

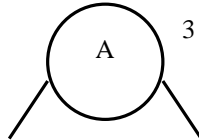
CS209 HOW-TO Series

How to build Patricia Tries

Okay, the most requested one so far. Building Patricia tries is in fact quite simple, and generally just a little bit misunderstood. Before we get into the building, I'll go through the rules:

The Rules of Patricia Tries

1. Every node has a 'bit index'. This is the number sitting next to the node, and is where we do the 'bit comparison' on the search:



In this case, the node 'A' has a bit index of three. Thus, when you are doing a search, and you come to this node, you do the comparison on bit 3 of the key you are searching for, and branch left on a 0 and right on a 1.

2. The bit index of nodes in the trie **MUST** decrease as you move down from the root. Thus, a parent node in the trie will **ALWAYS** have a higher bit index than its children. This is probably the most important point, so keep it in mind.

3. A search on a Patricia trie is concluded when you follow an *upward link*. Wherever you end up after following that link is the result of your search.

4. Keys. The keys in a Patricia trie are bit strings, and the indices are labeled in descending order from left to right. An example is below:

Index:	4	3	2	1	0
Key:	1	0	1	0	0

Building Patricia tries

There are two rules associated with building Patricia tries – the 'existing trie procedure' and the 'new trie procedure'. The 'new trie procedure' is used to establish a new tree, and is only usually used when there is no appropriate branch of the trie to search on. The 'existing trie procedure' is the procedure you will use the most often to insert nodes into your trie.

General rules

These rules always apply when inserting a node in the trie. However, the method in which you apply them will differ from case to case. These will be explained as we go on.

Step 1 – Conduct an unsuccessful search of the trie to find the closest node.

Step 1 – Find the bit index of the new node.

Step 2 – Find position in trie for new node (using the bit index), and insert it.

Step 3 – Establish links.

The new trie procedure

This rule is used when you hit a 'NULL' value in the trie when you do your search for the closest node, i.e. you follow an unconnected branch.

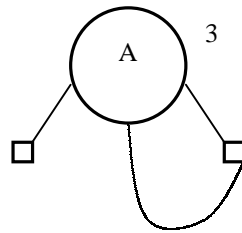
Step 1 – Conduct a search on the trie for the new node. If you a

Step 2 – Bit index of the new node = position of the leftmost '1' in the key.

Example: Key 01100 would have a bit index of 3, since the leftmost 1 is at position 3.

Step 3 – Insert node in trie. There will only be one position to insert it in, as we're doing the insertion on an unconnected branch.

Step 4 – Establish link. For a new node in an unestablished trie, the left link will always be unconnected, and the right link will always be an upward link to itself:



The existing trie procedure

This is the procedure you will use to do most of the insertions on a trie.

Step 1 – Conduct a search for the new node on the trie. The node you end up at is known as the 'closest node', and you will use this to determine the bit index. It's also a good idea to mark (for later reference) the upward link you followed to reach the 'closest node'.

Step 2 – The bit index of the new node = the leftmost bit where the new node's key and the closest node's key differ.

Example: New node: A = 00101 Closest node: B = 00111 The bit index of A will be 1, since the leftmost bit where A's key and B's key differ is at position 1, bolded above.

Step 3 – Insert the node in the tree. Where you insert the node is dependent on the bit index of the new node – if it is lower than that of the closest node, it goes below, otherwise it goes above.

For new bit index < closest bit index: If the bit index of the new node is lower, you can simply insert the new node in place of the upward link you followed on the unsuccessful search.

For new bit index > closest bit index: This can sometimes be a bit tricky – you need to traverse back up the trie to find an appropriate position for the new node. You determine the new position by traversing upwards from the closest node and inserting the new node in the first valid position as specified by the bit index.

Step 4 – Determine links. One of the links will always lead back to the new node, and this is determined by looking at the key of the new node at the new node's bit index. If the new node's key is a '0' at the bit index, the left link will lead back up to the new node; if it is a '1' at the bit index, the right link will lead back upwards. The other link will lead either back to the closest node (if you inserted below or one level above) or alternatively if you had to go up a few levels, the other link will lead to the node below the new one's position.

The example

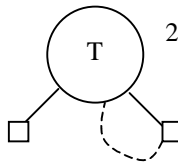
Insert the following pieces of data into a new Patricia trie in the order given:

```
Index:  43210
        T 00100
        O 01111
        E 00101
        A 01001
        S 10110
        Y 11101
```

Insert T - 00100

Since there is no trie to do searching on, we use the 'new trie procedure'.

- 1) Find bit index. In this case it is the index of the leftmost '1', which is at index 2.
- 2) Build links. As we remember, for a new trie node, the leftmost link is unconnected and the right link always leads back up to itself:



Insert O - 01111

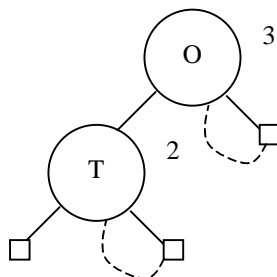
We do a search for O on our existing tree. We start at T, which has a bit index of 2, and branch on bit 2 of our new key. Since bit 2 of O is a 1, we branch right, which leads us to T – thus, T is our closest node, and we use the 'existing trie procedure'

- 1) Find bit index – compare O and T and find the leftmost bit where they differ:

```
      43210
O    01111
T    00100
```

The leftmost bit where the two keys differ is at bit 3 – thus O has a bit index of 3.

- 2) Find position: Since 3 is greater than T's bit index of 2, it must be inserted above T in the trie. We traverse up the trie to find the first available position. In this case, it is on the level directly above T
- 3) Establish links – At bit index 3, O has a 1. This immediately tells us that the right link will lead to O. By a process of elimination, this also tells us that the left link will lead back to T:



Insert E – 00101

As always, we conduct a search on the trie to find E. Now starting at O, we branch left (E has a 0 at bit 3), which leads us to T. At T, we branch right (E is 1 at bit 2), leading us back up to T. Thus, T is our closest node.

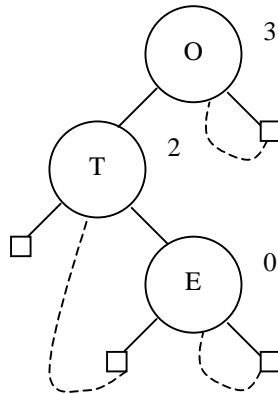
- 1) Find bit index for E – This is the leftmost bit where T and E differ:

```

      43210
E    00101
T    00100
    
```

The leftmost bit where E and T differ is at bit 0, shown in bold. Therefore, E's bit index will be 0.

- 2) Insert node – since E's index 0 is lower than T's index of 2, we can simply insert E at the link that we followed to get back up to T, i.e. T's right link.
 3) Establish links – E has a 1 at its bit index of 0, so it gets the right hand link. By the process of elimination, T gets the left hand link:



Insert A – 01001

As always, we conduct a search for our new node in the trie. A search for A starting at the root, O leads us back up to O (A has a 1 at bit index 3, so we branch right). O is now our closest node.

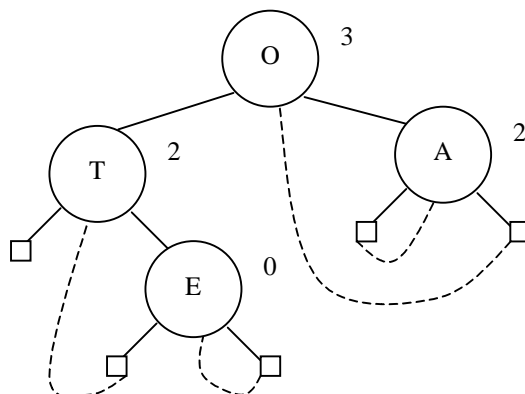
- 1) Find bit index of A = Leftmost bit where O and A differ:

```

      43210
A    01001
O    01111
    
```

The leftmost bit where O and A differ is at bit index 2. Thus, A's bit index is 2.

- 2) Find position – Since A's index 2 is lower than O's 3, we can insert A below O in the trie. Thus, A replaces the link we followed to get to O, which is O's right link.
 3) Establish links – A has a 0 at its bit index of 2, so it gets the left-hand link. O has the 1 at bit index 2, so the right link will lead to O:



Insert S - 10110

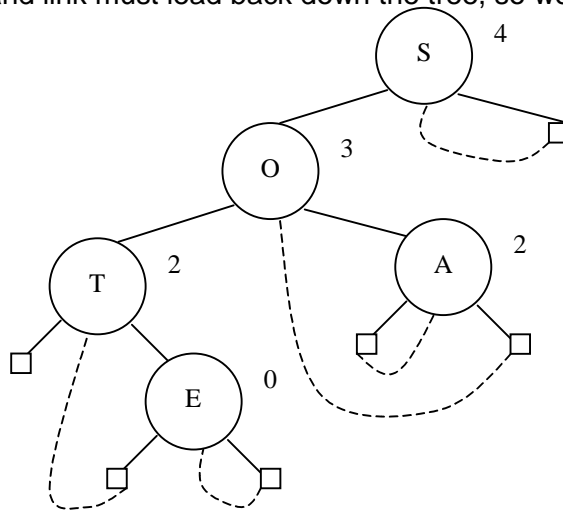
Conducting a search for S in the trie leads us through the path O → T → E → T, so T is our closest node.

- 1) Find bit index. Bit index of S is the leftmost bit where S and T's keys differ:

	43210
S	10110
T	00100

The leftmost bit where the two keys differ is at bit 4. Therefore, T has a bit index of 4.

- 2) Insert S into the tree – S's bit index of 4 means that we must insert S above T, so we traverse up the tree. Going up one level brings us to O, with a bit index of 3. Since S's bit index of 4 is still greater than this, we must traverse up another level. Thus, S will be inserted above O in the tree. Because of this, one of the links must lead back to O.
- 3) Establish links – Since S has the 1 at bit index 4, S's right link will lead back up to O. The left hand link must lead back down the tree, so we connect that to O:



Insert Y – 11101

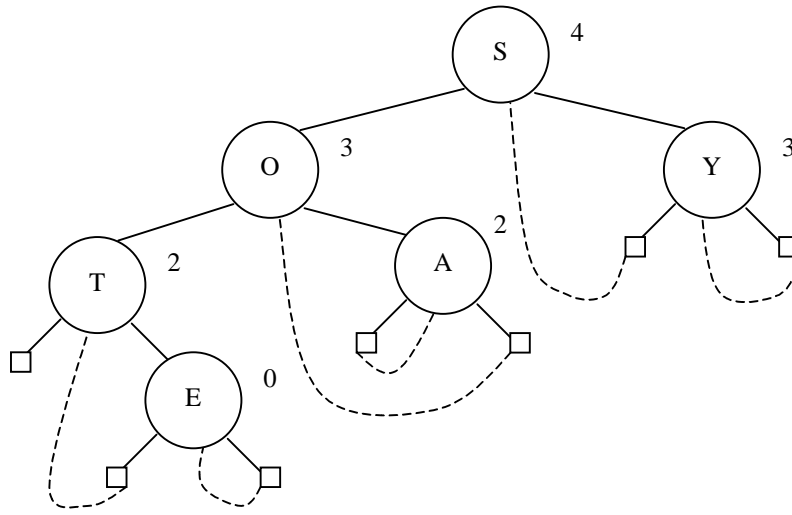
As always we search for our new node in the tree. This leads us back up to S, which is Y's closest node.

- 1) Bit index = leftmost node where S and Y's keys differ:

	43210
Y	11101
S	10110

Thus, Y's bit index is 3

- 2) Insert in tree – 3 is less than S's 4, so we insert below, using the link we followed to get back to S – S's right link.
- 3) Establish links – Since Y has a 1 at bit index 3, the right link leads to Y. S has a 0 at bit 3, so it gets Y's left link:



And there we have our final Patricia trie! That wasn't so bad, was it?

Just note that Patricia tries, like all other digital search tries do not handle duplicates except for internal counts within nodes. So bear in mind that you cannot insert a duplicate key into a Patricia trie without things going very wrong.