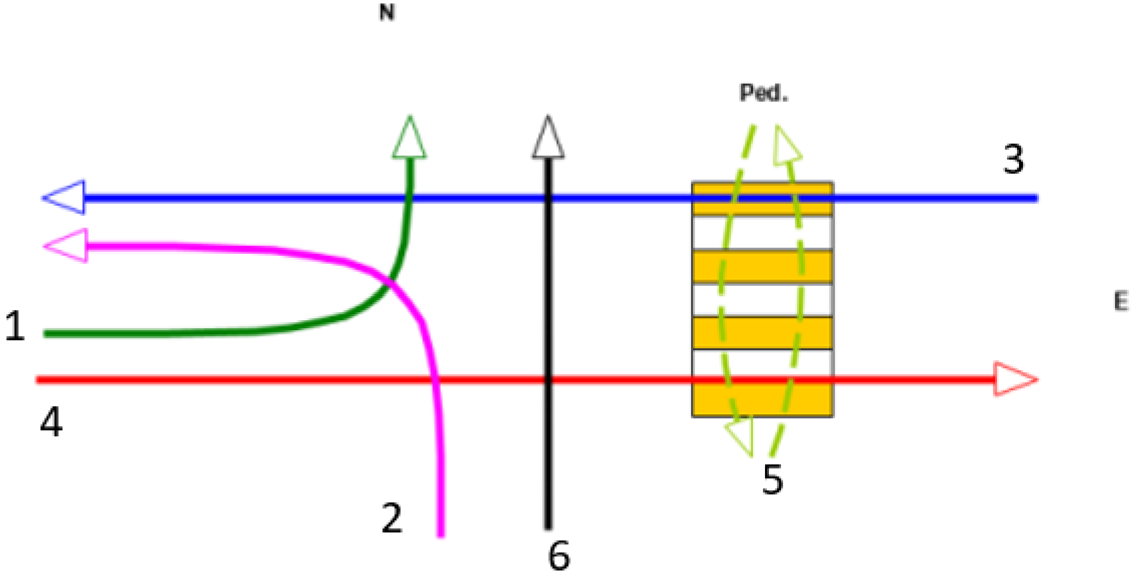
All wants to run

Current turn: **6**

Requests:

[13 , 3 , **15** , ∞ , 23 , <u>24</u>]





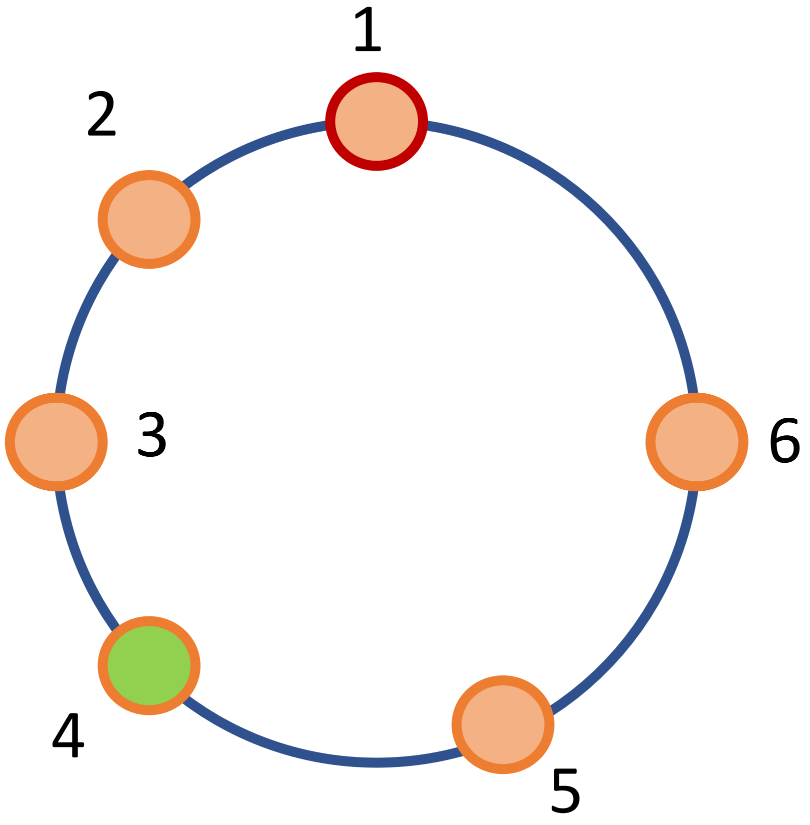
....

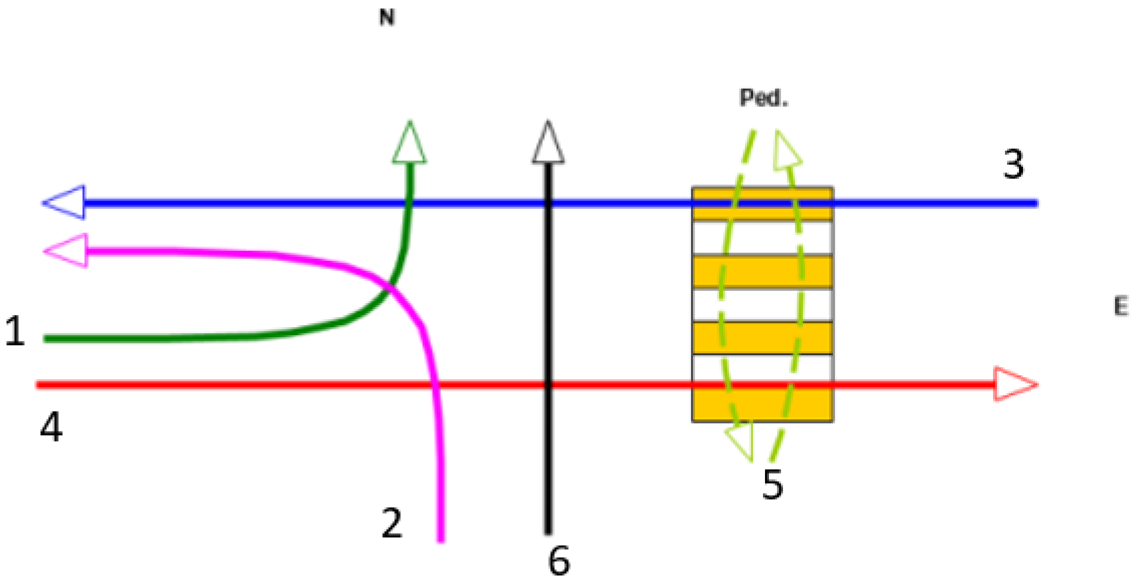
All wants to run

Current turn: **1**

Requests:

[13 , 3 , 15 , ∞ , 23 , 24]





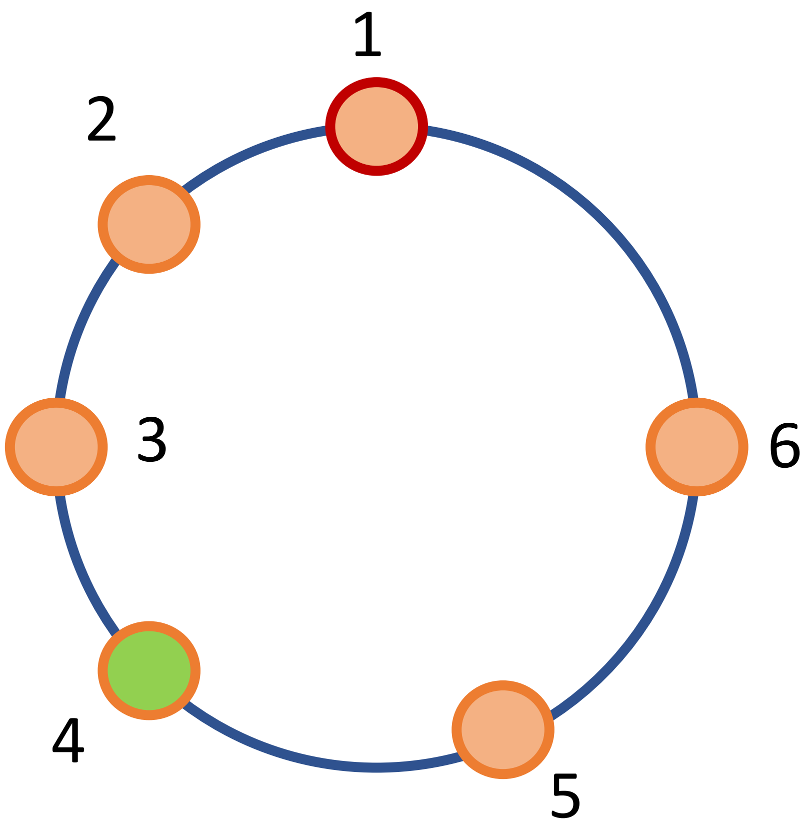
....

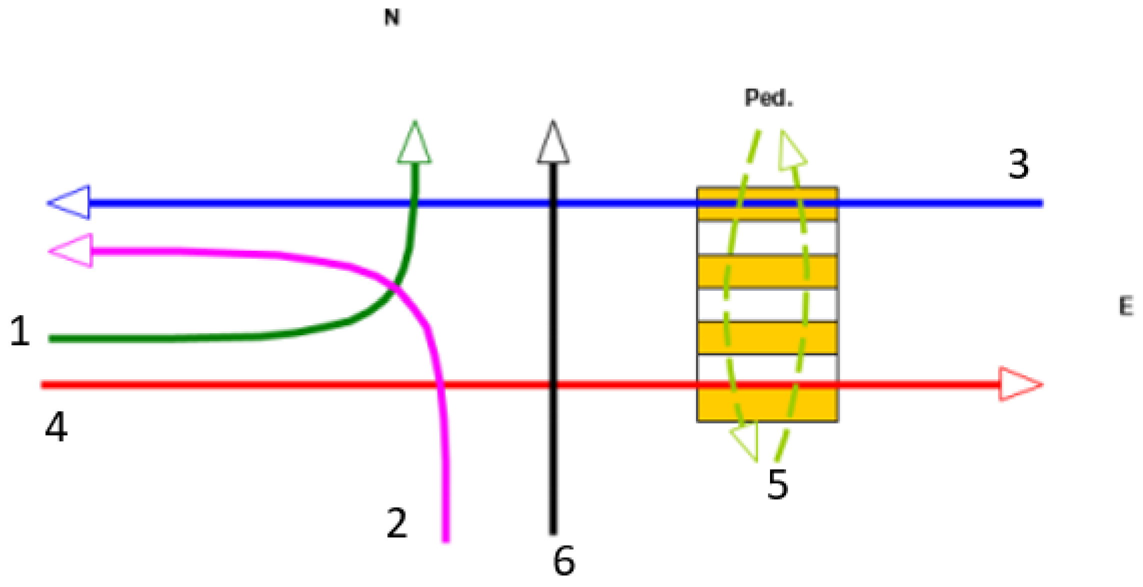
All wants to run

Current turn: **1**

Requests:

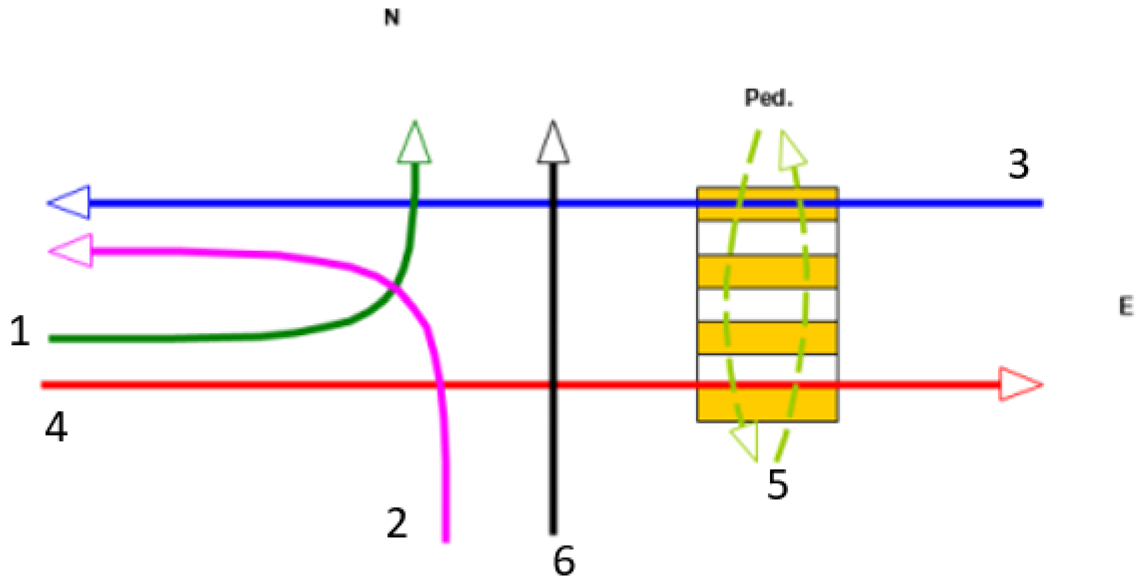
[13 , 3 , 15 , ∞ , 23 , 24]





Validation. Safety

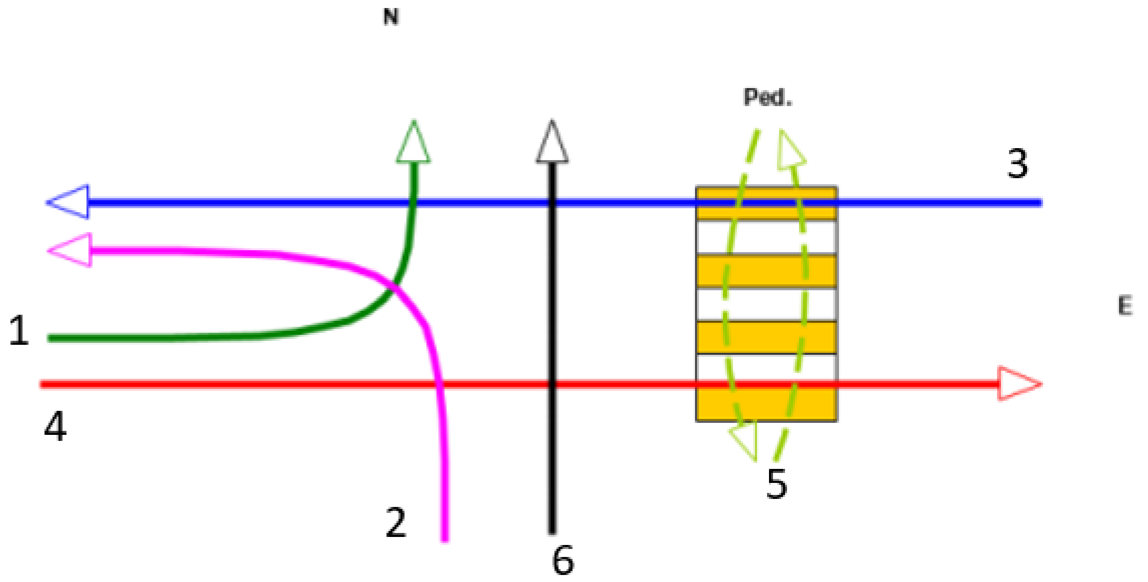
```
[] (!(statuses [i] == true && statuses [j] == true))
```



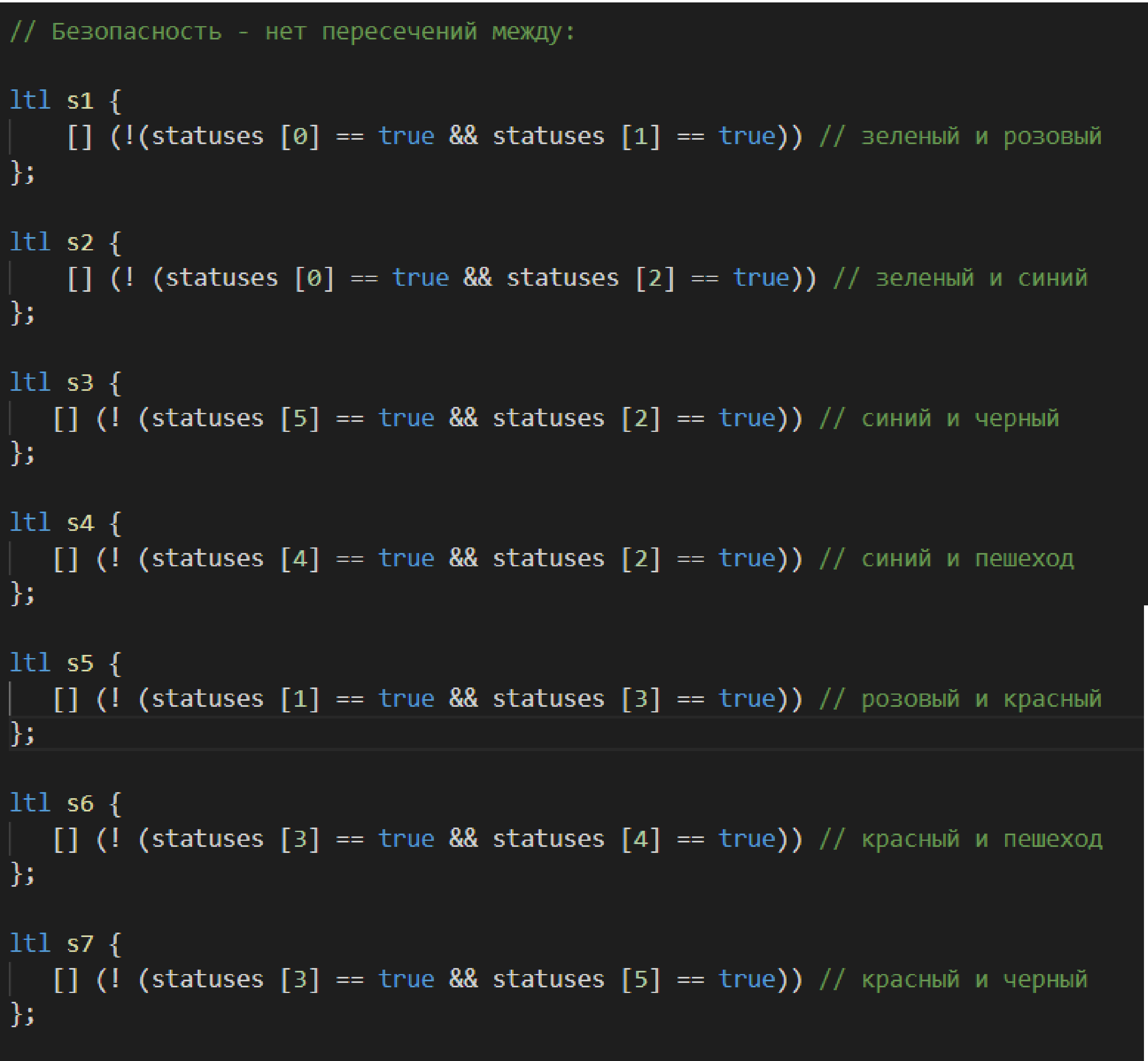
Validation. Safety

```
[] (!(statuses [i] == true && statuses [j] == true))
```

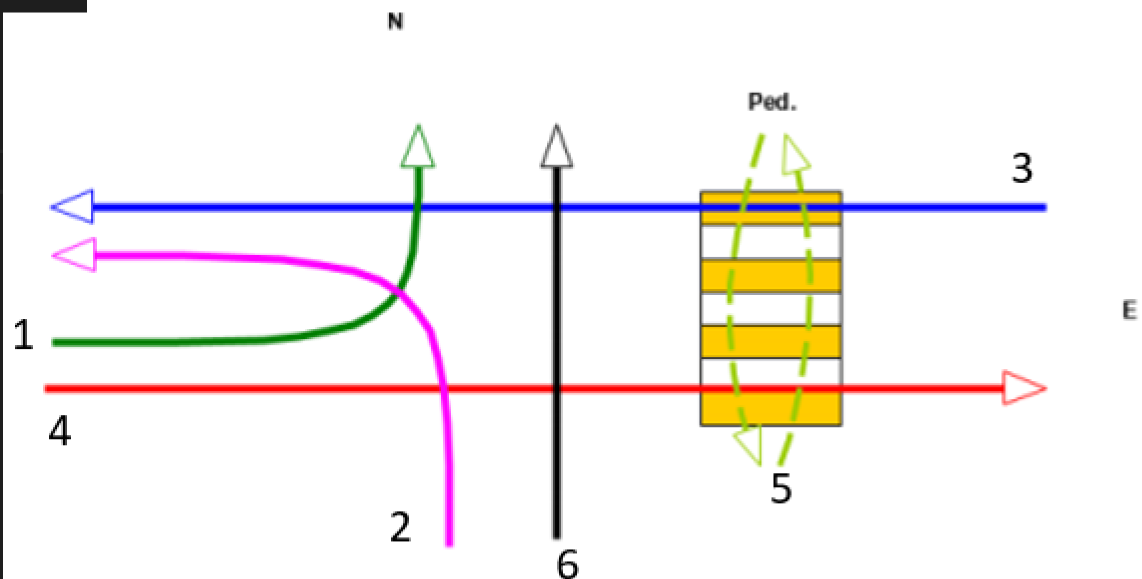
```
ltl s1 {
        // зеленый и розовый
    [] (!(statuses [0] == true && statuses [1] == true))
};
```



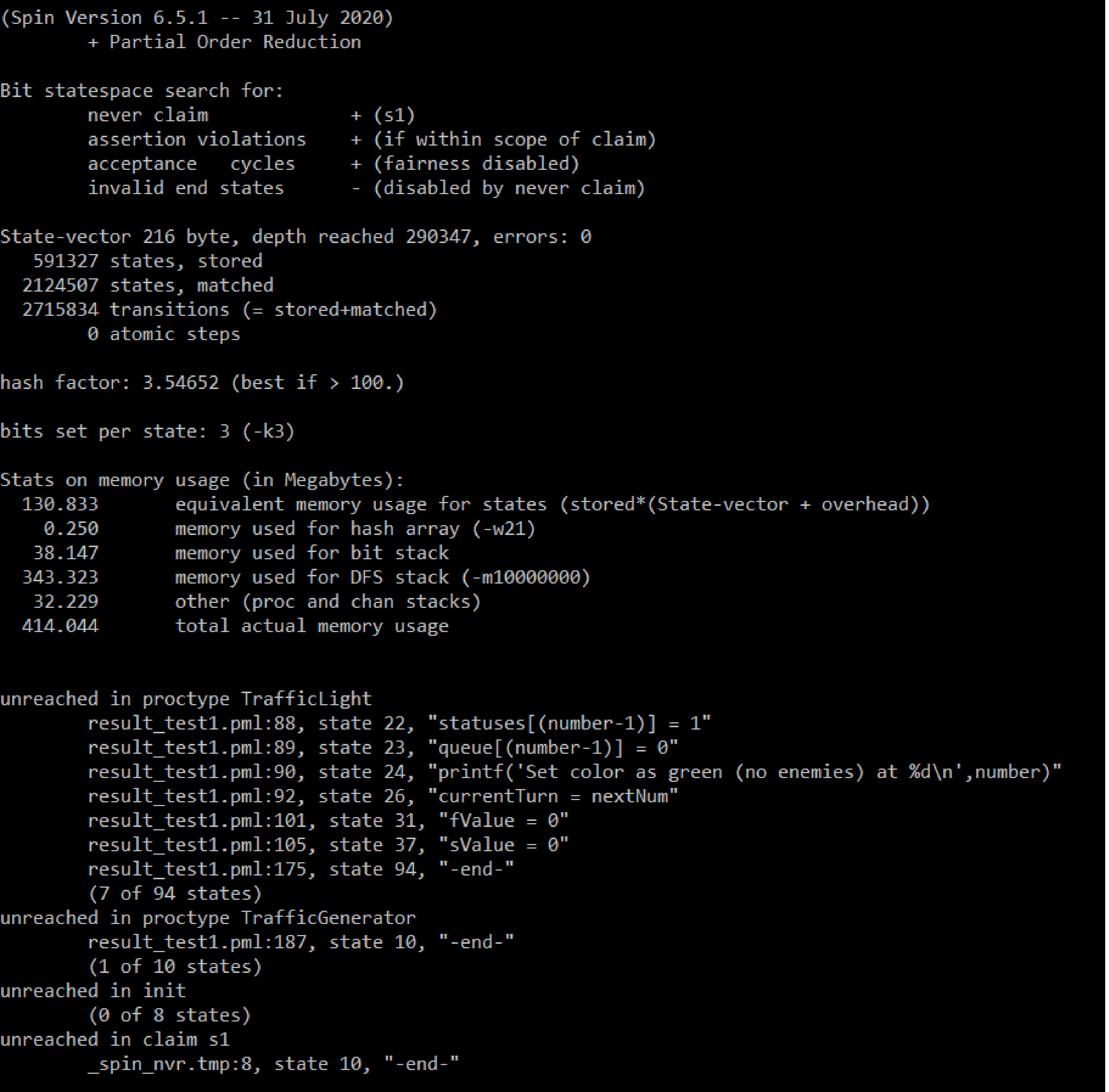
Safety



```
// Безопасность - нет пересечений между:

ltl s1 {
    [] (!(statuses [0] == true && statuses [1] == true)) // зеленый и розовый
};

ltl s2 {
    [] (! (statuses [0] == true && statuses [2] == true)) // зеленый и синий
};

ltl s3 {
   [] (! (statuses [5] == true && statuses [2] == true)) // синий и черный
};

ltl s4 {
   [] (! (statuses [4] == true && statuses [2] == true)) // синий и пешеход
};

ltl s5 {
   [] (! (statuses [1] == true && statuses [3] == true)) // розовый и красный
};

ltl s6 {
   [] (! (statuses [3] == true && statuses [4] == true)) // красный и пешеход
};

ltl s7 {
   [] (! (statuses [3] == true && statuses [5] == true)) // красный и черный
};
```



Validation. Safety



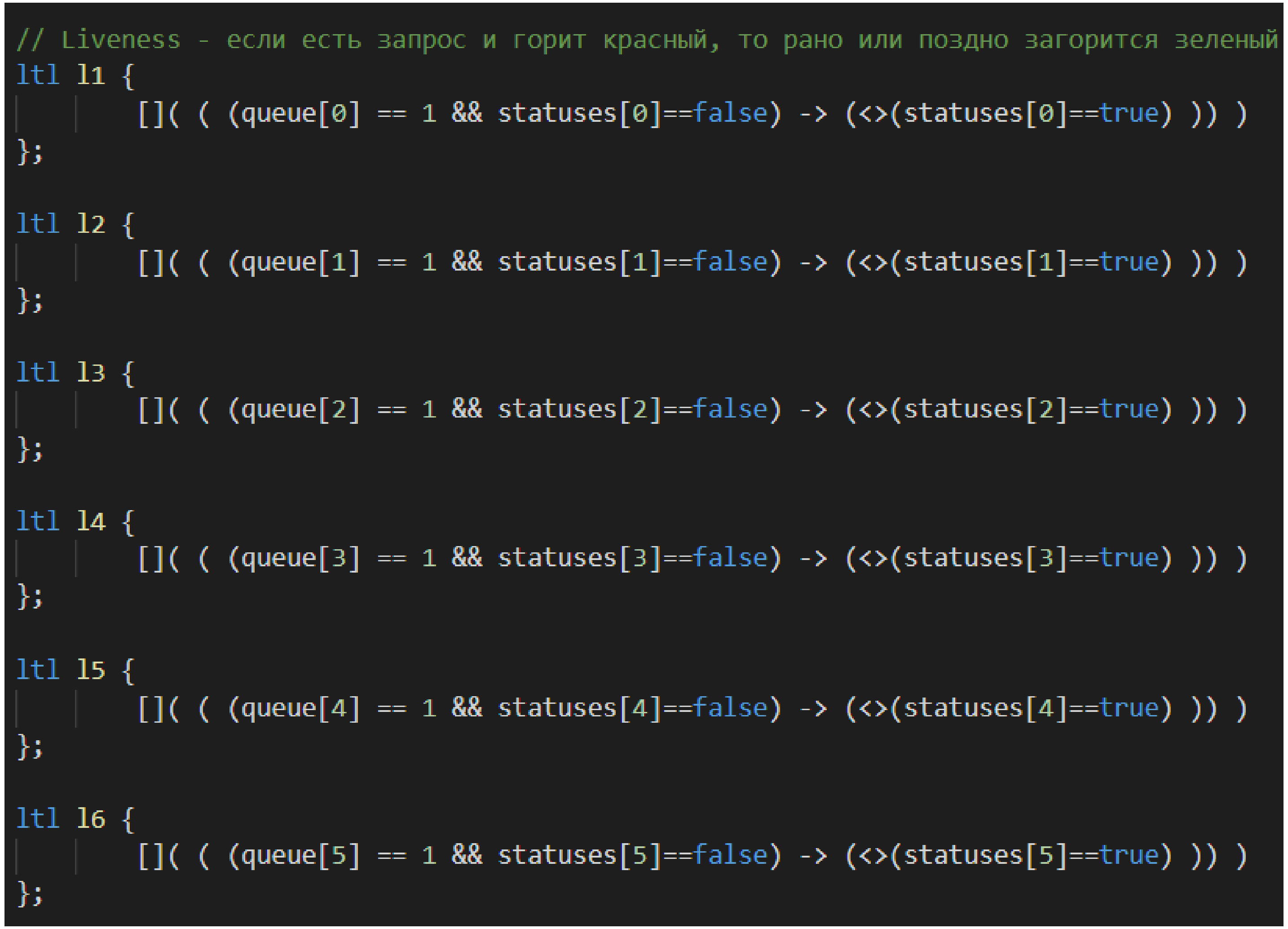
```
(Spin Version 6.5.1 -- 31 July 2020)
        + Partial Order Reduction

Bit statespace search for:
        never claim             + (s1)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 216 byte, depth reached 290347, errors: 0
   591327 states, stored
  2124507 states, matched
  2715834 transitions (= stored+matched)
        0 atomic steps

hash factor: 3.54652 (best if > 100.)

bits set per state: 3 (-k3)

Stats on memory usage (in Megabytes):
  130.833       equivalent memory usage for states (stored*(State-vector + overhead))
    0.250       memory used for hash array (-w21)
   38.147       memory used for bit stack
  343.323       memory used for DFS stack (-m10000000)
   32.229       other (proc and chan stacks)
  414.044       total actual memory usage


unreached in proctype TrafficLight
        result_test1.pml:88, state 22, "statuses[(number-1)] = 1"
        result_test1.pml:89, state 23, "queue[(number-1)] = 0"
        result_test1.pml:90, state 24, "printf('Set color as green (no enemies) at %d\n',number)"
        result_test1.pml:92, state 26, "currentTurn = nextNum"
        result_test1.pml:101, state 31, "fValue = 0"
        result_test1.pml:105, state 37, "sValue = 0"
        result_test1.pml:175, state 94, "-end-"
        (7 of 94 states)
unreached in proctype TrafficGenerator
        result_test1.pml:187, state 10, "-end-"
        (1 of 10 states)
unreached in init
        (0 of 8 states)
unreached in claim s1
        _spin_nvr.tmp:8, state 10, "-end-"
```

```
spin.exe
            -search
            -m10000000
            -DBITSTATE
            -w21
            -a
            -ltl
s1
result_test1.pml
```

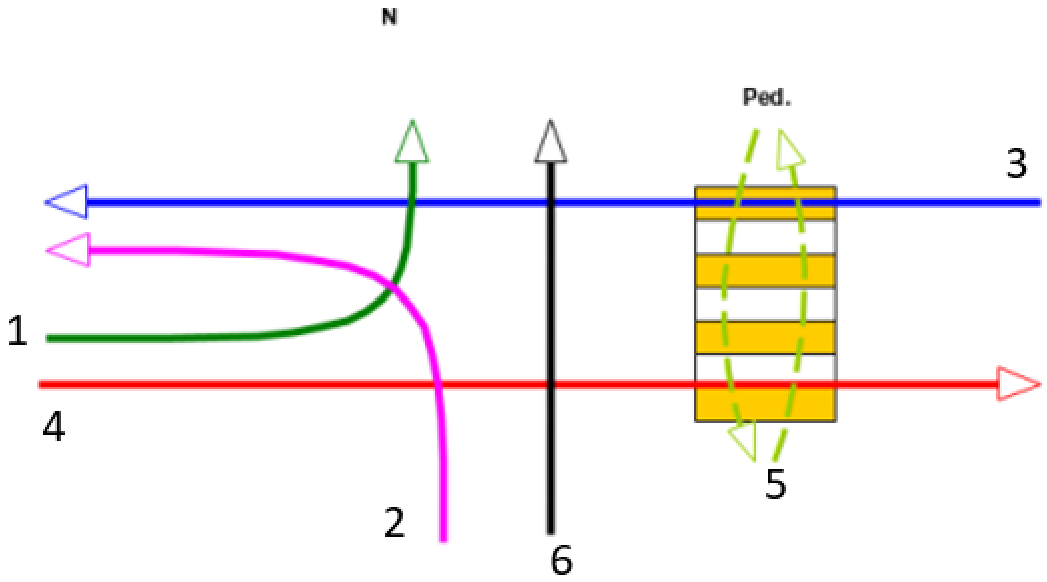
Validation. Liveness

```
[](  (queue[i] == 1 && statuses[i]==false) ->
                                          <>(statuses[i]== true))
```

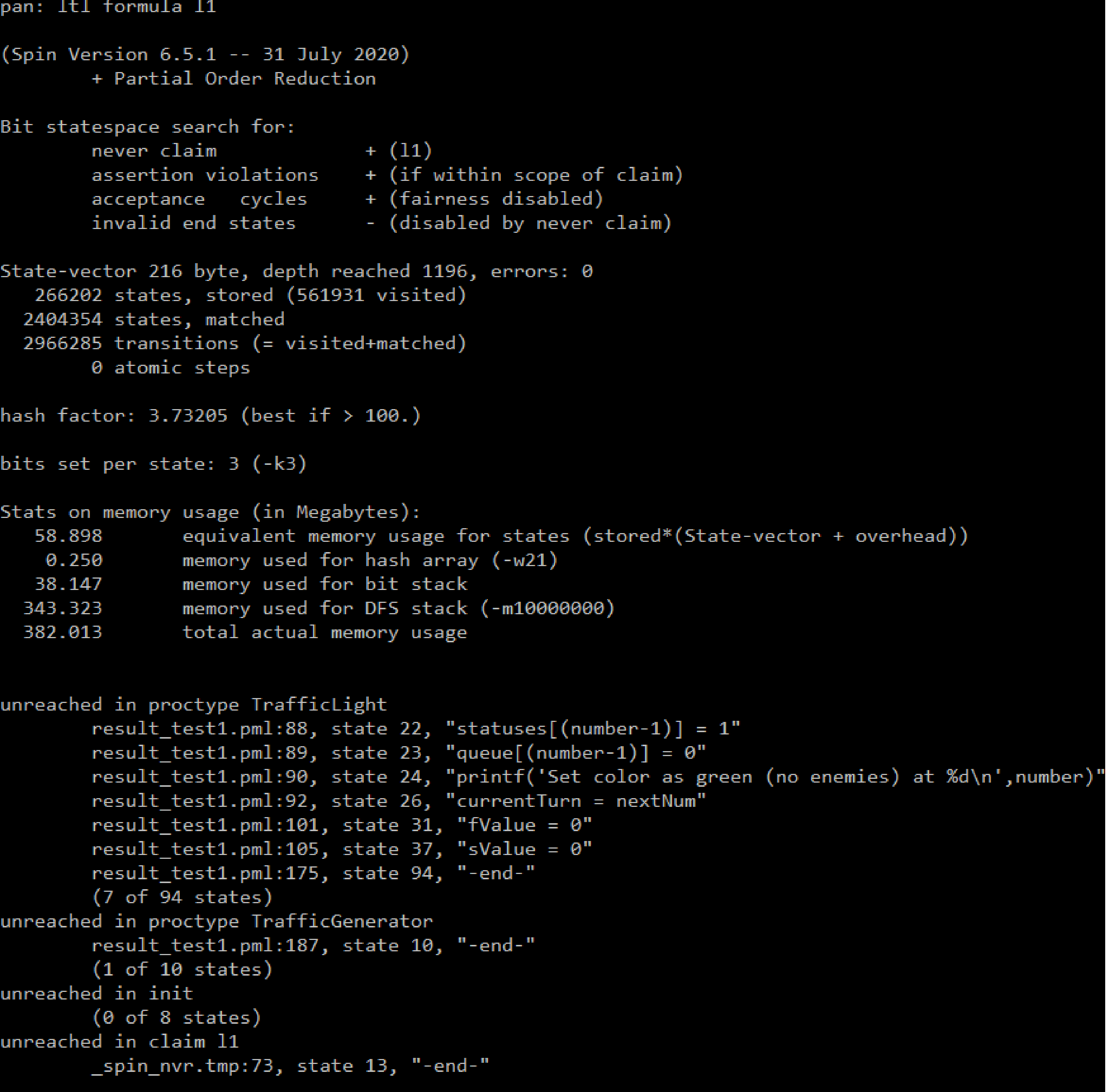
Liveness



```
// Liveness - если есть запрос и горит красный, то рано или поздно загорится зеленый
ltl l1 {
        []( ( (queue[0] == 1 && statuses[0]==false) -> (<>(statuses[0]==true) )) )
};

ltl l2 {
        []( ( (queue[1] == 1 && statuses[1]==false) -> (<>(statuses[1]==true) )) )
};

ltl l3 {
        []( ( (queue[2] == 1 && statuses[2]==false) -> (<>(statuses[2]==true) )) )
};

ltl l4 {
        []( ( (queue[3] == 1 && statuses[3]==false) -> (<>(statuses[3]==true) )) )
};

ltl l5 {
        []( ( (queue[4] == 1 && statuses[4]==false) -> (<>(statuses[4]==true) )) )
};

ltl l6 {
        []( ( (queue[5] == 1 && statuses[5]==false) -> (<>(statuses[5]==true) )) )
};
```



Validation. Liveness



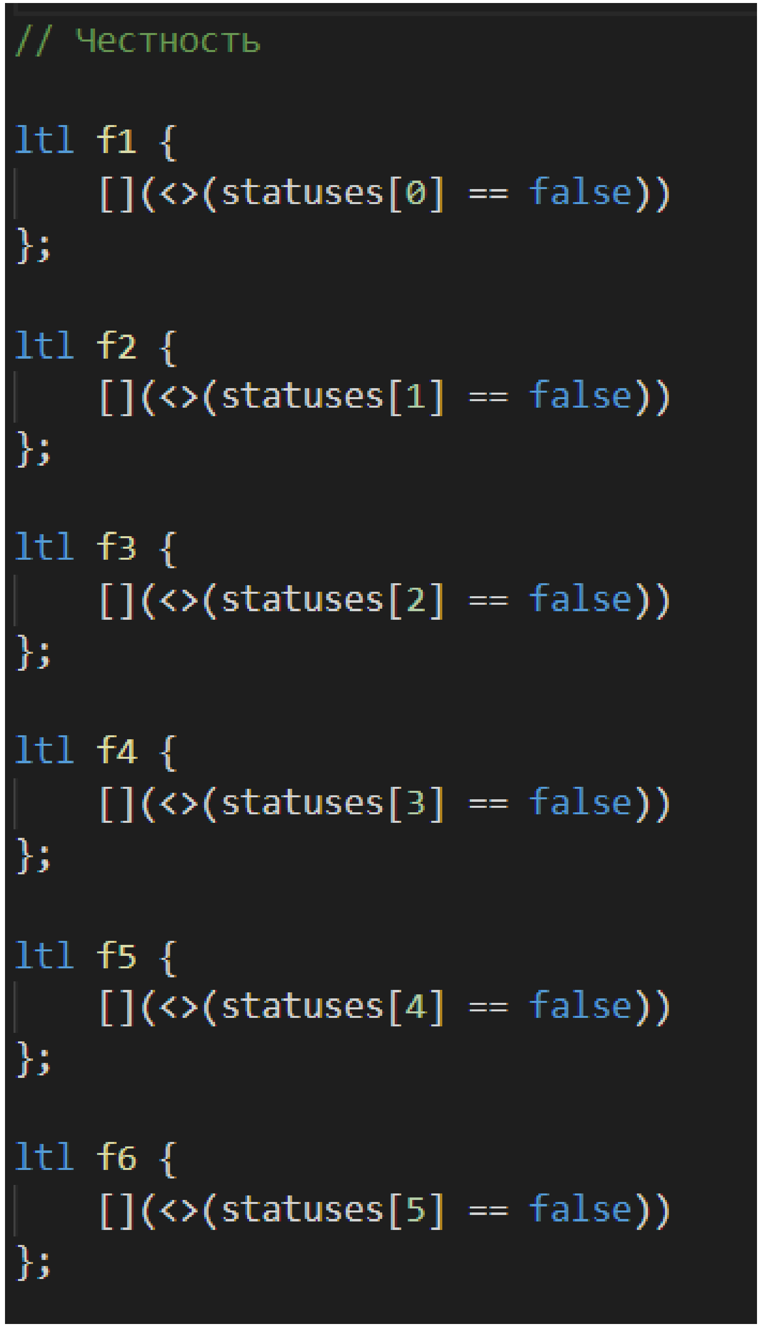
```
pan: ltl formula l1

(Spin Version 6.5.1 -- 31 July 2020)
        + Partial Order Reduction

Bit statespace search for:
        never claim             + (l1)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 216 byte, depth reached 1196, errors: 0
   266202 states, stored (561931 visited)
  2404354 states, matched
  2966285 transitions (= visited+matched)
        0 atomic steps

hash factor: 3.73205 (best if > 100.)

bits set per state: 3 (-k3)

Stats on memory usage (in Megabytes):
   58.898       equivalent memory usage for states (stored*(State-vector + overhead))
    0.250       memory used for hash array (-w21)
   38.147       memory used for bit stack
  343.323       memory used for DFS stack (-m10000000)
  382.013       total actual memory usage


unreached in proctype TrafficLight
        result_test1.pml:88, state 22, "statuses[(number-1)] = 1"
        result_test1.pml:89, state 23, "queue[(number-1)] = 0"
        result_test1.pml:90, state 24, "printf('Set color as green (no enemies) at %d\n',number)"
        result_test1.pml:92, state 26, "currentTurn = nextNum"
        result_test1.pml:101, state 31, "fValue = 0"
        result_test1.pml:105, state 37, "sValue = 0"
        result_test1.pml:175, state 94, "-end-"
        (7 of 94 states)
unreached in proctype TrafficGenerator
        result_test1.pml:187, state 10, "-end-"
        (1 of 10 states)
unreached in init
        (0 of 8 states)
unreached in claim l1
        _spin_nvr.tmp:73, state 13, "-end-"
```

```
spin.exe
            -search
            -m10000000
            -DBITSTATE
            -w21
            -a
            -ltl
l1
result_test1.pml
```

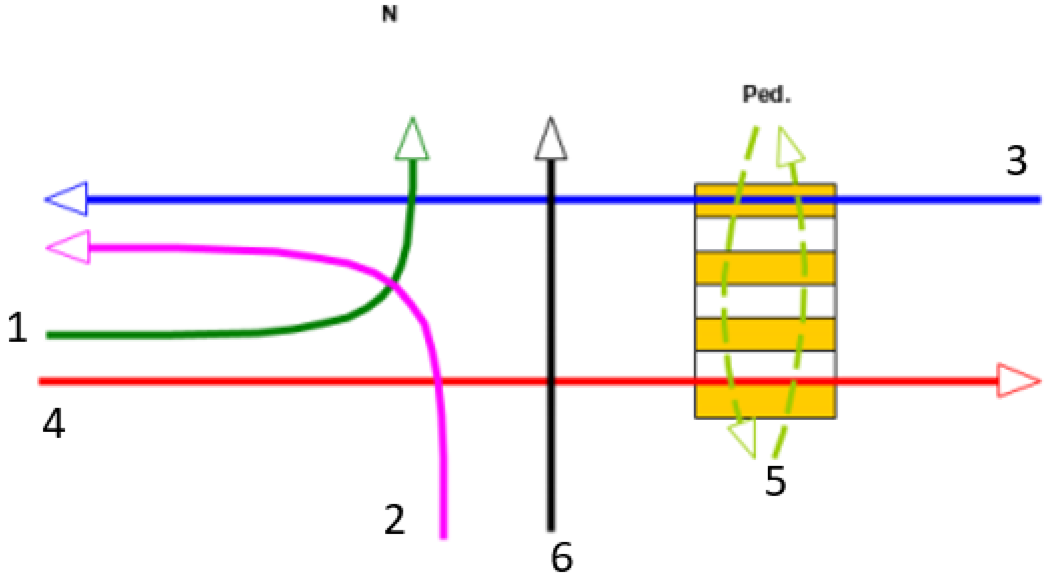
Validation. Fairness

```
[](<>(statuses[i] == false))
```

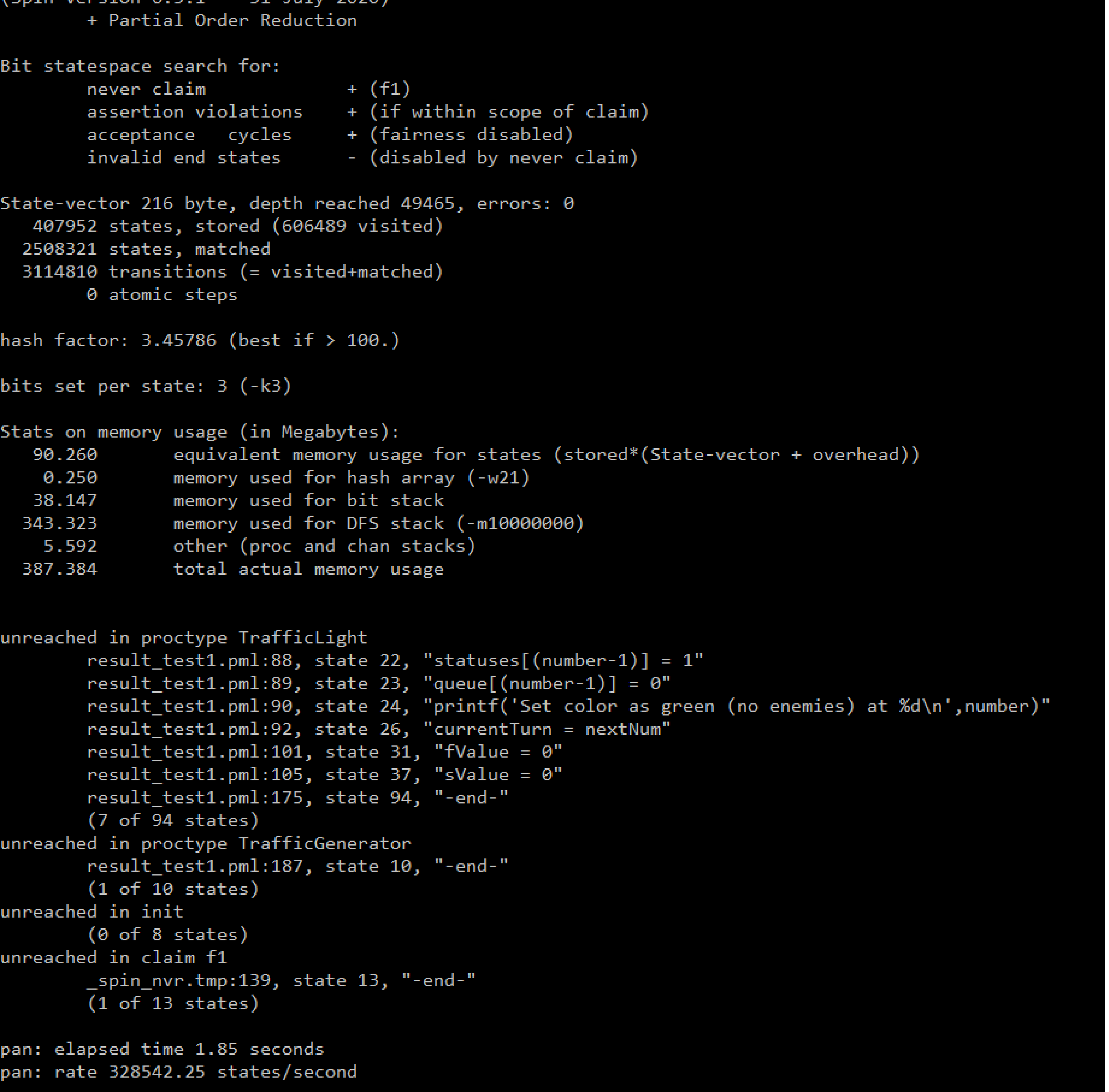
Fairness



```
// Честность

ltl f1 {
    [](<>(statuses[0] == false))
};

ltl f2 {
    [](<>(statuses[1] == false))
};

ltl f3 {
    [](<>(statuses[2] == false))
};

ltl f4 {
    [](<>(statuses[3] == false))
};

ltl f5 {
    [](<>(statuses[4] == false))
};

ltl f6 {
    [](<>(statuses[5] == false))
};
```



Validation. Liveness



```
(Spin Version 6.5.1 -- 31 July 2020)
        + Partial Order Reduction

Bit statespace search for:
        never claim             + (f1)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 216 byte, depth reached 49465, errors: 0
   407952 states, stored (606489 visited)
  2508321 states, matched
  3114810 transitions (= visited+matched)
        0 atomic steps

hash factor: 3.45786 (best if > 100.)

bits set per state: 3 (-k3)

Stats on memory usage (in Megabytes):
   90.260       equivalent memory usage for states (stored*(State-vector + overhead))
    0.250       memory used for hash array (-w21)
   38.147       memory used for bit stack
  343.323       memory used for DFS stack (-m10000000)
    5.592       other (proc and chan stacks)
  387.384       total actual memory usage


unreached in proctype TrafficLight
        result_test1.pml:88, state 22, "statuses[(number-1)] = 1"
        result_test1.pml:89, state 23, "queue[(number-1)] = 0"
        result_test1.pml:90, state 24, "printf('Set color as green (no enemies) at %d\n',number)"
        result_test1.pml:92, state 26, "currentTurn = nextNum"
        result_test1.pml:101, state 31, "fValue = 0"
        result_test1.pml:105, state 37, "sValue = 0"
        result_test1.pml:175, state 94, "-end-"
        (7 of 94 states)
unreached in proctype TrafficGenerator
        result_test1.pml:187, state 10, "-end-"
        (1 of 10 states)
unreached in init
        (0 of 8 states)
unreached in claim f1
        _spin_nvr.tmp:139, state 13, "-end-"
        (1 of 13 states)

pan: elapsed time 1.85 seconds
pan: rate 328542.25 states/second
```

spin.exe

-search

-m10000000

-DBITSTATE

-w21

-a

-ltl

s1

result_test1.pml

Thank you for your attention!

https://github.com/MaximGilman/Promela-traffic-verification/