Pairs

Christophe Rhodes

Motivation

- can (in principle) build all record types out of pairs
- basic building block

Definition

A pair is a 2-tuple of data

tuple an ordered collection

2- with exactly two elements

Operations

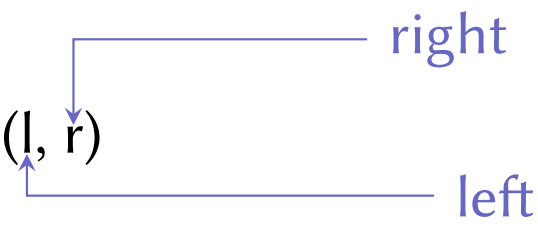
left	return the left element of the pair
right	return the right element of the pair
set-left![o]	set the left element of the pair to o
set-right![o]	set the right element of the pair to o

In pseudocode, respectively:

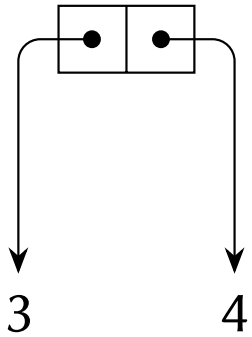
left	LEFT(p)
right	RIGHT(p)
set-left![o]	LEFT(p) ⟵ o
set-right![o]	RIGHT(p) ⟵ o

Constructor:

- **new** Pair(l, r)



Implementation



Complexity analysis

left, set-left!, right, set-right!

1. pointer read (left, right) or write (set-left!, set-right!)

constructor

1. fixed-size (two-word) allocation
2. two pointer writes

Higher-cardinality tuples

(a,b,c)	((a,b),c)
(a,b,c,d)	(((a,b),c),d)
(a,b,…,z)	…((a,b),…z)

Work

1. Implement a pair data structure in Java or C++.
2. Using your pair data structure, implement a triple data structure, with operations first, second, third, corresponding setters, and a constructor with three arguments.
3. Consider the implementation of an N-tuple from pairs. What is the time and space overhead, in terms of N, for the implementation presented in the lecture?
4. Can you come up with an implementation of N-tuples from pairs with a lower time cost? What is the space overhead cost of this implementation?