Introduction :
Radix tree are space optimized data structures where each node with one child is merged with its parent.Radix trees are useful for constructing associative arrays with keys that can be expressed as strings.They are particularly used in IP routing as IP address have large range of values. For eg :

```
foo/
|\_b
|   \_ar
|    |  \_*
|    \_iz
|        \_*
|        \__ness
|              \_*
\_crap
       \_*
```

This is radix tree that can recognize the strings :
1. /foo/bar
2. /foo/biz
3. /foo/bizness
4. /foo/crap

Use of Radix Trees in Nginx :

The radix tree concept is used in nginx only at one place that is in the HttpGeoIPModule.This module creates and assign values to variables based on the IP address of the request client. For e.g. one of the most common uses is to set the country of the end user as a nginx variable.
The Radix tree is used to parse this IP address and determine the country by comparing it with GeoIP databases.
In nginx implementation of radix tree bits are stored rather than strings.Also the IP address is a 32 bit address.A bit can take only 0 or 1.Thus radix tree will have at most to children left and right which is similar to binary trees.
The radix tree provides functions like insertion,deletion and searching.

Structure of Radix Tree :

To avoid frequent allocation and deallocation of space for ngx_radix_node_t we make the use of memory pool. To allocate space for radix tree, a page is allocated.Nginx defines two structures one or defining the radix tree nodes and other defining the radix tree itself.
The node of a radix tree is defined as :

```
typedef struct ngx_radix_node_s  ngx_radix_node_t;
struct ngx_radix_node_s {
    ngx_radix_node_t  *right;
    ngx_radix_node_t  *left;
    ngx_radix_node_t  *parent;
    uintptr_t          value;
};
```

Here,
right : pointer to the right subtree of the root
left : Pointer to the left subtree of the root

parent :pointer to the parent of the node

To allocate space for radix tree, a page is allocated.The structure of the radix tree is defined as :

```
typedef  struct {
    ngx_radix_node_t * root;
    ngx_pool_t * pool;
    ngx_radix_node_t * free;
    char                * start;
    size_t size;
} ngx_radix_tree_t;
```

Here,
root : The pointer to the root of the Radix tree
pool : The pointer to the memory pool which is used to allocate memory for the radix tree.
free : Pointer to starting of free space in the page
start : space is allocated to the radix tree as a unit called the page.start is the pointer to the position of the page in memory.
size : Represents the size of the remaining available memory space in the page.