APL Code Golf Autumn Tournament

Adám Brudzewsky

October 26th, 2017

Background

- What is Code Golf?

Background

- What is Code Golf?
- Tradition – known to have been popular with earlier APL hackers.[Wikipedia]

Background

- What is Code Golf?
- Tradition – known to have been popular with earlier APL hackers.[Wikipedia]
- Dyalog '17 workshop

Do Code Golf!

- Encourages exploration – achieve excellence
- Opens mind – think out-of-the-box!

Types of Code

Types of Code

- Readable

Types of Code

- Readable
- Fast

Types of Code

- Readable
- Fast
- Short

Types of Code

- Readable
- Fast
- Short
- Balanced

Selected Problems

- Conway's Game of Life
- Interval Index
- Complete Teams

Conway's Game of Life

Conway's Game of Life

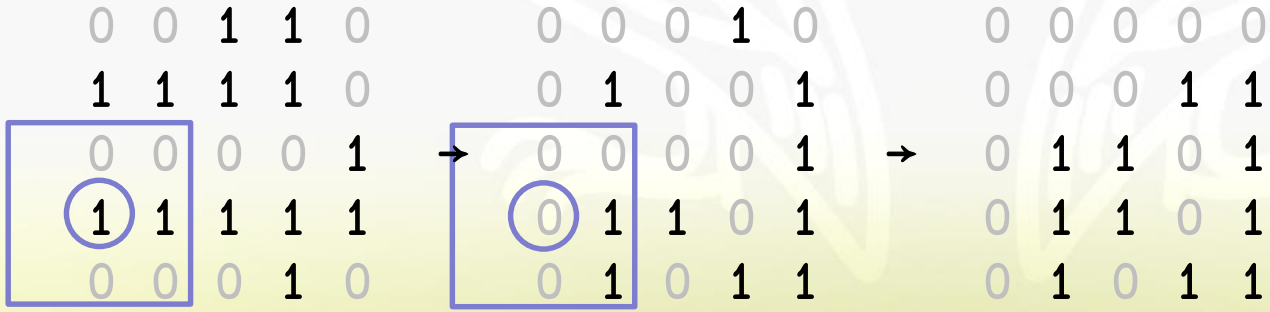
Given a Boolean matrix world, generate the next generation where

- any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- any live cell with two or three live neighbours lives on to the next generation.
- any live cell with more than three live neighbours dies, as if by overpopulation.
- any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- there are hard walls (equivalent to always dead cells) surrounding the world.

Conway's Game of Life

Given a Boolean matrix world, generate the next generation where

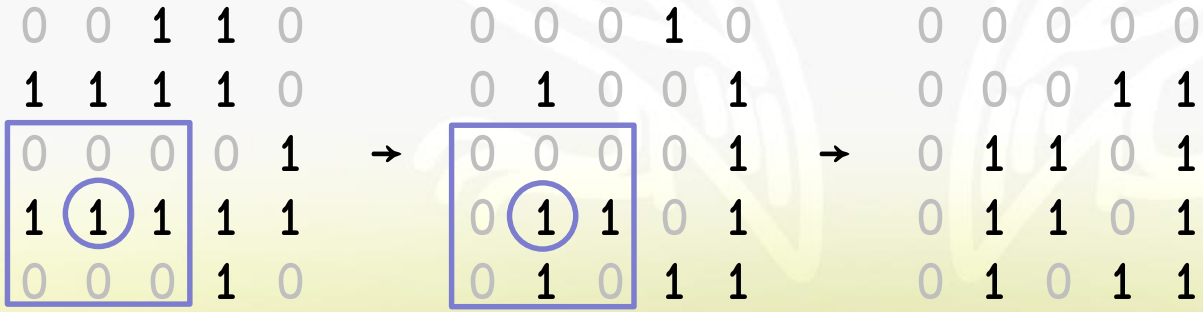
- any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- any live cell with two or three live neighbours lives on to the next generation.
- any live cell with more than three live neighbours dies, as if by overpopulation.
- any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- there are hard walls (equivalent to always dead cells) surrounding the world.



Conway's Game of Life

Given a Boolean matrix world, generate the next generation where

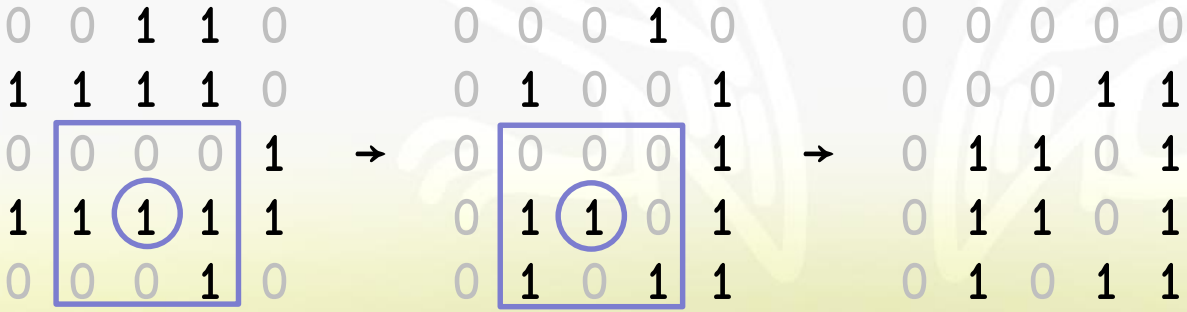
- any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- any live cell with two or three live neighbours lives on to the next generation.
- any live cell with more than three live neighbours dies, as if by overpopulation.
- any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- there are hard walls (equivalent to always dead cells) surrounding the world.



Conway's Game of Life

Given a Boolean matrix world, generate the next generation where

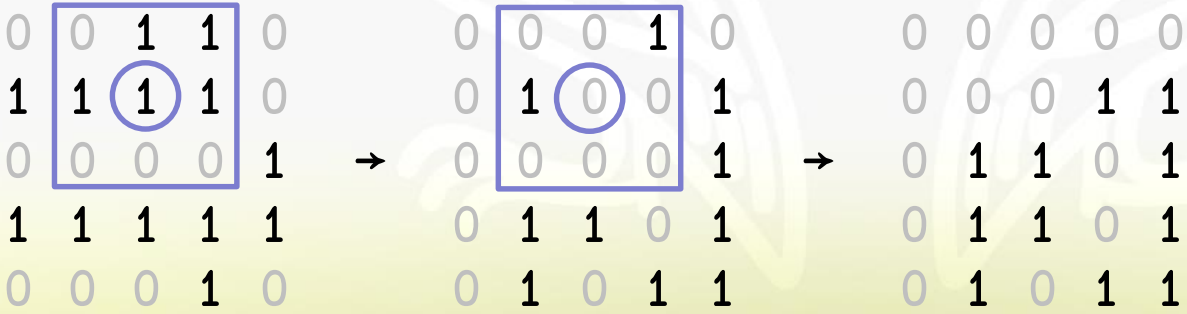
- any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- any live cell with two or three live neighbours lives on to the next generation.
- any live cell with more than three live neighbours dies, as if by overpopulation.
- any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- there are hard walls (equivalent to always dead cells) surrounding the world.



Conway's Game of Life

Given a Boolean matrix world, generate the next generation where

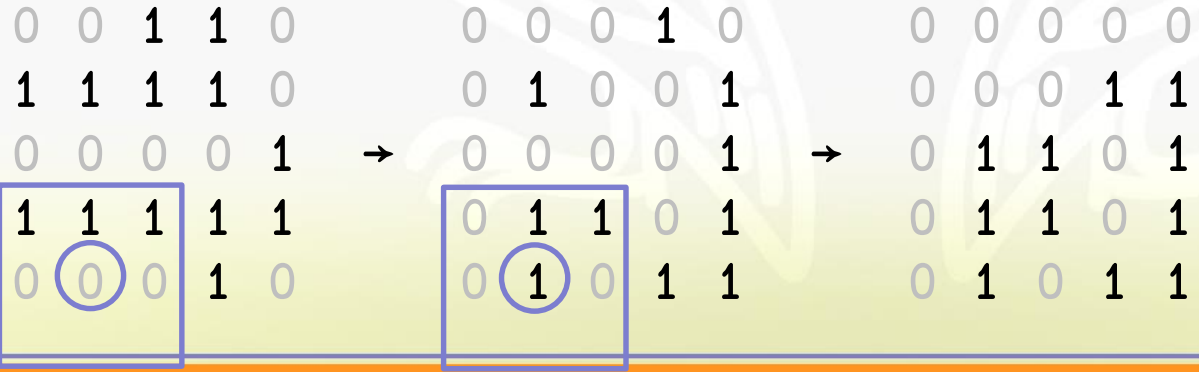
- any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- any live cell with two or three live neighbours lives on to the next generation.
- any live cell with more than three live neighbours dies, as if by overpopulation.
- any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- there are hard walls (equivalent to always dead cells) surrounding the world.



Conway's Game of Life

Given a Boolean matrix world, generate the next generation where

- any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- any live cell with two or three live neighbours lives on to the next generation.
- any live cell with more than three live neighbours dies, as if by overpopulation.
- any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- there are hard walls (equivalent to always dead cells) surrounding the world.



Conway's Game of Life

Given a Boolean matrix world, generate the next generation where

- any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- any live cell with two or three live neighbours lives on to the next generation.
- any live cell with more than three live neighbours dies, as if by overpopulation.
- any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- there are hard walls (equivalent to always dead cells) surrounding the world.

Conway's Game of Life

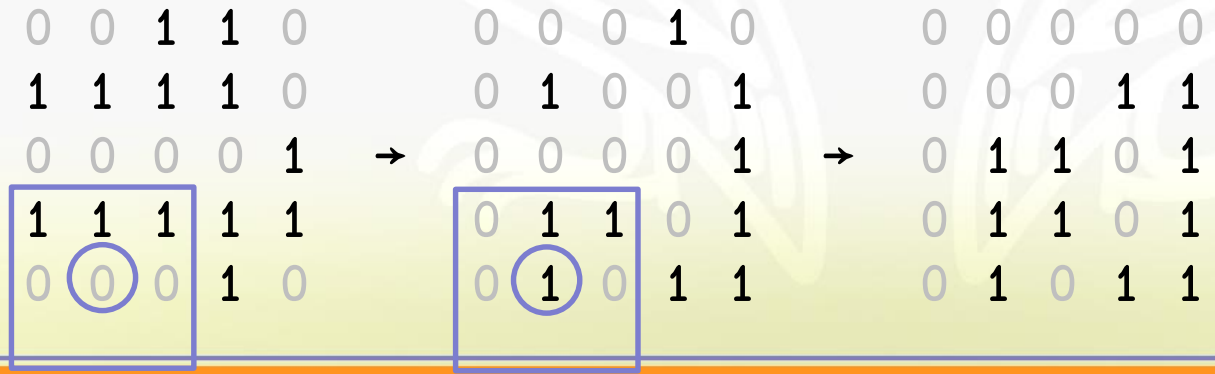
Given a Boolean matrix world, generate the next generation where a cell is alive if

- it was dead but had three alive neighbours (3 "including itself")
- it was alive and had two alive neighbours (3 including itself)
- it was alive and had three alive neighbours (4 including itself)

Conway's Game of Life

Given a Boolean matrix world, generate the next generation where a cell is alive if

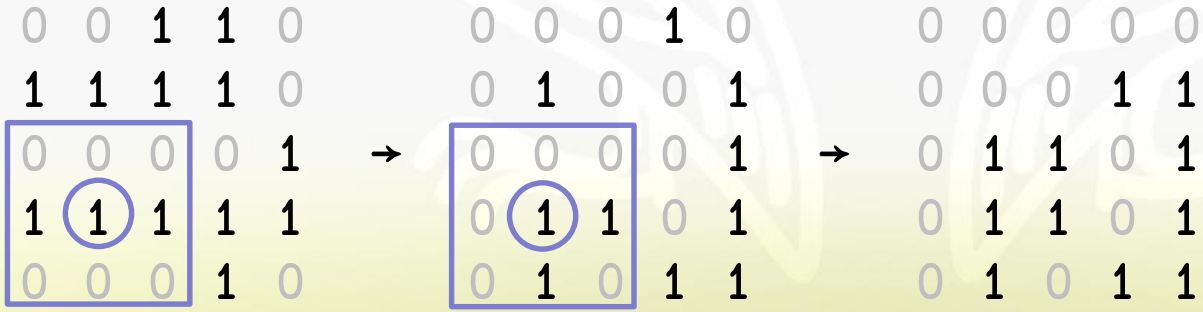
- it was dead but had three alive neighbours (3 "including itself")
- it was alive and had two alive neighbours (3 including itself)
- it was alive and had three alive neighbours (4 including itself)



Conway's Game of Life

Given a Boolean matrix world, generate the next generation where a cell is alive if

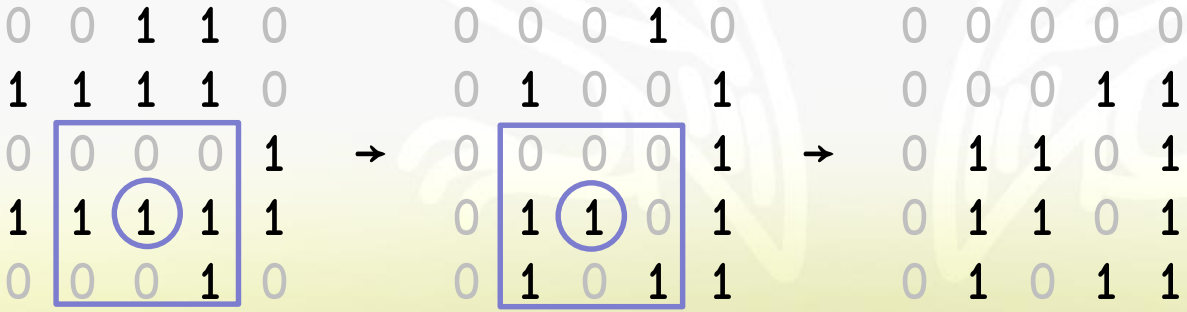
- it was dead but had three alive neighbours (3 "including itself")
- it was alive and had two alive neighbours (3 including itself)
- it was alive and had three alive neighbours (4 including itself)



Conway's Game of Life

Given a Boolean matrix world, generate the next generation where a cell is alive if

- it was dead but had three alive neighbours (3 "including itself")
- it was alive and had two alive neighbours (3 including itself)
- it was alive and had three alive neighbours (4 including itself)



Conway's Game of Life

- Traditional solution

```
{↑1 ⍵∨.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂⍵}
```

See https://youtu.be/a9xAKttWgP4

Conway's Game of Life

- Traditional solution

```
{↑1 ⍵∨.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂⍵}
```

See https://youtu.be/a9xAKttWgP4

- Golfed using version 16.0 features

```
{≠⍳⍵}⌹3 3∊¨3+0,¨⊢
```

Conway's Game of Life

- Traditional solution*

criteria neighbourhood

```
{↑1 ⍵∨.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂⍵}
```

See https://youtu.be/a9xAKttWgP4

- Golfed using version 16.0 features**

neighbourhood criteria

```
{≠⍳⍵}⌺3 3∊¨3+0,¨⊢
```

* Life on a torus ** Life on a plane

Conway's Game of Life

```
{≠⍳⍵}⌺3 3∊¨3+0,¨⊢
```

Conway's Game of Life

```
({≠⍨⍵}⌺3 3) ⍷⍨ (3+0,⍨⊢)
```

Conway's Game of Life

```
({≠⍵⍷⍨3 3}⍣⍵)
```

Conway's Game of Life

```
({≢⍸⍵}⌺3 3)
 ({⊂⍵}⌺3 3) W
```

Conway's Game of Life

```
(⊂{≠⌿1⍵}⌺3 3)
 (⊂{⊂⍵}⌺3 3) W
```

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
(⊂{≠⌿⍵}⌺3 3)
 (⊂{⊂⍵}⌺3 3) W
```

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	1	0	0
0	0	0	0	0	1	0	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

W

0	0	1	1	0
1	1	1	1	0
0	0	0	0	1
1	1	1	1	1
0	0	0	1	0

Conway's Game of Life

```
      ({≠⌿⍵}⌺3 3)
        ({⊂⍵}⌺3 3) W
```

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	1	0	0
0	0	0	0	0	1	0	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

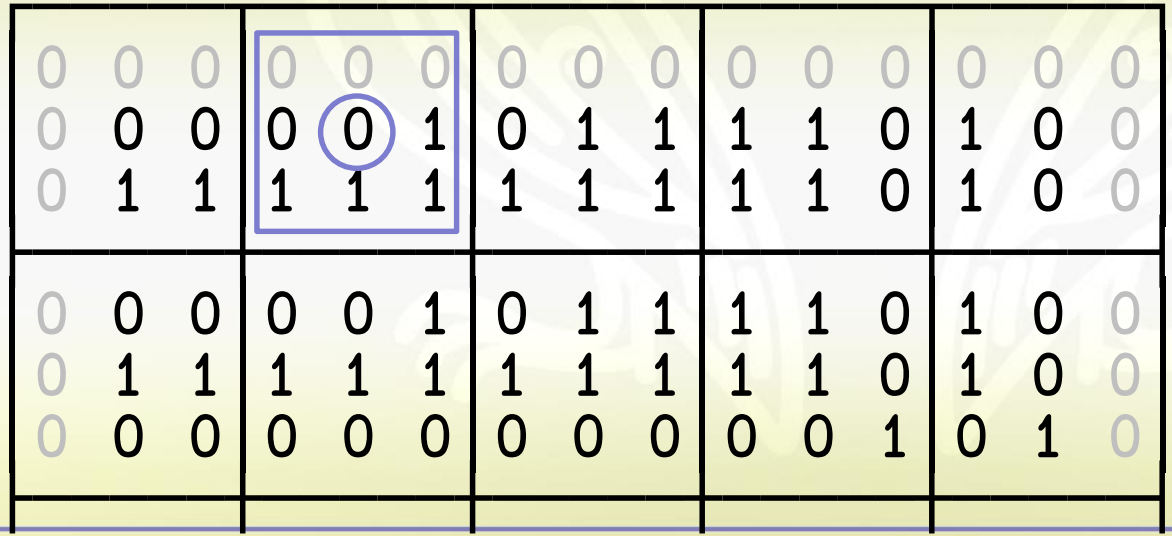
```
({≠⊂⍵}⌺3 3)
 ({⊂⍵}⌺3 3) W
```

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	1	0	0
0	0	0	0	0	1	0	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
({≠⌶ω}⌺3 3)
 ({⊂ω}⌺3 3) W
```



```
  W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
      ({≠⌿⍵}⌺3 3)
        ({⊂⍵}⌺3 3) W
```

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	1	0	0
0	0	0	0	0	1	0	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
(({≠⌶⍵}⌺3 3)
 ({⊂⍵}⌺3 3) W
```

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
      ({≠⌿⍵}⌺3 3)
    ⊃ ({⊂⍵}⌺3 3) W
```

```
0 0 0
0 0 0
0 1 1
```

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
      ({≠⍳⍨ω}⌺3 3)
    ⊃ ({⊂ω}⌺3 3) W
```

```
0 0 0
0 0 0
0 1 1
```

```
    ⍳⍨  ⊃ ({⊂ω}⌺3 3) W
```

3 2	3 3

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
      ({≠⍸⍵}⌺3 3)
    ⊃ ({⊂⍵}⌺3 3) W
```

```
0 0 0
0 0 0
0 1 1
```

```
    ⍸    ⊃ ({⊂⍵}⌺3 3) W
```

3 2	3 3

```
   ≠⍸    ⊃ ({⊂⍵}⌺3 3) W
2
```

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
      ({≠⍳⍨ω}⌺3 3)
    ⊃ ({⊂ω}⌺3 3) W
```

```
0 0 0
0 0 0
0 1 1
```

```
    ⍳⍨  ⊃ ({⊂ω}⌺3 3) W
```

3 2	3 3

```
  +/,   ⊃ ({⊂ω}⌺3 3) W
2
```

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
({≠⍸⍵}⌺3 3) ∊¨ (3+0,¨⊢)
```

Conway's Game of Life

```
(3+0,¨⊢)
```

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
(3+0,¨⊢)
(3 + (0,¨⊢))
```

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
  (3+0,¨⊢)
(3 + (0,¨⊢))
     (0 ,¨ ⊢) W
```

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
  (3+0,¨⊢)
(3 + (0,¨⊢))
     (0 ,¨ ⊢) W
```

0	0	0	0	0	1	0	1	0	0
0	1	0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0	0	1

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
  (3+0,¨⊢)
(3 + (0,¨⊢))
(3 + 0 ,¨ ⊢) W
```

3	3	3	3	3	4	3	4	3	3
3	4	3	4	3	4	3	4	3	3
3	3	3	3	3	3	3	3	3	4

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
({≠⌶⍵}⌺3 3) ⍷⍨ (3+0,⍨⊢)
```

2	4	5	4	2
2	4	5	5	3
4	6	6	6	4
2	3	4	5	4
2	3	4	4	3

3	3	3	3	3	4	3	4	3	3
3	4	3	4	3	4	3	4	3	3
3	3	3	3	3	3	3	3	3	4
3	4	3	4	3	4	3	4	3	4
3	3	3	3	3	3	3	4	3	3

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
({≠⍳⍵}⌺3 3 3∊¨3+0,¨⊢) W
```

0	0	0	1	0
0	1	0	0	1
0	0	0	0	1
0	1	1	0	1
0	1	0	1	1

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Conway's Game of Life

```
({≠⊂⍵}⌺3 3∊¨3+0,¨⊢) W
```

0	0	0	1	0
0	1	0	0	1
0	0	0	0	1
0	1	1	0	1
0	1	0	1	1

• ONLY 17 CHARS •

```
      W
0 0 1 1 0
1 1 1 1 0
0 0 0 0 1
1 1 1 1 1
0 0 0 1 0
```

Interval Index

using pre-16.0 primitives

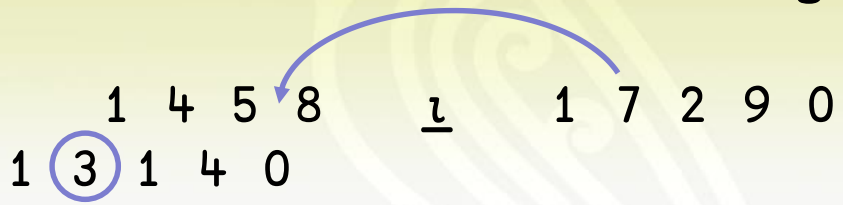
Interval Index

Implement 16.0's α⍸ω for vectors, using pre-16.0 Dyalog APL.
For each element of ω, find which "gap" of α it belongs in.

```
    1 4 5 8   ⍸   1 7 2 9 0
1 3 1 4 0
```

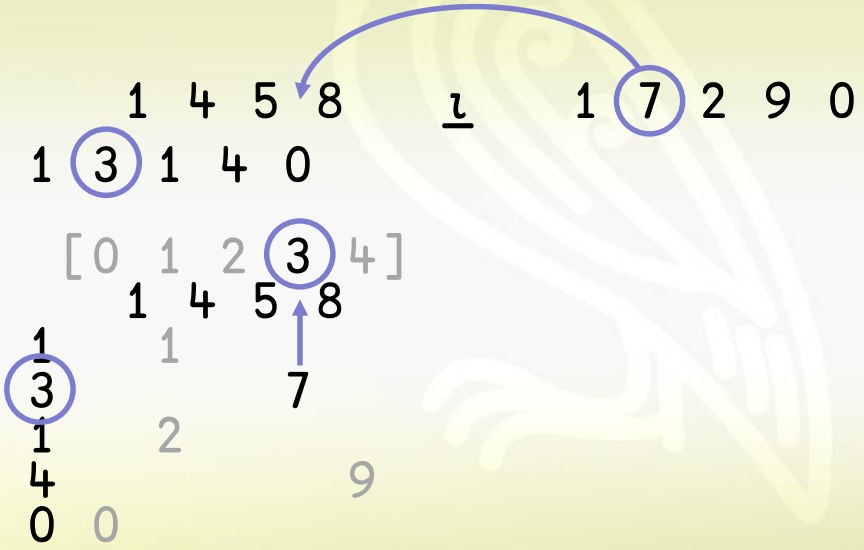
Interval Index

Implement 16.0's α⍸ω for vectors, using pre-16.0 Dyalog APL.
For each element of ω, find which "gap" of α it belongs in.



Interval Index

Implement 16.0's α⍸ω for vectors, using pre-16.0 Dyalog APL.
For each element of ω, find which "gap" of α it belongs in.



Interval Index

Implement 16.0's α⍸ω for vectors, using pre-16.0 Dyalog APL.
For each element of ω, find which "gap" of α it belongs in.

```
     1  4  5  8    ⍸    1  7  2  9  0
1  3  1  4  0

  [0  1  2  3  4]
     1  4  5  8
1     1
3           7
1     2
4              9
0  0
```

Interval Index

- Traditional solution

```
{
    a←⍋⍺,⍵                ⍝ indices which will sort (⍺,⍵) ascending
    i←(⍳≠⍺),(≠⍵)⍴0        ⍝ indices of ⍺, and one zero per item of ⍵
    g←⌈\i[a]              ⍝ the index of ⍺ which the sorted (⍺,⍵) follows (= gap number)
    g[a⍳(≠⍺)+⍳≠⍵]         ⍝ gap numbers where items of ⍵ are found in the sorted array
}
```

Interval Index

- Traditional solution

```
{
    a←⍋⍺,⍵                  ⍝ indices which will sort (⍺,⍵) ascending
    i←(⍳≢⍺),(≢⍵)⍴0          ⍝ indices of ⍺, and one zero per item of ⍵
    g←⌈\i[a]                ⍝ the index of ⍺ which the sorted (⍺,⍵) follows (= gap number)
    g[a⍳(≢⍺)+⍳≢⍵]           ⍝ gap numbers where items of ⍵ are found in the sorted array
}
```

- Golfed

```
+⌿∘.(</∘⍋,)
```

Interval Index

- Traditional solution

```
{
    a←⍋⍺,⍵                  ⍝ indices which will sort (⍺,⍵) ascending
    i←(⍳≠⍺),(≠⍵)⍴0          ⍝ indices of ⍺, and one zero per item of ⍵
    g←⌈\i[a]                ⍝ the index of ⍺ which the sorted (⍺,⍵) follows (= gap number)
    g[a⍳(≠⍺)+⍳≠⍵]           ⍝ gap numbers where items of ⍵ are found in the sorted array
}
```

- Golfed

```
+⌿∘.(</∘⍋,)
```

- In version 16.0

```
⍸
```

Interval Index

```
4 7 (+⌿∘.<) 5 2 8
```

Interval Index

```
4 7  (+⌿∘.<)  5 2 8
```

```
4 7      ∘.<  5 2 8
```

Interval Index

```
      4 7 (+⌿∘.<) 5 2 8

      4 7   ∘.<   5 2 8
< 5 2 8
4 1 0 1
7 0 0 1
```

Interval Index

```
      4 7 (+⌿∘.<) 5 2 8
0 1 2
      4 7   ∘.<   5 2 8
< 5 2 8
4 1 0 1
7 0 0 1
+⌿
  1 0 2
```

Interval Index

```
      4 7 (+⌿∘.<) 5 2 8
1 0 2

 [0 1 2]
   4 7
1    5
0 2
2      8
```

Interval Index

```
'D' 'G' (+⌿∘.<) 'E' 'B' 'H'
```

Interval Index

```
      'D' 'G' (+⍀∘.<) 'E' 'B' 'H'
DOMAIN ERROR
```

Interval Index

```
      'D' 'G' (+⌿∘.<) 'E' 'B' 'H'
DOMAIN ERROR
      'D' 'G'    ∘.<  'E' 'B' 'H'
DOMAIN ERROR
```

Interval Index

```
      'D' 'G' (+⍀∘.<) 'E' 'B' 'H'
DOMAIN ERROR
      'D'       ∘.<  'E'
```

Interval Index

```
      'D' 'G' (+≠∘.<) 'E' 'B' 'H'
DOMAIN ERROR
      'D'           <   'E'
```

Interval Index

```
      'D' 'G' (+≠∘.<) 'E' 'B' 'H'
DOMAIN ERROR
      'D'           <  'E'
DOMAIN ERROR
```

Interval Index

```
      'D' 'G' (+⌿∘.<) 'E' 'B' 'H'
DOMAIN ERROR
      ⍋ 'D'           ,  'E'
```

Interval Index

```
      'D' 'G' (+⍤⍥.<) 'E' 'B' 'H'
DOMAIN ERROR
      ⍋ 'D'           ,  'E'
1 2
```

Interval Index

```
      'D' 'G' (+⌿∘.<) 'E' 'B' 'H'
DOMAIN ERROR
      ⍋ 'D'           ,  'E'
1 2
   </ ⍋ 'D'           ,  'E'
1
```

Interval Index

```
      'D' 'G' (+⌿∘.<) 'E' 'B' 'H'
DOMAIN ERROR
      ⍋ 'D'           ,  'E'
1 2
  </ ⍋ 'D'           ,  'E'
1
      'D'      (</∘⍋,) 'E'
1
```

Interval Index

Interval Index

```
'D' 'G'  ∘.(</∘⍋,)  'E' 'B' 'H'
```

Interval Index

```
      'D' 'G'    ∘.(</∘⍋,) 'E' 'B' 'H'
</∘⍋, E B H
    D 1 0 1
    G 0 0 1
```

Interval Index

```
      'D' 'G' (+⌿∘.(<⌿∘⍋,)) 'E' 'B' 'H'

      'D' 'G'     ∘.(<⌿∘⍋,)  'E' 'B' 'H'
<⌿∘⍋, E B H
   D 1 0 1
   G 0 0 1
```

Interval Index

```
      'D' 'G' (+⌿∘.(</∘⍋,)) 'E' 'B' 'H'
  [0   1   2]
    'D' 'G'     ∘.(</∘⍋,)  'E' 'B' 'H'
</∘⍋, E B H
  D 1 0 1
  G 0 0 1
 +⌿
    1 0 2
```

Interval Index

```
      'DG' (+⌿∘.(<⌿∘⍋,)) 'EBH'
1 0 2

  [0 1 2]
   D G
1    E
0 B
2     H
```

Interval Index

```
      'DG' ⍳ 'EBH'
1 0 2

  [0 1 2]
    D G
1     E
0 B
2     H
```

Complete Teams

Complete Teams

Get the numbers that occur exactly twice.

```
      7 1 8 2 8 1 8 2
1 2

      1 4 1 5 9 9 6 5
1 5 9
```

Complete Teams using pre-16.0 features only

- Naïve traditional solution (“Readable”)

```
{∪(2=+/ω∘.=ω)/ω}  ⍝ compare all, then select those with two matches
```

Complete Teams using pre-16.0 features only

- Naïve traditional solution ("Readable")

```
{∪(2=+/ω∘.=ω)/ω}  ⍝ compare all, then select those with two matches
```

- Efficient traditional solution ("Fast")

```
{
   s←ω[⍋ω]                ⍝ sort
   m←1,2≠/s               ⍝ mark partitions
   i←{ω/⍳⍴ω}m,1           ⍝ where partitions begin
   t←(2=¯2-/i,1)/i        ⍝ ... and are length-two
   s[t]                   ⍝ get those
}
```

Complete Teams including using 16.0 features

- Golfed (“Short”)

  ∈{2=≢⍵}⌸⊆∪

Complete Teams including using 16.0 features

- Golfed (“Short”)

  ```
  ∈{2=≢⍵}⌸⊆∪
  ```

- Production efficiency (“Balanced”)

  ```
  {(2={≢⍵}⌸⍵)/∪⍵}
  ```

Complete Teams

```
∈{2=≢⍵}⌸⊆∪
```

Complete Teams

```
∈  ({2=≢⍵}⌸⊆∪)
```

Complete Teams

```
(⊂{2=≢⍵}⌸⊆∪)
```

Complete Teams

```
((⊂{2=≢⍵}⌸)  ⊆  (∪))
```

Complete Teams

```
      N
7 1 8 2 8 1 8 2
```

```
(⊢{2=≢⍵}⌸) N
```

Complete Teams

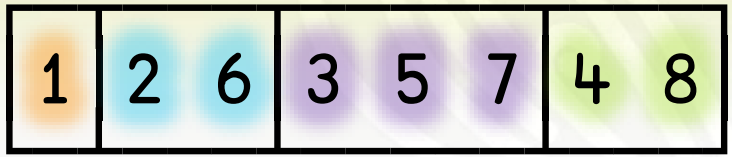
```
(⍸{2=≢⍵}⌸) N
(⍸{⊂⍵}⌸) N
```

```
      N
7 1 8 2 8 1 8 2
```

Complete Teams

```
      N
7 1 8 2 8 1 8 2
1 2 3 4 5 6 7 8
```

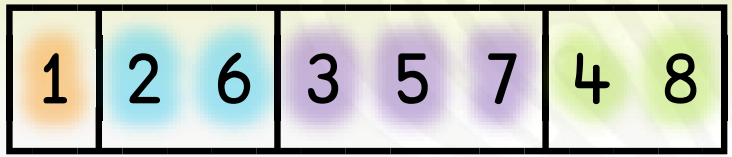
```
(⍸{2=≢⍵}⌸) N
(⍸{⊂⍵}⌸) N
```



Complete Teams

```
      N
7 1 8 2 8 1 8 2
[1 2 3 4 5 6 7 8]
```

```
      (⍸{2=≢⍵}⌸) N
      (⍸{⊂⍵}⌸) N
```

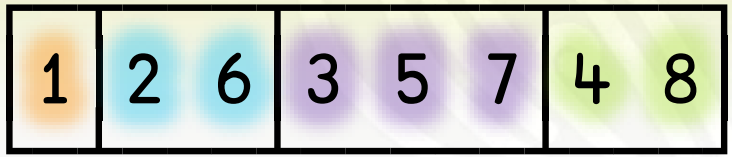



```
      (⍸{≢⍵}⌸) N
1 2 3 2
```

Complete Teams

```
      N
7 1 8 2 8 1 8 2
[1 2 3 4 5 6 7 8]
```

```
(⍸{2=≢⍵}⌸) N
(⍸{⊂⍵}⌸) N
```




```
(⍸{≢⍵}⌸) N
1 2 3 2
(⍸{2=≢⍵}⌸) N
0 1 0 1
```

Complete Teams

```
      N
7 1 8 2 8 1 8 2
```

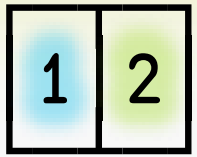
```
((⊂{2=≢⍵}⌸)  ⊆  (∪))
```

Complete Teams

```
      N
7 1 8 2 8 1 8 2
(∪)   N
7 1 8 2
```

Complete Teams

```
      N
7 1 8 2 8 1 8 2
((⊂{2=≠⍵}⌸)⍷∪)N
0 1 0 1 ⊆ 7 1 8 2
```



Complete Teams

```
                                          N
                                  7 1 8 2 8 1 8 2

((({2=≢⍵}⌸)   ⊆   (∪))  N

∊     0 1 0 1     ⊆  7 1 8 2

1 2
```

Complete Teams

N

7 1 8 2 8 1 8 2

∈ ((⊂{2=≢⍵}⌸) ⊆ (∪)) N

1 2

Complete Teams

```
      N
7 1 8 2 8 1 8 2
```

```
(∊{2=≠⍵}⌸⊆∪) N
```

```
1 2
```

Complete Teams performance comparison

```
      N←?1000⍴1000

      ]Runtime -compare "Readable N" "Fast N" "Short N" "Balanced N"

  Readable N → 3.0E¯3 |    0% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
* Fast N     → 2.7E¯5 | -100%
  Short N    → 6.3E¯4 |  -80% ⎕⎕⎕⎕⎕⎕⎕⎕
  Balanced N → 7.2E¯5 |  -98% ⎕
```

Complete Teams performance comparison

```
      N←?10000⍴10000

      ]Runtime -compare "Readable N" "Fast N" "Short N" "Balanced N"

   Readable N → 6.4E¯3 |     0% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
 * Fast N     → 6.3E¯5 | -100%
   Short N    → 2.7E¯3 |   -58% □□□□□□□□□□□□□□□□□
   Balanced N → 1.0E¯4 |   -99% □
```

Thanks for Playing!

1.	Nikolay Georgiev Nikolov	143
2.	Stefano Lanzavecchia	145
3.	Francesco Garue	145
4.	Michele Bellon	157
5.	Zachary Taylor	174
6.	Pierre	208
7.	Veli-Matti Jantunen	215
8.	Gilgamesh Athoraya	218
9.	Giacomo Manfredi	220
10.	Moris Zucca	222

Want more?
codegolf.stackexchange.com

New Features in Dyalog APL

- 14.0
  - `(+⌿÷1⌈≢)` trains
  - `⌸` Key
  - `⍤` Rank
- 15.0
  - `⎕MKDIR ⎕NDELETE ⎕NEXISTS ⎕NGET ⎕NINFO ⎕NPARTS ⎕NPUT`
- 16.0
  - `⍸ω` Where `α⍸ω` Interval Index
  - `⊆ω` Nest `α⊆ω` Partition
  - `@` At
  - `⌺` Stencil
  - `⎕CSV ⎕JSON`

Next Webinar

Source Code Management with GitHub and APL

December 7th, 2017