D’s Gems: Ranges

Part II: Forward, Bidirectional, Random-Access

Dragoș Carp

Range Characteristics

- Convenient
- Composable
- Lazy
- Non-owning
- Non-mutating

Input Ranges

	`empty`	`front`	`popFront`	`save`	`back`	`popBack`	`opIndex`
`InputRange`	■	■	■				
`ForwardRange`	■	■	■	■			
`BidirectionalRange`	■	■	■	■	■	■	
`RandomAccessRange`	■	■	■	■	■	■	■

High Order Ranges

`auto chain(Ranges...)(Ranges rs)`

Spans multiple ranges in sequence. The function `chain` takes any number of ranges and returns a `Chain!(R1, R2,...)` object. The ranges may be different, but they must have the same element type. The result is a range that offers the `front`, `popFront`, and `empty` primitives. If all input ranges offer random access and `length`, `chain` offers them as well.

High Order Ranges

`auto chain(Ranges...)(Ranges rs)`

> Spans multiple ranges in sequence. The function `chain` takes any number of ranges and returns a `Chain!(R1, R2,...)` object. The ranges may be different, but they must have the same element type. The result is a range that offers the `front`, `popFront`, and `empty` primitives. If all input ranges offer random access and `length`, `chain` offers them as well.

Design by Introspection (Dbl)

Querying Ranges

Kind of range:

- `isInputRange`
- `isForwardRange`
- `isBidirectionalRange`
- `isRandomAccessRange`

Capabilities:

- `ElementType`
- `isInfinite`
- `hasAssignableElements`
- `hasLength`
- `hasLvalueElements`
- `hasMobileElements`
- `hasSlicing`
- `hasSwappableElements`

std.algorithm

Searching	`all any balancedParens boyerMooreFinder canFind commonPrefix count countUntil endsWith find findAdjacent findAmong findSkip findSplit findSplitAfter findSplitBefore minCount maxCount minElement maxElement minPos maxPos skipOver startsWith until`
Comparison	`among castSwitch clamp cmp either equal isPermutation isSameLength levenshteinDistance levenshteinDistanceAndPath max min mismatch predSwitch`
Iteration	`cache cacheBidirectional chunkBy cumulativeFold each filter filter-Bidirectional fold group joiner map permutations reduce splitter sum uniq`
Sorting	`multiSort nextEvenPermutation nextPermutation partialSort partition partition3 schwartzSort sort topN topNCopy topNIndex`
Set operations	`cartesianProduct largestPartialIntersection largestPartialIntersection-Weighted nWayUnion setDifference setIntersection setSymmetricDifference`
Mutation	`bringToFront copy fill initializeAll move moveAll moveSome remove reverse strip stripLeft stripRight swap swapRanges uninitializedFill`

std.range

```
chain choose chooseAmong chunks cycle drop dropExactly dropOne enumerate
evenChunks frontTransversal indexed iota lockstep only padLeft padRight
radial recurrence repeat retro roundRobin sequence stride tail take
takeExactly takeNone takeOne tee transposed transversal zip
```

Three-Legged Algorithms

```
size_t bringToFront(Range1, Range2)(Range1 front, Range2 back)
    if (isInputRange!Range1 && isForwardRange!Range2);
```

vs.

```
template <class ForwardIterator>
ForwardIterator rotate(ForwardIterator first,
                       ForwardIterator middle,
                       ForwardIterator last);
```

Contributions

cumulativeFold returns a lazily-evaluated range containing the successive reduced values.

Examples:

```
cumulativeFold!((a, b) => a + b)([1, 2, 3, 4])
```

returns: `1, 3, 6, 10`

```
["foo", "bar", "baz"]
    .cumulativeFold!((a, b) => a + b.length)(0)
```

returns: `3, 6, 9`

DbI in Action (Demo)

https://github.com/dcarp/protobuf/blob/dlang_support/d/src/google/protobuf/package.d#L371

Isn’t it something missing?

Isn’t it something missing?

What about `OutputRange`?

Isn’t it something missing?

`OutputRange`

- It is defined: `put` operation
- Not composable
- `std.range.NullSink`

See also:

- http://tour.dlang.org/tour/en/basics/ranges
- http://tour.dlang.org/tour/en/gems/range-algorithms