A Programming Language for Thinking about Algorithms

Richard Park
A Programming Language for Thinking about Algorithms

Richard Park
What are Algorithms?

A set of instructions to accomplish some task

Something like a recipe
What is Algorithms?

```python
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.

\[
\text{big_nuggets} = 0
\]

\[
\text{for } \text{nugget in bag_of_nuggets} \\
\quad \text{if } \text{nugget.weight} > 50 \\
\quad \quad \text{big_nuggets} += 1 \\
\quad \text{endif}
\]

\text{endfor}

Initialisation
Iteration
Comparison
Accumulation
Spot patterns. Subordinate detail.

```python
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
```
A Programming Language for Thinking about Algorithms

Pretense: Striking a Balance

Computer's thoughts / Execution speed

My thoughts / Development speed

01001001
LEGO-brick Programming
LEGO-brick Programming
The Unreasonable Effectiveness

Strip away everything that is not the problem
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 │
  └────┴──┴─┴────────┘

- Take 4 words
  
  `4↑’(≠⊆⊢)sentence`

  ┌────┬──┬─┬────────┐
  │ this│is│a│sentence │
  └────┴──┴─┴────────┘

- Rejoin into sentence
Example: Take 4 Words

`'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

\[
+ 1 \ 2 \ 5 \ 6 \\
\frac{14}{1+2+5+6} \\
\frac{14}{\times 1 \ 2 \ 5 \ 6} \\
60 \\
\frac{6}{\lceil 1 \ 2 \ 5 \ 6 \rceil} \\
\frac{6}{\lfloor 1 \ 2 \ 5 \ 6 \rfloor} \\
1
\]
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 5 5 5 5 5 5 5
4>+ ' '=sentence
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 0 0
sentence⌿⍨4>+
' '=sentence
this is a sentence
Computational Complexity
1 Problem, 4 More Programming Languages (Python vs Kotlin vs F# vs Wolfram)

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!

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.

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.
sentence =~ 4 > + \ ' =sentence

= \ O(n)
+\ \ O(n)
> \ O(n)
 =~ \ O(n)
sentence≠~4>+\’′=sentence

1↓∈′′.⊂,4↑′′(≠⊆⊢)sentence
“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.”

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."
Experimentation at many levels of abstraction
Benefits of Array Programming Techniques

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

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