# Package 'h3r'

March 12, 2024

**Type** Package

**Title** Hexagonal Hierarchical Geospatial Indexing System

**Version** 0.1.1

**Date** 2024-03-12

**Description** Provides access to Uber's 'H3' geospatial indexing system via 'h3lib' <https://CRAN.R-project.org/package=h3lib>. 'h3r' is designed to mimic the 'H3' Application Programming Interface (API) <https://h3geo.org/docs/api/indexing/>, so that any function in the API is also available in 'h3r'.

**License** MIT + file LICENSE

**URL** https://symbolixau.github.io/h3r/

**BugReports** https://github.com/symbolixau/h3r/issues

**Encoding** UTF-8

**LazyData** true

**Depends** h3lib (>= 0.1.3), R (>= 2.10)

**LinkingTo** h3lib

**Suggests** sfheaders, tinytest

**RoxygenNote** 7.3.1

**NeedsCompilation** yes

**Author** David Cooley [aut, cre],
Ray Shao [aut]

**Maintainer** David Cooley <dcooley@symbolix.com.au>

**Repository** CRAN

**Date/Publication** 2024-03-12 13:20:02 UTC

## R topics documented:

---

`areNeighborCells` *Are neighbor cells*

---

## Description

Returns whether or not the provided H3 cell indexes are neighbors

## Usage

```
areNeighborCells(origin, destination)
```

## Arguments

| | |
|---|---|
| `origin` | vector of origin H3 cell indexes |
| `destination` | vector of destination H3 cell indexes |

## Value

1 if the indexes are neighbors, 0 otherwise.

## Examples

```
areNeighborCells(
  origin = c("85283473fffffff","85283473fffffff")
  , destination = c("85283471fffffff","85283477fffffff")
 )
```

---

`cellAreaKm2` *Exact area of specific cell in square kilometers.*

---

## Description

Exact area of specific cell in square kilometers.

## Usage

```
cellAreaKm2(cell)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |

### Value

the exact area of specific cell in square kilometers.

### Examples

```
cellAreaKm2(cell = c("8cbe63562a54bff","8cbe635631103ff"))
```

---

## cellAreaM2 *Exact area of specific cell in square meters.*

---

### Description

Exact area of specific cell in square meters.

### Usage

```
cellAreaM2(cell)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |

### Value

the exact area of specific cell in square meters.

### Examples

```
cellAreaM2(cell = c("8cbe63562a54bff","8cbe635631103ff"))
```

## cellAreaRads2 *Exact area of specific cell in square radians.*

### Description

Exact area of specific cell in square radians.

### Usage

```
cellAreaRads2(cell)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |

### Value

the exact area of specific cell in square radians.

### Examples

```
cellAreaRads2(cell = c("8cbe63562a54bff","8cbe635631103ff"))
```

## cellsToDirectedEdge *Cells to directed edge*

### Description

Returns a unidirectional edge H3 index based on the provided origin and destination.

### Usage

```
cellsToDirectedEdge(origin, destination)
```

### Arguments

| | |
|---|---|
| origin | vector of origin H3 cell indexes |
| destination | vector of destination H3 cell indexes |

### Value

a unidirectional edge H3 index based on the provided origin and destination.

## Examples

```
cellsToDirectedEdge(
  origin = c("85283471fffffff","85283473fffffff")
  , destination = c("85283475fffffff","85283477fffffff")
 )
```

---

`cellToBoundary` *Cell To Boundary*

---

## Description

Cell To Boundary

## Usage

```
cellToBoundary(cell)
```

## Arguments

| | |
|---|---|
| `cell` | vector of H3 cells |

## Value

named list, each element named with the input H3 cell, and containing a `lat` and `lng` vector

## Examples

```
cellToBoundary(cell = c("8cbe63562a54bff","8cbe635631103ff"))
```

---

`cellToCenterChild` *Provies the center child index contained by* `cell` *at the* `childRes` *resolution*

---

## Description

Provies the center child index contained by `cell` at the `childRes` resolution

## Usage

```
cellToCenterChild(cell, childRes)
```

### Arguments

| | |
|---|---|
| `cell` | vector of H3 cells |
| `childRes` | integer vector specifying the child resolution for each `cell` |

### Value

index of the child cells

### Examples

```
cellToCenterChild(
  cell = c("85283473fffffff","85283473fffffff")
  , childRes = c(7L, 8L)
 )
```

---

## `cellToChildPos` *Returns the position of the child cell within an ordered list of all children of the cell's parent at the specified resolution parentRes.*

---

### Description

Returns the position of the child cell within an ordered list of all children of the cell's parent at the specified resolution parentRes.

### Usage

```
cellToChildPos(cell, parentRes)
```

### Arguments

| | |
|---|---|
| `cell` | vector of H3 cells |
| `parentRes` | integer vector specifying the parent resolution for each `cell` |

### Value

the position of the child cell

### Examples

```
cellToChildPos(
  cell = c("8cbe63562a54bff","8cbe635631103ff")
  , parentRes = c(1L, 2L)
 )
```

## cellToChildren *Cell To Children*

### Description

Returns all the H3 indexes contained by the input `cell` at the defined child resolution

### Usage

```
cellToChildren(cell, childRes)
```

### Arguments

| | |
|---|---|
| `cell` | vector of H3 cells |
| `childRes` | integer vector specifying the child resolution for each `cell` |

### Value

a named list, where each element is the input `cell`, and the values of each element are the child H3 cells

### Examples

```
cell <- "8cbe63562a54bff"
currentResolution <- getResolution(cell = cell)

cellToChildren(cell, childRes = currentResolution + 1L)
cellToChildren(cell, childRes = currentResolution + 2L)

res0 <- getRes0Cells()
cellToChildren(res0[1], 1L)
cellToChildren(res0[1], 2L)

cellToChildren(res0[1:5], 1L:5L)
```

## cellToLatLng *Cell To Lat Lon*

### Description

Finds the center of the cell in grid space

### Usage

```
cellToLatLng(cell)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |

### Value

a list of two vectors, lat and lng, each the same length as cell, giving the center of cell

### Examples

```
cellToLatLng(cell = c("8cbe63562a54bff","8cbe635631103ff"))
```

---

## cellToLocalIj *Cell to Local IJ*

---

### Description

Produces local IJ coordinates for an H3 index anchored by an origin.

### Usage

```
cellToLocalIj(origin, cell)
```

### Arguments

| | |
|---|---|
| origin | vector of anchor cell |
| cell | vector of cell you input |

### Value

(i, j) coordinates

### Examples

```
cellToLocalIj(
  origin = c("85283473fffffff","85283473fffffff")
  , cell = c("8528342bfffffff","85283477fffffff")
 )
```

## cellToParent *Cell To Parent*

### Description

Provides the parent (coarser) index containing `cell`

### Usage

```
cellToParent(cell, parentRes)
```

### Arguments

| | |
|---|---|
| `cell` | vector of H3 cells |
| `parentRes` | integer vector specifying the parent resolution for each `cell` |

### Value

vector of parent cells for each `cell`

### Examples

```
cell <- "8cbe63562a54bff"
getResolution(cell = cell)

## The `parentRes` should be a lower value than the result of  `getResolution()`
cellToParent(cell = rep(cell, 2), parentRes = c(11L, 10L))

## Specifying a single resolution
cells <- c("8cbe63562a54bff", "8cbe635631103ff")
getResolution(cell = cells)

cellToParent(cell = cells, parentRes = 6)
cellToParent(cell = cells, parentRes = 7)
```

## cellToVertex *Cell To Vertex*

### Description

Returns the index for the specified cell vertex. Valid vertex numbers are between 0 and 5 (inclusive) for hexagonal cells, and 0 and 4 (inclusive) for pentagonal cells.

### Usage

```
cellToVertex(cell, vertexNum)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |
| vertexNum | integer giving the vertex number of the index to return |

### Value

vector of vertex indexes

### Examples

```
cellToVertex(
  cell = c(rep("8cbe63562a54bff", 6))
  , vertexNum = c(0L:5L)
  )
```

---

## cellToVertexes *Cell To Vertexes*

---

### Description

Returns the indexes for all vertices of the given `cell`. The Output will have a 0 in the result if the input `cell` is a pentagon

### Usage

```
cellToVertexes(cell)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |

### Value

list of vectors giving the vertices of each cell. Each list element corresponds to the cell index given in the `cell` argument, and each element of the vector are the cell vertexes.

### Examples

```
cellToVertexes(cell = c("8cbe63562a54bff", "8cbe635631103ff") )
```

`childPosToCell` *Returns the child cell at a given position within an ordered list of all children of parent at the specified resolution childRes.*

## Description

Returns the child cell at a given position within an ordered list of all children of parent at the specified resolution childRes.

## Usage

```
childPosToCell(childPos, cell, childRes)
```

## Arguments

| | |
|---|---|
| `childPos` | the position of the child cell |
| `cell` | vector of H3 cells |
| `childRes` | integer vector specifying the child resolution for each `cell` |

## Value

the position of the child cell

## Examples

```
childPosToCell(
  childPos = c(42, 41)
  , cell = c("85283473fffffff","85283473fffffff")
  , childRes = c(7L, 7L)
)
```

`compactCells` *Compacts the set cellSet of indexes as best as possible, into the array compactedSet.*

## Description

Compacts the set cellSet of indexes as best as possible, into the array compactedSet.

## Usage

```
compactCells(cellSet)
```

### Arguments

| | |
|---|---|
| cellSet | list of character vectors containing to be compacted H3 cell indexes |

### Value

a list of character vectors containing the compacted H3 cell indexes

### Examples

```
compactCells(gridDisk(cell = c("8cbe63562a54bff","8cbe635631103ff"), k = c(1L, 2L)))
```

---

## degsToRads *Degrees To Rads*

---

### Description

Converts Degrees to Radians

### Usage

```
degsToRads(deg)
```

### Arguments

| | |
|---|---|
| deg | vector of degrees |

### Value

numeric vector giving the input deg values as radians

### Examples

```
degsToRads(deg = seq(0, 360, by = 15))
```

## directedEdgeToBoundary *Directed edge To Boundary*

### Description

Directed edge To Boundary

### Usage

```
directedEdgeToBoundary(edge)
```

### Arguments

| | |
|---|---|
| edge | unidirectional edge |

### Value

named list, each element named with the input H3 cell, and containing a `lat` and `lng` vector

### Examples

```
directedEdgeToBoundary(edge = c("115283473fffffff","115283477fffffff"))
```

## directedEdgeToCells *Directed Edge To Cells*

### Description

Get the origin and destination cells of the unidirectional edge

### Usage

```
directedEdgeToCells(edge)
```

### Arguments

| | |
|---|---|
| edge | vector of unidirectional edge H3 indexes |

### Value

the origin, destination pair of hexagon IDs for the given edge ID

### Examples

```
directedEdgeToCells(edge = c("115283473fffffff","115283471fffffff"))
```

---

edgeLengthKm *Get the exact edge length of specific unidirectional edge in kilometers.*

---

### Description

Get the exact edge length of specific unidirectional edge in kilometers.

### Usage

```
edgeLengthKm(edge)
```

### Arguments

| | |
|---|---|
| edge | vector of unidirectional H3 edges |

### Value

the exact edge length of specific unidirectional edge in kilometers.

### Examples

```
edgeLengthKm(edge = c("13d2a1672b34ffff","16a2a1072b59ffff"))
```

---

edgeLengthM *Get the exact edge length of specific unidirectional edge in meters.*

---

### Description

Get the exact edge length of specific unidirectional edge in meters.

### Usage

```
edgeLengthM(edge)
```

### Arguments

| | |
|---|---|
| edge | vector of unidirectional H3 edges |

## Value

the exact edge length of specific unidirectional edge in meters.

## Examples

```
edgeLengthM(edge = c("13d2a1672b34ffff","16a2a1072b59ffff"))
```

---

`edgeLengthRads` *Get the exact edge length of specific unidirectional edge in radians.*

---

## Description

Get the exact edge length of specific unidirectional edge in radians.

## Usage

```
edgeLengthRads(edge)
```

## Arguments

| | |
|---|---|
| edge | vector of unidirectional H3 edges |

## Value

the exact edge length of specific unidirectional edge in radians.

## Examples

```
edgeLengthRads(edge = c("13d2a1672b34ffff","16a2a1072b59ffff"))
```

---

`getBaseCellNumber` *Get Base Cell Number*

---

## Description

Returns the base cell number of the index

## Usage

```
getBaseCellNumber(cell)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |

### Value

a vector the same length as cell giving the base cell number of each index

### Examples

```
getBaseCellNumber(cell = c("8cbe63562a54bff","8cbe635631103ff"))
```

---

## getDirectedEdgeDestination
*Get Directed Edge Destination*

---

### Description

Get the destination cell of the unidirectional edge

### Usage

```
getDirectedEdgeDestination(edge)
```

### Arguments

| | |
|---|---|
| edge | vector of unidirectional edge H3 indexes |

### Value

the destination hexagon from the unidirectional edge H3Index.

### Examples

```
getDirectedEdgeDestination(edge = c("115283473fffffff","16a2a1072b59ffff"))
```

## getDirectedEdgeOrigin *Get Directed Edge Origin*

### Description

Get the origin cell of the unidirectional edge

### Usage

```
getDirectedEdgeOrigin(edge)
```

### Arguments

| | |
|---|---|
| edge | vector of unidirectional edge H3 indexes |

### Value

the origin hexagon from the unidirectional edge H3Index.

### Examples

```
getDirectedEdgeOrigin((edge = c("115283473fffffff","16a2a1072b59ffff")))
```

## getHexagonAreaAvgKm2 *Get the average hexagon area in square kilometers at the given resolution. Excludes pentagons.*

### Description

Get the average hexagon area in square kilometers at the given resolution. Excludes pentagons.

### Usage

```
getHexagonAreaAvgKm2(resolution)
```

### Arguments

| | |
|---|---|
| resolution | cell resolution |

### Value

Average hexagon area in square kilometers at the given resolution. Excludes pentagons.

## Examples

```
getHexagonAreaAvgKm2(resolution = c(12L,10L))
```

---

`getHexagonAreaAvgM2` *Get the average hexagon area in square meters at the given resolution. Excludes pentagons.*

---

## Description

Get the average hexagon area in square meters at the given resolution. Excludes pentagons.

## Usage

```
getHexagonAreaAvgM2(resolution)
```

## Arguments

| | |
|---|---|
| `resolution` | cell resolution |

## Value

Average hexagon area in square meters at the given resolution. Excludes pentagons.

## Examples

```
getHexagonAreaAvgM2(resolution = c(12L,10L))
```

---

`getHexagonEdgeLengthAvgKm` *Get the average hexagon edge length in kilometers at the given resolution. Excludes pentagons.*

---

## Description

Get the average hexagon edge length in kilometers at the given resolution. Excludes pentagons.

## Usage

```
getHexagonEdgeLengthAvgKm(resolution)
```

## Arguments

| | |
|---|---|
| `resolution` | cell resolution |

## Value

Average hexagon edge length in kilometers at the given resolution. Excludes pentagons.

## Examples

```
getHexagonEdgeLengthAvgKm(resolution = c(12L,10L))
```

---

`getHexagonEdgeLengthAvgM` *Get the average hexagon edge length in meters at the given resolution. Excludes pentagons.*

---

## Description

Get the average hexagon edge length in meters at the given resolution. Excludes pentagons.

## Usage

```
getHexagonEdgeLengthAvgM(resolution)
```

## Arguments

| | |
|---|---|
| `resolution` | cell resolution |

## Value

Average hexagon edge length in meters at the given resolution. Excludes pentagons.

## Examples

```
getHexagonEdgeLengthAvgM(resolution = c(12L,10L))
```

## getIcosahedronFaces *Get Icosahedron Faces*

### Description

Find all icosahedron faces intersected by a given H3 index. Faces are represented as integers from 0-19, inclusive. The array is sparse, and empty (no intersection) array values are represented by -1.

### Usage

```
getIcosahedronFaces(cell)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |

### Value

list of vectors. Each list element corresponds to the input `cell` values. Each vector in a list element gives the faces intersected by the `cell`

### Examples

```
getIcosahedronFaces(cell = c("8cbe63562a54bff","8cbe635631103ff"))

getIcosahedronFaces(cell = cellToParent(c("8cbe63562a54bff" ,"8cbe635631103ff"), c(7L, 7L)))
```

## getNumCells *Get the number of unique H3 indexes at the given resolution.*

### Description

Get the number of unique H3 indexes at the given resolution.

### Usage

```
getNumCells(resolution)
```

### Arguments

| | |
|---|---|
| resolution | cell resolution |

### Value

the number of unique H3 indexes at the given resolution

## Examples

```
getNumCells(resolution = c(12L,10L))
```

---

`getPentagons` *Get all the pentagon H3 indexes at the specified resolution.*

---

## Description

Get all the pentagon H3 indexes at the specified resolution.

## Usage

```
getPentagons(resolution)
```

## Arguments

| | |
|---|---|
| `resolution` | cell resolution |

## Value

all the pentagon H3 indexes at the specified resolution.

## Examples

```
getPentagons(resolution = c(12L,10L))
```

---

`getRes0Cells` *Get all the resolution 0 H3 indexes.*

---

## Description

Get all the resolution 0 H3 indexes.

## Usage

```
getRes0Cells()
```

## Value

all the resolution 0 H3 indexes.

### Examples

```
getRes0Cells()
```

---

## getResolution *Get Resolution*

---

### Description

Returns the resolution of the index.

### Usage

```
getResolution(cell)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |

### Value

a vector the same length as `cell` giving the resolution of each index

### Examples

```
getResolution(cell = c("8cbe63562a54bff","8cbe635631103ff"))
```

---

## greatCircleDistanceKm *Great Circle Distance In Kilometers*

---

### Description

Gives the "great circle" or "haversine" distance between pairs of lat/lng coordinates in kilometers.

### Usage

```
greatCircleDistanceKm(aLat, aLng, bLat, bLng)
```

### Arguments

| | |
|---|---|
| aLat | vector of latitude cooridnates (from) |
| aLng | vector of longitude coordinates (from) |
| bLat | vector of latitude coordinates (to) |
| bLng | vector of longitude coordinates (to) |

## Value

numeric vector giving the great circle distance in kilometres

## Examples

```
greatCircleDistanceKm(
  aLat = c(-37.820197)
  , aLng = c(144.983324)
  , bLat = c(-37.818476)
  , bLng = c(144.967354)
)
```

---

# greatCircleDistanceM *Great Circle Distance In Meters*

---

## Description

Gives the "great circle" or "haversine" distance between pairs of lat/lng coordinates in meters.

## Usage

```
greatCircleDistanceM(aLat, aLng, bLat, bLng)
```

## Arguments

| | |
|---|---|
| aLat | vector of latitude cooridnates (from) |
| aLng | vector of longitude coordinates (from) |
| bLat | vector of latitude coordinates (to) |
| bLng | vector of longitude coordinates (to) |

## Value

numeric vector giving the great circle distance in metres

## Examples

```
greatCircleDistanceM(
  aLat = c(-37.820197)
  , aLng = c(144.983324)
  , bLat = c(-37.818476)
  , bLng = c(144.967354)
)
```

greatCircleDistanceRads *Great Circle Distance In Radians*

## Description

Gives the "great circle" or "haversine" distance between pairs of lat/lng coordinates in radians

## Usage

```
greatCircleDistanceRads(aLat, aLng, bLat, bLng)
```

## Arguments

| | |
|---|---|
| aLat | vector of latitude cooridnates (from) |
| aLng | vector of longitude coordinates (from) |
| bLat | vector of latitude coordinates (to) |
| bLng | vector of longitude coordinates (to) |

## Value

numeric vector giving the great circle distance in radians

## Examples

```
greatCircleDistanceRads(
  aLat = c(-37.820197)
  , aLng = c(144.983324)
  , bLat = c(-37.818476)
  , bLng = c(144.967354)
)
```

gridDisk *Grid Disk*

## Description

Get indices within k distance of the origin index.

## Usage

```
gridDisk(cell, k)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |
| k | int distance |

### Details

Elements of the output array may be left as zero, which can happen when crossing a pentagon.

k-ring 0 is defined as the origin index, k-ring 1 is defined as k-ring 0 and all neighboring indexes, and so on.

### Value

the indices within k distance of the origin index

### Examples

```
gridDisk(cell = c("8cbe63562a54bff","8cbe635631103ff"), k = c(1L, 2L))
```

---

## gridDiskDistances *Grid Disk Distances*

---

### Description

Get indices within k distance of the origin index.

### Usage

```
gridDiskDistances(cell, k)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |
| k | int distance |

### Details

k-ring 0 is defined as the origin index, k-ring 1 is defined as k-ring 0 and all neighboring indexes, and so on.

### Value

indices within k distance of the origin index.

## Examples

```
gridDiskDistances(cell = c("8cbe63562a54bff","8cbe635631103ff")
                  , k = c(1L, 2L))
```

---

`gridDistance` *Grid Distance*

---

## Description

Provides the distance in grid cells between the two indexes.

## Usage

```
gridDistance(origin, destination)
```

## Arguments

| | |
|---|---|
| `origin` | vector of origin H3 cell indexes |
| `destination` | vector of destination H3 cell indexes |

## Value

the grid distance between the two H3 cells

## Examples

```
gridDistance(origin = c("85283473fffffff","85283473fffffff"),
             destination = c("8528342bfffffff","85283477fffffff"))
```

---

`gridPathCells` *Grid Path Cells*

---

## Description

Given two H3 indexes, return the line of indexes between them (inclusive).

## Usage

```
gridPathCells(origin, destination)
```

### Arguments

| | |
|---|---|
| origin | vector of origin H3 cell indexes |
| destination | vector of destination H3 cell indexes |

### Details

This function may fail to find the line between two indexes, for example if they are very far apart. It may also fail when finding distances for indexes on opposite sides of a pentagon.

### Value

the line of indexes between the two H3 cells

### Examples

```
gridPathCells(origin = c("85283473fffffff","85283473fffffff")
              , destination = c("85283471fffffff","85283477fffffff"))
```

---

## gridRingUnsafe *Grid Ring Unsafe*

---

### Description

Produces the hollow hexagonal ring centered at origin with sides of length k.

### Usage

```
gridRingUnsafe(cell, k)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |
| k | side length |

### Value

the indices of the hollow hexagonal ring centered at origin with sides of length k.

### Examples

```
gridRingUnsafe(cell = c("8cbe63562a54bff","85283473fffffff"), k = c(2L, 1L))
```

## isPentagon *Is Pentagon*

### Description

Returns non-zero if this index represents a pentagonal cell.

### Usage

```
isPentagon(cell)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |

### Value

a vector the same length as `cell` indicating if the cell is Class II

### Examples

```
isPentagon(cell = c("8cbe63562a54bff","8cbe635631103ff"))
```

## isResClassIII *Is Res Class III*

### Description

Returns non-zero if this index has a resolution with Class III orientation.

### Usage

```
isResClassIII(cell)
```

### Arguments

| | |
|---|---|
| cell | vector of H3 cells |

### Value

a vector the same length as `cell` indicating if the cell is Class II

## Examples

```
isResClassIII(cell = c("8cbe63562a54bff","8cbe635631103ff"))

hex8 <- "88bf4ac0cdfffff"
hex9 <- "89bf4ac0cd7ffff"
hex10 <- "8abf4ac0cd67fff"
isResClassIII(cell = c(hex8, hex9, hex10))
```

---

# isValidCell *Is Valid Cell*

---

## Description

Returns non-zero if this is a valid H3 cell index

## Usage

```
isValidCell(cell)
```

## Arguments

| | |
|---|---|
| cell | vector of H3 cells |

## Value

a vector the same length as `cell` giving the validity of the cell

## Examples

```
isValidCell(cell = c("8cbe63562a54bff","8cbe635631103ff", "abc", "3"))
```

---

# isValidDirectedEdge *Is valid directed edge*

---

## Description

Determines if the provided H3Index is a valid unidirectional edge index.

## Usage

```
isValidDirectedEdge(edge)
```

### Arguments

| | |
|---|---|
| edge | vector of unidirectional edge H3 indexes |

### Value

1 if it is a unidirectional edge H3Index, otherwise 0.

### Examples

```
isValidDirectedEdge(edge = c("13d2a1672b34ffff","16a2a1072b59ffff"))
```

---

## isValidVertex *Is Valid Vertex*

---

### Description

Tests if the given vertex is a valid H3 vertex

### Usage

```
isValidVertex(vertex)
```

### Arguments

| | |
|---|---|
| vertex | H3 Vertex index |

### Value

returns 1 if the given index is a valid H3 vertex

### Examples

```
isValidVertex(vertex = c("24cbe63562a549ff", "abc"))
```

## latLngToCell *lat lng to cell*

### Description

Indexes the location at the specified resolution, returning the index of the cell containing the location. This buckets the geographic point into the H3 grid

### Usage

```
latLngToCell(lat, lng, resolution)
```

### Arguments

| | |
|---|---|
| lat | latitude |
| lng | longitude |
| resolution | cell resolution |

### Value

vector giving the H3 cell for each input lat/lng pair, at the given resolution

### Examples

```
latLngToCell(
  lat = c(-37.820197, -37.818476)
  , lng = c(144.983324, 144.967354)
  , resolution = c(12, 12)
  )
```

## localIjToCell *Local IJ To Cell*

### Description

Produces an H3 index from local IJ coordinates anchored by an origin.

### Usage

```
localIjToCell(origin, i, j)
```

### Arguments

| | |
|---|---|
| origin | vector of anchor cell |
| i | vector of local I coordinate |
| j | vector of local I coordinate |

### Value

cell vector of H3 cells

### Examples

```
localIjToCell(
  origin = c("85283473fffffff","85283473fffffff")
  , i = c(1L, 2L)
  , j = c(2L, 1L)
 )
```

---

## originToDirectedEdges *Origin To Directed Edges*

---

### Description

Get all of the directed edges from an origin

### Usage

```
originToDirectedEdges(origin)
```

### Arguments

| | |
|---|---|
| origin | vector of origin H3 cell indexes |

### Value

a vector for each origin with all of the directed edges from the current H3Index

### Examples

```
originToDirectedEdges(origin = c("85283473fffffff","8cbe635631103ff"))
```

polygonToCells *Polygon to cells*

## Description

Returns the h3 indexes for the input GeoJSON-like data structure

## Usage

```
polygonToCells(polygons, resolution, isLatLng = TRUE)
```

## Arguments

| | |
|---|---|
| polygons | A list of polygons. Each polygon is list of matrices. |
| resolution | The resolution of the output cells |
| isLatLng | TRUE (default) if the coordinates are in lat / lng order. FALSE otherwise |

## Value

h3 indexes for the input GeoJSON-like data structure

## Examples

```
## single polygon
polygon <- list(
  list(
  matrix(
    c(
    37.813318999983238, -122.4089866999972145,
    37.7198061999978478, -122.3544736999993603,
    37.8151571999998453, -122.4798767000009008
    )
    , ncol = 2
    , byrow = TRUE
  )
  )
)

polygonToCells(polygon, resolution = 7L)

## poylgon with a hole
polygon <- list(
list(
  matrix(
    c(
    37.813318999983238, -122.4089866999972145,
    37.7198061999978478, -122.3544736999993603,
    37.8151571999998453, -122.4798767000009008
```

```
    )
    , ncol = 2
    , byrow = TRUE
  ),
  matrix(
    c(
    37.813318999983238, -122.4089866999972145,
    37.7198061999978478, -122.3544736999993603,
    37.8151571999998453, -122.4498767000009008
    )
    , ncol = 2
    , byrow = TRUE
  )
)
)

polygonToCells(polygon, resolution = 7L)

## Many polygons
polygon <- list(
  list(
    matrix(
      c(
      37.813318999983238, -122.4089866999972145,
      37.7198061999978478, -122.3544736999993603,
      37.8151571999998453, -122.4798767000009008
      )
      , ncol = 2
      , byrow = TRUE
    )
   ),
 list(
  matrix(
    c(
    37.813318999983238, -122.4089866999972145,
    37.7198061999978478, -122.3544736999993603,
    37.8151571999998453, -122.4498767000009008
    )
    , ncol = 2
    , byrow = TRUE
    )
  )
)

polygonToCells(polygon, resolution = c(7L, 7L))
```

---

radsToDegs *Rads to Degrees*

---

## Description

Converts Radians to Degrees

## Usage

```
radsToDegs(rad)
```

## Arguments

| | |
|---|---|
| rad | vector of radians |

## Value

numeric vector giving the input rad values as degrees

## Examples

```
radsToDegs(rad = seq(0, 2 * pi, by = (pi / 12) ) )
```

---

# stations *Stations*

---

## Description

A data set of train stations in Melbourne

## Usage

```
stations
```

## Format

**stop_id** The ID of the station

**stop_name** The name of the station

**lat** The latitude coordinate of the station

**lon** The longitude coordinate of the station

## Details

Obtained from https://discover.data.vic.gov.au/dataset/timetable-and-geographic-information-gtfs and distributed under the Creative Commons 4 License https://creativecommons.org/licenses/by/4.0/

## uncompactCells *Uncompacts a set of compacted H3 cell indexes to a given resolution.*

### Description

This function uncompacts the provided set of compacted H3 cell indexes to the specified resolution.

### Usage

```
uncompactCells(compactedSet, resolution)
```

### Arguments

| | |
|---|---|
| compactedSet | list of character vectors containing compacted H3 cell indexes |
| resolution | integer specifying the resolution for the uncompacted cells |

### Value

a list of character vectors containing the uncompacted H3 cell indexes at the specified resolution

### Examples

```
uncompactCells(
  compactCells(
    gridDisk(
      cell = c("85283477fffffff", "85283423fffffff")
      , k = c(1L, 2L)
    )
   )
   , res = c(5L, 5L)
 )
```

## vertexToLatLng *Vertex To Lat Lng*

### Description

Returns the latitude and longitude of the given vertex

### Usage

```
vertexToLatLng(vertex)
```

## Arguments

| | |
|---|---|
| vertex | H3 Vertex index |

## Value

data.frame of the lat/lng coordinates of the input vertex

## Examples

```
vertices <- cellToVertex(
  cell = c(rep("8cbe63562a54bff", 6))
  , vertexNum = c(0L:5L)
  )

vertexToLatLng(vertex = vertices)
```

# Index