*Tìm kiếm khóa (Find)*

```
get(p, key, d) {
   if (p == NULL)             return NULL;
   if (d == strlen(key))      return p;
   c = key[d];
   return get(p->next[c], key, d + 1);
}
findNode(p, key) {
   p = get(p, key, 0);
   return   (p && p->value != NIL) ? p : NULL;
}
```

*Thêm khóa (Insertion)*

```
put(p, key, val, d) {
   if (p == NULL) {
      p = new node;
      p->value = NIL;
      for (int i = 0; i < |Σ|; i++)       p->next[i] = NULL;
   }
   if (d == strlen(key)) {
      p->value = val;
      return p;
   }
   c = key[d];
   p->next[c] = put(p->next[c], key, val, d + 1);
   return p;
}
Insertion(root, key, val) {
   root = put(root, key, val, 0);
}
```

*Tìm các khóa có cùng tiền tố (keysWithPrefix)*

```
collect(p, prefix, d) {
   if (p == NULL)       return;
   if (p->value != NIL)
      cout << p->value << " " << prefix << endl;
   for (c = 0; c < |Σ|; c++)
      collect(p->next[c], <prefix + c>, d + 1);
}
get(p, key, d) {
   if (p == NULL)             return NULL;
   if (d == strlen(key))      return p;
```

```
   c = key[d];
   return get(p->next[c], key, d + 1);
}
keysWithPrefix(root, prefix) {
   ref p = get(root, prefix, 0);
   d = strlen(prefix);
   collect(p, prefix, d);
}
```

*Tìm khóa dài nhất là tiền tố của một chuỗi (longestPrefixOf)*

```
search(p, s, d, length) {
   if (p == NULL)             return length;
   if (p->value != NIL)       length = d;
   if (d == strlen(s))        return length;
   c = s[d];
   return search(p->next[c], s, d + 1, length);
}
longestPrefixOf(root, s) {
   int length = search(root, s, 0, 0);
   return s[0 .. length – 1];
}
```

*Xóa nút trên tries (Deletion)*

```
del(p, key, d) {
   if (p == NULL)             return NULL;
   if (d == strlen(key))
      p->value = NIL;
   else {
      c = key[d];
      p->next[c] = del(p->next[c], key, d + 1);
   }
   if (p->value != NIL)             return p;
   for (c = 0; c < |Σ|; c++)
      if (p->next[c] != NULL)       return p;
   delete   p;
   return NULL;
}
Deletion(root, key) {
   root = del(root, key, 0);
}
```