A Radix Trie - Version 1.1

1 The `route` structure

Each node has a forwarding table which consists of `route` structures, one for each entry. The `route` structure contains:

route: bit position
destination
output interface
on and off children

2 Insertion

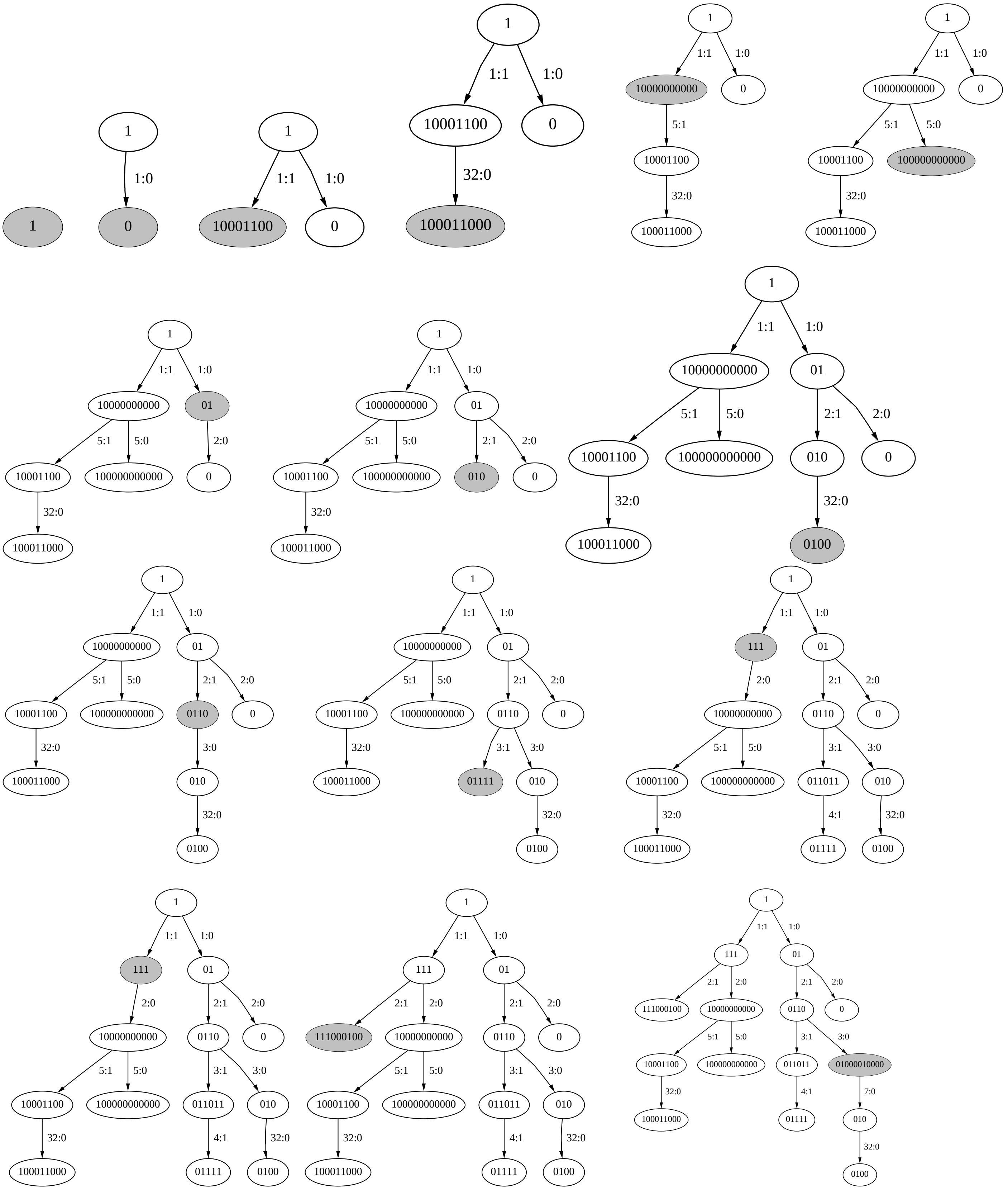
- allocate a new `route` structure
- traverse the tree beginning at the root, call the current `route` the node being visited in the traversal
- find the position of the first bit which differs between the destination of the new node and the destination of the current `route` in the traversal
- if the index of the differing bit position is less than the bit position being examined by the current `route`, insert yourself between the current `route` and its parent. The bit position being compared at the new node is the position of the bit which differed between the new and current node. Insure that the old current `route` in this position is still linked by setting the correct child pointer of the new node.
- if the index of the differing bit is greater than the bit position being examined by the current `route`, continue the traversal by following the edge which has the same bit value as the new destination at that position
- if the current `route` is empty, insert the new node in place. Set the bit position of the new node to the greatest difference between the new node and the parent, or default this to `1` in the initial case where there is no parent.
- if new new node matches an existing key exactly, both the destination and the prefix length. replace it.
- if the new node differs only in its prefix length, insert along the child edge, with the same bit position (the strange case)
- Treat the default route (prefix length 0) as a special case by inserting it into the `default_route` element of the `node` structure.

3 Example

This example represents the incremental construction of a tree through repeated addition of new nodes. Each node is labeled with its bit value, and the length of the binary string represents the prefix length. The remainder of the binary digits to make up the total key length of 32 are assumed to be zero.

The edges between nodes are labeled `x:y` where `x` is the bit position which is tested to determine which edge to follow, and `y` is whether or not that bit was a `0` or `1`.
Each new node inserted into the tree is drawn filled in grey.

Remember that the convention in all these examples is that valid bit positions range from 1 to 32.



1
1 1:0 0
1 1:1 1:0 10001100 0
1 1:1 1:0 10001100 0 32:0 100011000
1 1:1 1:0 10000000000 0 5:1 10001100 32:0 100011000
1 1:1 1:0 10000000000 0 5:1 5:0 10001100 100000000000 32:0 100011000
1 1:1 1:0 10000000000 01 5:1 5:0 2:0 10001100 100000000000 0 32:0 100011000
1 1:1 1:0 10000000000 01 5:1 5:0 2:1 2:0 10001100 100000000000 010 0 32:0 100011000
1 1:1 1:0 10000000000 01 5:1 5:0 2:1 2:0 10001100 100000000000 010 0 32:0 32:0 100011000 0100
1 1:1 1:0 10000000000 01 5:1 5:0 2:1 2:0 10001100 100000000000 0110 0 32:0 3:0 100011000 010 32:0 0100
1 1:1 1:0 10000000000 01 5:1 5:0 2:1 2:0 10001100 100000000000 0110 0 32:0 3:1 3:0 100011000 01111 010 32:0 0100
1 1:1 1:0 111 01 2:0 2:1 2:0 10000000000 0110 0 5:1 5:0 3:1 3:0 10001100 100000000000 011011 010 32:0 4:1 32:0 100011000 01111 0100
1 1:1 1:0 111 01 2:0 2:1 2:0 10000000000 0110 0 5:1 5:0 3:1 3:0 10001100 100000000000 011011 010 32:0 4:1 32:0 100011000 01111 0100
1 1:1 1:0 111 01 2:1 2:0 2:1 2:0 111000100 10000000000 0110 0 5:1 5:0 3:1 3:0 10001100 100000000000 011011 010 32:0 4:1 32:0 100011000 01111 0100
1 1:1 1:0 111 01 2:1 2:0 2:1 2:0 111000100 10000000000 0110 0 5:1 5:0 3:1 3:0 10001100 100000000000 011011 01000010000 32:0 4:1 7:0 100011000 01111 010 32:0 0100

4 Lookup

destination $\rightarrow$ interface

- use the default route as the current best choice
- start visiting nodes at the root of the tree
- if the current node matches the destination (mask and match), and it has a longer prefix length, use that route as the current best
- follow the *on* edge if the value of the destination at the bit position of this node is a one
- follow the *off* edge if the value of the destination at the bit position of this node is a zero
- at the end of this process, the output interface is the one associated with the best route, if there is one

5 Drawing Your Trees

You may consider it useful to draw your tree structures visually while debugging your project. The following instructions should allow you to produce drawings of your tree at each step after insertion:

- visit new page at `http://www.cs.washington.edu/education/courses/461/99au/project.html`
- download and install graph drawing software
  - click on graph drawing software link
  - select `Run this program from current location`
  - click `Yes` to install software
- insert the line `dot` *`filename`* at the head of a test file to write out the graph output to *filename*
- run

  ```
  C:\Program Files\graphviz\bin\dot
  ```

  on the output graph file, passing `-Tps` *`filename`* as arguments
- view output postscript using

  ```
  Programs->Desktop Tools->GS Tools->GSView
  ```

- there should be one page in the output for each addition to your tree