

ACCU London Online – May 2021

local – global – everywhere – nowhere

A Programming Language for Thinking about Algorithms

Richard Park



DYALOG

APL

Richard Park

What are Algorithms?

A set of instructions to accomplish some task

Something like a recipe

What is Algorithms?

```
big_nuggets = 0
for nugget in bag_of_nuggets
    if nugget.weight > 50
        big_nuggets += 1
    endif
endfor
```

Abstraction

Pro

Get rid of "unnecessary" details

Con

Black boxes are opaque

Spot patterns. Subordinate detail.

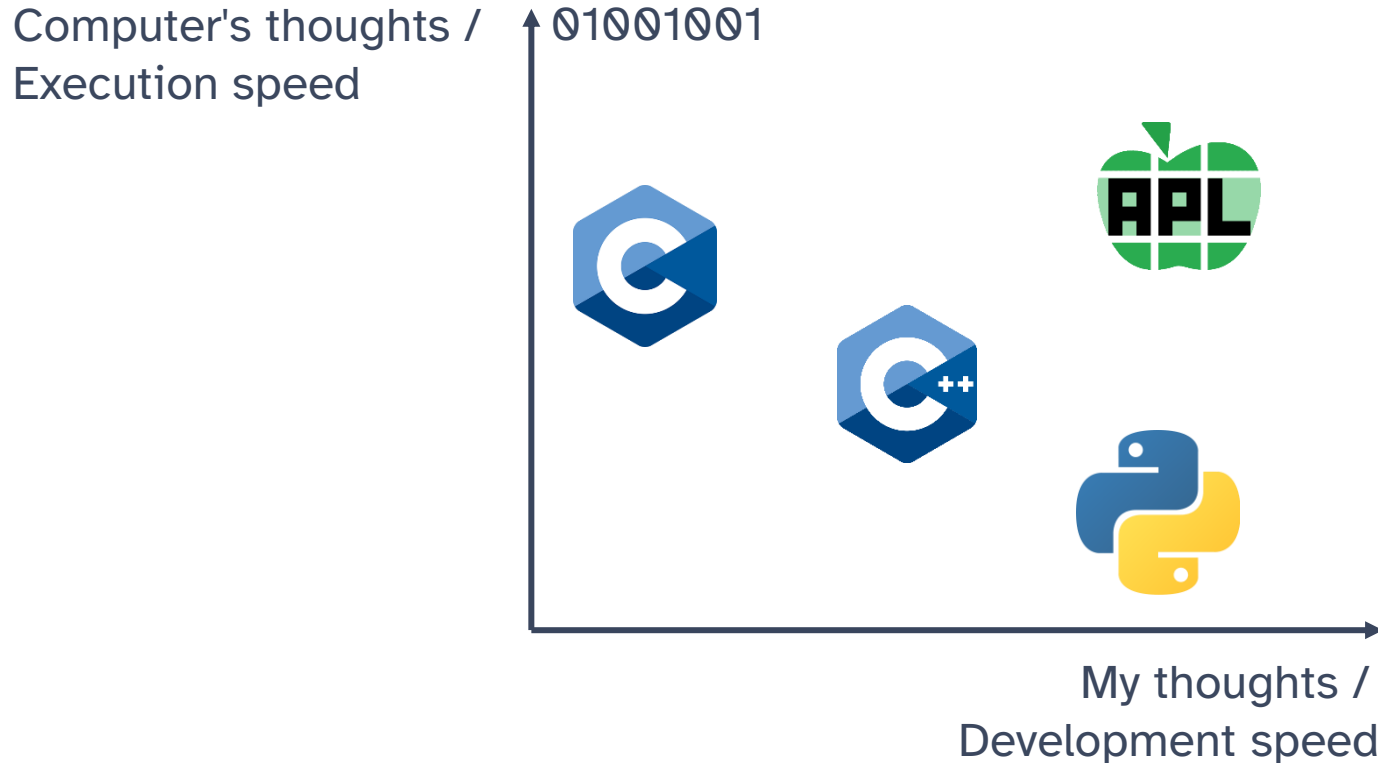
| | |
|--|----------------|
| <i>big_nuggets</i> = 0 | Initialisation |
| for <i>nugget</i> in <i>bag_of_nuggets</i> | Iteration |
| if <i>nugget.weight</i> > 50 | Comparison |
| <i>big_nuggets</i> += 1 | Accumulation |
| endif | |
| endfor | |

Spot patterns. Subordinate detail.

```
big_nuggets = 0
for nugget in bag_of_nuggets
  if nugget.weight > 50
    big_nuggets += 1
  endif
endfor

big_nuggets ← +/ bag_of_nuggets.weight > 50
```

Pretense: Striking a Balance



LEGO-brick Programming



LEGO-brick Programming



The Unreasonable Effectiveness

Strip away everything that is not the problem



WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)

Not logged in [Talk](#) [Contributions](#) [Create account](#) [Log in](#)

Article [Talk](#)

Read [Edit](#) [View history](#)



The Unreasonable Effectiveness of Mathematics in the Natural Sciences

From Wikipedia, the free encyclopedia

"**The Unreasonable Effectiveness of Mathematics in the Natural Sciences**" is a 1960 article by the [physicist Eugene Wigner](#).^[1] In the paper, Wigner observes that a [physical theory's mathematical](#) structure often points the way to further advances in that theory and even to [empirical](#) predictions.

Example: Take 4 Words

```
sentence ← 'this is a sentence with words'
```

```
this is a sentence
```

Example: Take 4 Words

```
sentence ← 'this is a sentence with words'
```

⌚ Split into words

```
' '(≠⊥)sentence
```

| | | | | | |
|------|----|---|----------|------|-------|
| this | is | a | sentence | with | words |
|------|----|---|----------|------|-------|

⌚ Take 4 words

```
4↑' '(≠⊥)sentence
```

| | | | |
|------|----|---|----------|
| this | is | a | sentence |
|------|----|---|----------|

⌚ Rejoin into sentence

Example: Take 4 Words

'this' 'is' 'a' 'sentence'

| | | | |
|------|----|---|----------|
| this | is | a | sentence |
|------|----|---|----------|

' ' °, " 'this' 'is' 'a' 'sentence'

| | | | |
|------|----|---|----------|
| this | is | a | sentence |
|------|----|---|----------|

ε' ' °, " 'this' 'is' 'a' 'sentence'

this is a sentence

1↓ε' ' °, " 'this' 'is' 'a' 'sentence'

this is a sentence

Example: Take 4 Words

| | |
|----|-------------|
| | + / 1 2 5 6 |
| 14 | |
| | 1+2+5+6 |
| 14 | |
| | × / 1 2 5 6 |
| 60 | |
| | ⌈ / 1 2 5 6 |
| 6 | |
| | ⌊ / 1 2 5 6 |
| 1 | |

Example: Take 4 Words

'this' 'is' 'a' 'sentence'

| | | | |
|------|----|---|----------|
| this | is | a | sentence |
|------|----|---|----------|

'this' (←, ' ', →) 'that'

this that

⊃(←, ' ', →) / 'this' 'is' 'a' 'sentence'

this is a sentence

Example: Take 4 Words




```

sentence ← 'this is a sentence with words'
' '=sentence
0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0
+\' '=sentence
0 0 0 0 1 1 1 2 2 3 3 3 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 5
4>+\' '=sentence
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
sentence/˜4>+\' '=sentence
this is a sentence
```

Computational Complexity

1 Problem, 4 More Programming Languages (Python vs Kotlin vs F# vs Wolfram)

5,147 views • 11 Apr 2021

 40  0  SHARE  SAVE ...



code_report
27.4K subscribers

SUBSCRIBED



A video taking a look at 4 more programming language solutions (Python, Kotlin, F# & Wolfram Language) to one problem.

SHOW MORE



guibirow 5 hours ago

I wonder how performant all these magic operators are, probably a topic for a future video!



REPLY



Korv Makak 15 hours ago

How easy is it to reason about undocumented apl code?

I mean, it seems elegant and all.. But trying to reason about unknown algorithms seems tough. For me anyway.

Then again, that might not even be the point of the language. I have no idea, heh.



1



REPLY



Richard Park 1 second ago

One thing about APL is that solutions which make best use of the primitives (such as the one shown in this video) are incredibly easy to reason about. The computational complexity in the worst case can be shown as a simple combination of the complexities of the primitives.



REPLY

sentence \neq \approx $4 > + \backslash ' ' =$ sentence

$=$ $O(n)$

$+ \backslash$ $O(n)$

$>$ $O(n)$

$\neq \approx$ $O(n)$

sentence \neq $\{ \} \neq$ sentence

$1 \downarrow \in \{ \} \circ, \{ \} \uparrow \{ \} (\neq \subseteq \vdash)$ sentence

Experimentation at Many Levels of Abstraction

“The psychological profiling [of a programmer] is mostly the ability to shift levels of abstraction, from low level to high level. To see something in the small and to see something in the large.”

— *Jack Woehr. An interview with Donald Knuth. Dr. Dobb's Journal, pages 16–22 (April 1996)*

Experimentation at Many Levels of Abstraction

```
DFPT      Depth  ADT      APL (c.f. XML/JSON)
0      0      (0      adt←(0 (1 2) (3 (4 5) 6) (7 (8 9 10) (11 12 13) 14))
1      | 1      (1      ids←←adt ⍵ 15
2      | 2      2)      d←0 1 2 1 2 3 2 1 2 3 3 2 3 3 2
3      | 1      (3
4      | 2      (4      PATH MATRIX (c.f. Hsu, ARRAY 2016, extended)
5      | 3      5)      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6      | 2      6)      1 1 3 3 3 3 7 7 7 7 7 7 7 7 7
7      | 1      (7      2 4 4 6 8 8 8 11 11 11 14
8      | 2      (8      5 9 10 12 13
9      | 3      9
10     | 3      10) PARENT/SIBLING
11     | 2      (11     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
12     | 3      12     p←0 0 1 0 3 4 3 0 7 8 8 7 11 11 7
13     | 3      13)     l←0 1 2 1 4 5 4 3 8 9 9 8 12 12 11
14     | 2      14))
```

Dialog '18: High-performance Tree Wrangling, the APL Way

Experimentation at Many Levels of Abstraction

"I am supposed to consume that service from the front end. I maybe understand it from a 10,000 ft view but I know very little about how it works internally."




Language



How an APL Prototype Helped Designing a Service // Martin Janiczek // APL Seeds '21

Experimentation at many levels of abstraction



SOLUTIONS

- #1 $\wedge / > = \vdash$
- #2 $(1 = \neq) \cup$
- #3 $\wedge / 2 = / \vdash$
- #4 $\lfloor / = \lceil /$
- #5 $(1 \geq \neq) \rightarrow \boxplus$
- #6 $> \wedge . = \vdash$ R Aaron Hsu
- #7 $\uparrow \equiv \psi$ R Adám Brudzewsky
- #8 $1 \circ \phi \equiv \vdash$ R Bob Therriault

30:53 / 44:37

Algorithms as a Tool of Thought // Conor Hoekstra // APL Seeds '21

Benefits of Array Programming Techniques



A 100ns cache-miss is a
lost opportunity to execute
~1000 instructions on CPU

25:32 / 43:43

Dyalog '18: Rectangles All The Way Down

The image shows a video player interface. The main content is a slide with the text 'A 100ns cache-miss is a lost opportunity to execute ~1000 instructions on CPU'. Below the slide is a video control bar with a red progress line, play/pause buttons, a volume icon, and a timestamp of 25:32 / 43:43. To the right of the control bar is a small inset video showing a person speaking. Below the video player is the title 'Dyalog '18: Rectangles All The Way Down'.

What about normal people stuff?

We've got you covered

Text/CSV/JSON/XML

Databases

Web Services

Containerised Deployment

System Interaction

Foreign Function Interface

Graphics

So who's using this anyway?

Production

Management

Finance

Medicine

Science

Simulation

Computer Science

Why should I start learning today?

Annual APL Problem Solving Competition

Over two months left to enter

Compete for \$\$\$

Referral awards \$\$\$

Previous winners learned APL *while* participating

dyalogaplcompetition.com

Want more?

tryapl.org

apl.wiki

apl.chat

arraycast.com

dyalog.tv

Further Reading

rikedyp.uk/accu

Thank You