

0 1 0 1 \subseteq 3 4 5 6

4	5	6
---	---	---

text

some of your purple berries

A \subseteq Partition:

A left argument must be non-negative integer vector

A new partition when item > previous one (1,2</math> α)

A all 0s omit items

1 1 2 2 \subseteq 3 4 5 6

3	4	5	6
---	---	---	---

1 0 2 2 \subseteq 3 4 5 6

3	5	6
---	---	---

split $\leftarrow\{(\alpha\neq\omega)\subseteq\omega\}$ A α -split of ω

' ' split text

some	of	your	purple	berries
------	----	------	--------	---------

split $\leftarrow \neq\subseteq$ A coded as a fork

' ' split text

some	of	your	purple	berries
------	----	------	--------	---------

A Ex: Write \subseteq in terms of \subseteq

A Ex: Discover something interesting to share

AAAAAAAAAAAAAAAAAAAAAAAAAAAA @

A {val} (mod @ sel) array

A Sub-array selection:

A numeric indexing

A boolean mask

A ...

```

A   Right operand: [sel]ection:

A       array:
A           simple : major cell selection
A           nested : choose / reach

A       function: returns boolean mask

A   Items of [array] are replaced with {val} mod sub,
A   where [sub] is the sub-array of [sel]ected items.

A   One last thing: this commonly occurring form:
A       val ( $\neg$  @ sel) array
A   can be shortened to:
A       (val @ sel) array

A   Ex: write expressions for:
A       vector  $\omega$  with 0s at alternate positions
A       Sentence  $\omega$  with first letter of each word capital
A       Exchanging rows p and q in matrix  $\omega$ 
A       FizzBuzz
A   Ex: Discover something interesting to share

A Implementation:
A 0. Monitor usage in "real" apps.

A Performance improvements:
A Phase 1 (done): split single MAGIC fn into sub-cases so
A   that case-selection occurs in C rather than APL.
A Phase 2 (done): process _function_ right operand in C-
A   code to produce index array.
A Phase 3: Recode popular sub-cases into C to avoid inter-
A   pretative overhead, and Unsharing() where possible.
A Phase 4: Further reduce Unsharing() by overwriting
A   "garbage" right argument in-situ.
A Identifying the right argument as garbage might require
A   some help from the compiler.

A In-situ substitution:
A   0@1 $\neg$  $\omega$    A OK, because  $\neg\omega$  is not shared
A   0@1 $\neg\omega$    A OK, iff  $\omega$  has low refcount
A    $\omega$ +0@1 $\neg\omega$  A NOT OK
A    $v$ +0@1 $\neg\omega$    A NOT OK if  $\omega$  is referenced downstream.
A   1+0@1 $\neg\omega$    A NOT OK if a local error-guard references  $\omega$ 
A   ... etc

```