if `true` [ ]

Concepts in CUTLASS.CuTe

Basics

`IntTuple`
- `rank(IntTuple)` / `tuple_size` : number of elements
- `get<I>(IntTuple)`
- `depth(IntTuple)`
- `size(InputTuple)` : product of all elements of the IntTuple

`Layout=(Shape, Stride)`
- Layout can be considered as a mapping from coordinates to indices

Special cases:
- vector: any layout with rank = 1
- matrix: any layout with rank = 2

`Tensor`
- Tensor mainly consists of a layout and a data pointer

Layout Concepts

Layout Compatibility

layout A is **compatible** with layout B if the shape of A is compatible with the shape of B.

Shape A is compatible with shape B if
- the size of A is equal to the size of B and
- all coordinates within A are valid coordinates within B.

Compatible is a weak partial order on Shapes as it is **reflexive, antisymmetric, and transitive**.

Layout Coordinates

Every `Layout` accepts coordinates for any `Shape` that is compatible with it.

- 1-D coordinate space
- R-D coordinate space, where R is the rank of the layout
- h-D (natural) coordinate space, where h is “hierarchical”

Layouts provide two fundamental mappings:

- the map from an input coordinate to the corresponding natural coordinate via the `Shape`, and
- the map from a natural coordinate to the index via the `Stride`.

Input Coordinate → Natural Coordinate

Example:

Shape `(3,(2,3))` has three coordinate sets, 1D, 2D, and Natural:

1-D	2-D	Natural		1-D	2-D	Natural
`0`	`(0,0)`	`(0,(0,0))`		`9`	`(0,3)`	`(0,(1,1))`
`1`	`(1,0)`	`(1,(0,0))`		`10`	`(1,3)`	`(1,(1,1))`
`2`	`(2,0)`	`(2,(0,0))`		`11`	`(2,3)`	`(2,(1,1))`
`3`	`(0,1)`	`(0,(1,0))`		`12`	`(0,4)`	`(0,(0,2))`
`4`	`(1,1)`	`(1,(1,0))`		`13`	`(1,4)`	`(1,(0,2))`
`5`	`(2,1)`	`(2,(1,0))`		`14`	`(2,4)`	`(2,(0,2))`

6	(0,2)	(0,(0,1))		15	(0,5)	(0,(1,2))
7	(1,2)	(1,(0,1))		16	(1,5)	(1,(1,2))
8	(2,2)	(2,(0,1))		17	(2,5)	(2,(1,2))

- For this shape, 1D coordinates goes from `0` to `size(Shape)-1=3x2x3-1=17`
- How to translate 1D coordinate to 2D/Natural coordinates?

  "The map from an input coordinate to a natural coordinate is the application of a colexicographical order (reading right to left, instead of "lexicographical," which reads left to right) within the `Shape`."

  "generalized-column-major order"

  - Shape (3, (2, 3)), 1D coordinate 16 ⟶ 2D coordinate (1, 5) ⟶ Naturla (3D) coordinate (1, (1, 2))

    16 div 3 = 5 ··· 1, 2D coordinate is (1, 5),

    5 div 2 = 2 ··· 1, 3D coordinate is (1, (1, 2))
- API: `cute::idx2crd(idx, shape)`

Natural Coordinate ⟶ Index

The map from a natural coordinate to an index is performed by taking the inner product of the natural coordinate with the `Layout`'s `Stride`.

- Layout = (Shape, Stride) = `((3,(2,3), (3, (12, 1))))`, natural coordinate `(i, (j, k))` ⟶ index `i*3 + j*12 + k*1`
- Example:

```
       0     1     2     3     4     5     <== 1-D col coord
     (0,0) (1,0) (0,1) (1,1) (0,2) (1,2)   <== 2-D col coord (j,k)
    +-----+-----+-----+-----+-----+-----+
0   |  0  |  12 |  1  |  13 |  2  |  14 |
    +-----+-----+-----+-----+-----+-----+
1   |  3  |  15 |  4  |  16 |  5  |  17 |
    +-----+-----+-----+-----+-----+-----+
2   |  6  |  18 |  7  |  19 |  8  |  20 |
    +-----+-----+-----+-----+-----+-----+
```

- API: `cute::crd2idx(coord, shape, stride)`

Layout Manipulation

- sublayout
- concat
- grouping and flattening
- slicing