アルゴリズムとデータ構造

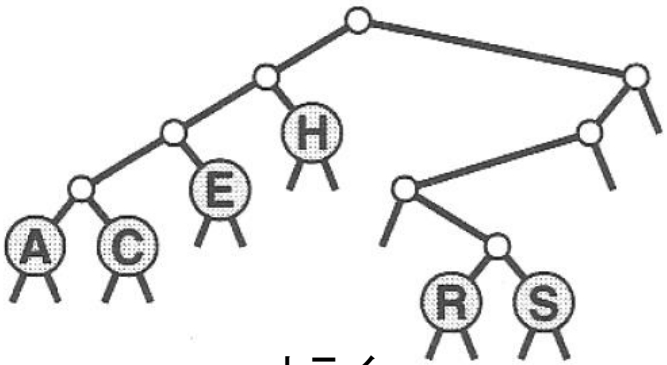
基数探索（その２）

目次

- パトリシア

トライ（Trie）（復習）

- キーを葉にだけ置くとどうなるか？
  - 各キーは，キーの先頭からのビットパタンで表される道の終端の葉に格納される
  - キー全体の比較は，最後の葉でだけ
  - こうすると，葉以外の内部節点では，キーの比較をしなくてよい!

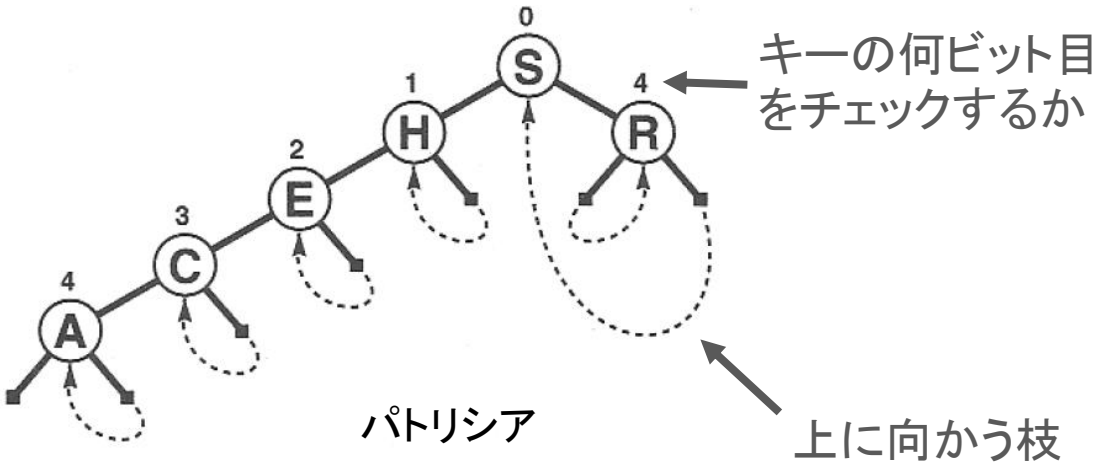


トライ

パトリシア

"Practical Algorithm To Retrieve Information Coded In Alphanumeric"

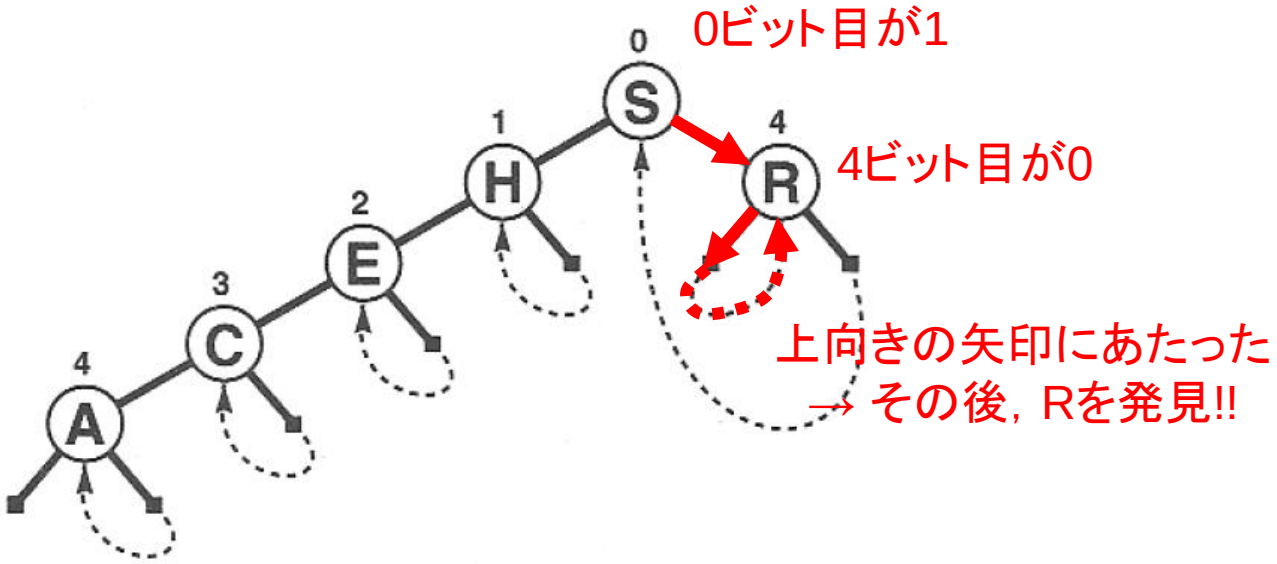
- トライは，キーを持たない内部節点が冗長
- どうにか，内部節点のすべてにキーを持たせることはできないか？



パトリシア

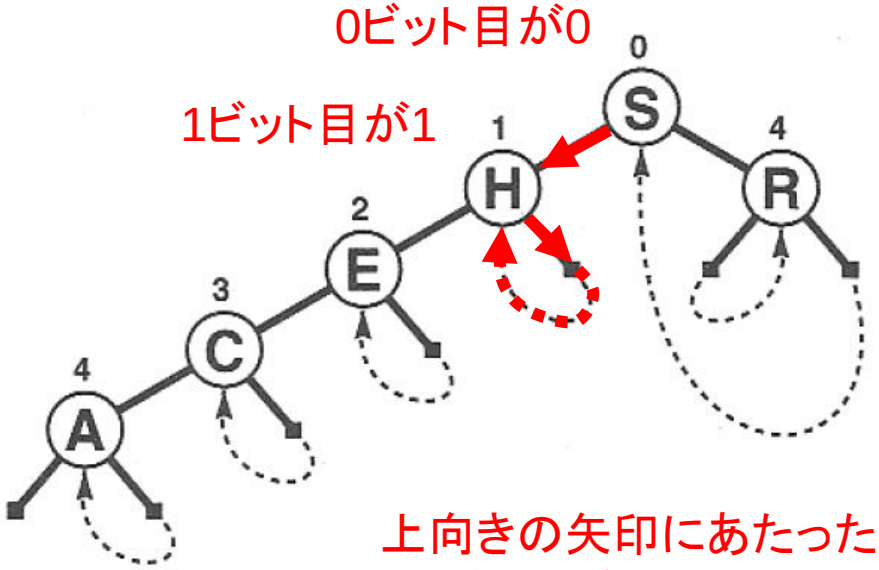
探索

R=10010の探索
　0ビット目=1
　4ビット目=0



探索

I=01001の探索
　0ビット目=0
　1ビット目=1



上向きの矢印にあたった
→ その後，Hを発見
→ Iは見付からなかった

探索

各節点の脇に付随していた番号(何ビット目をチェックするかを表す)

```
Item searchR(link h, Key v, int w)
  {
    if (h->bit <= w) return h->item;
    if (digit(v, h->bit) == 0)
        return searchR(h->l, v, h->bit);
    else return searchR(h->r, v, h->bit);
  }
Item STsearch(Key v)
  { Item t = searchR(head->l, v, -1);
    return eq(v, key(t)) ? t : NULLitem;
  }
```

上向き矢印に来たら，その先の節点を返す

h->bit番目のビットに基づいて，分岐

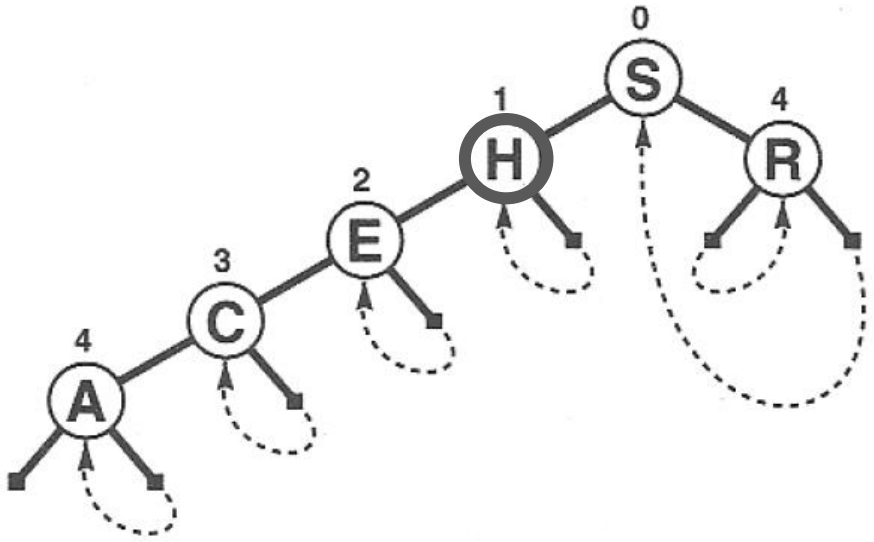
パトリシアの根へのポインタを表す

挿入1

I=01001の挿入

I=01001
H=01000

IとHとで異なるのは，4ビット目だけ

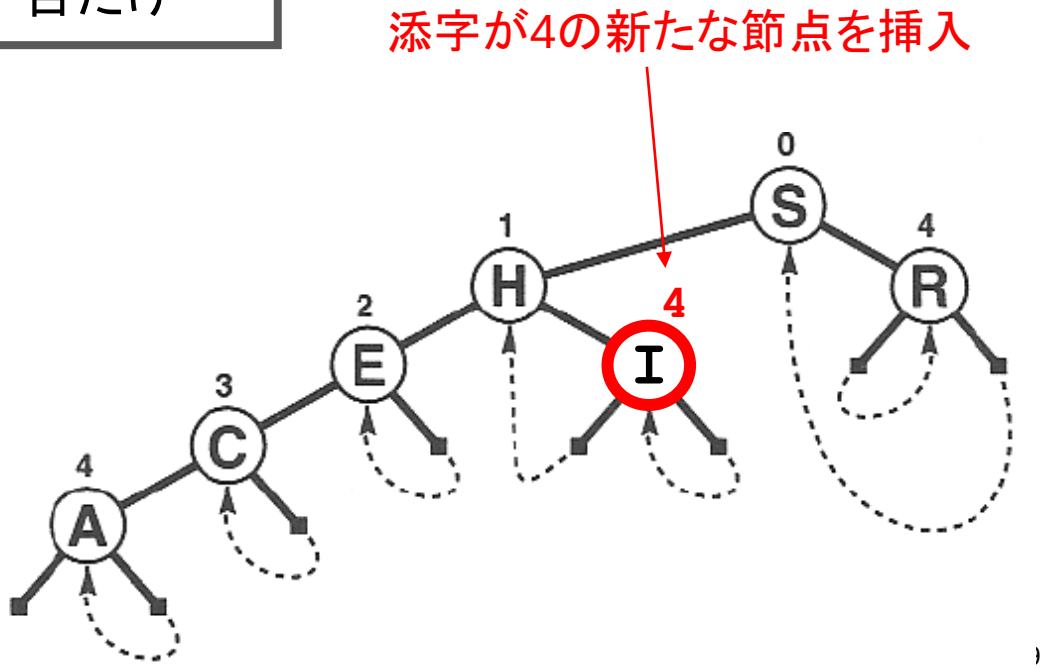


挿入1

I=01001の挿入

I=01001
H=01000

IとHとで異なるのは，4ビット目だけ



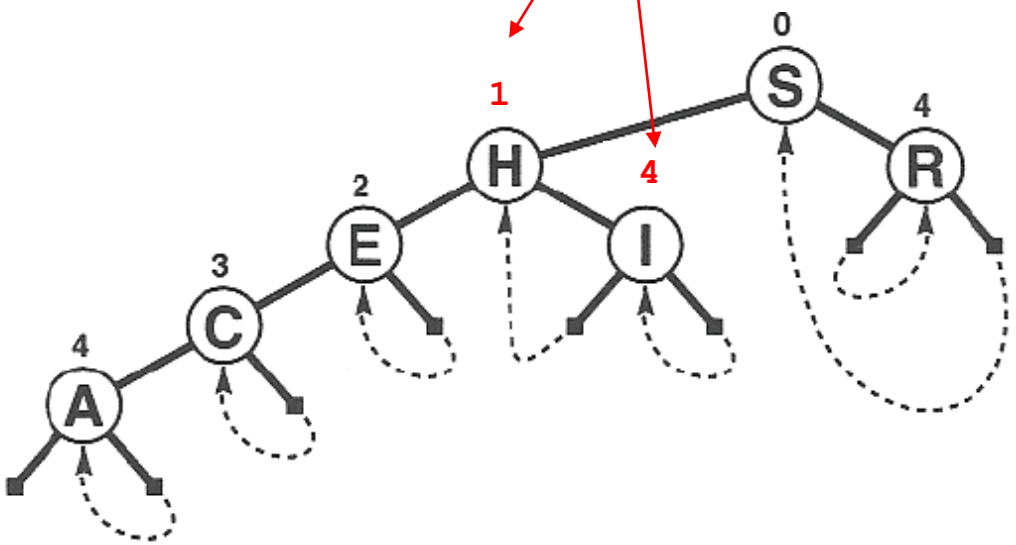
挿入2

N=01110の挿入

N=01110
H=01000

NとHとで最初に異なるのは2ビット目

Nを識別するためには，この間に節点を挿入すればよい

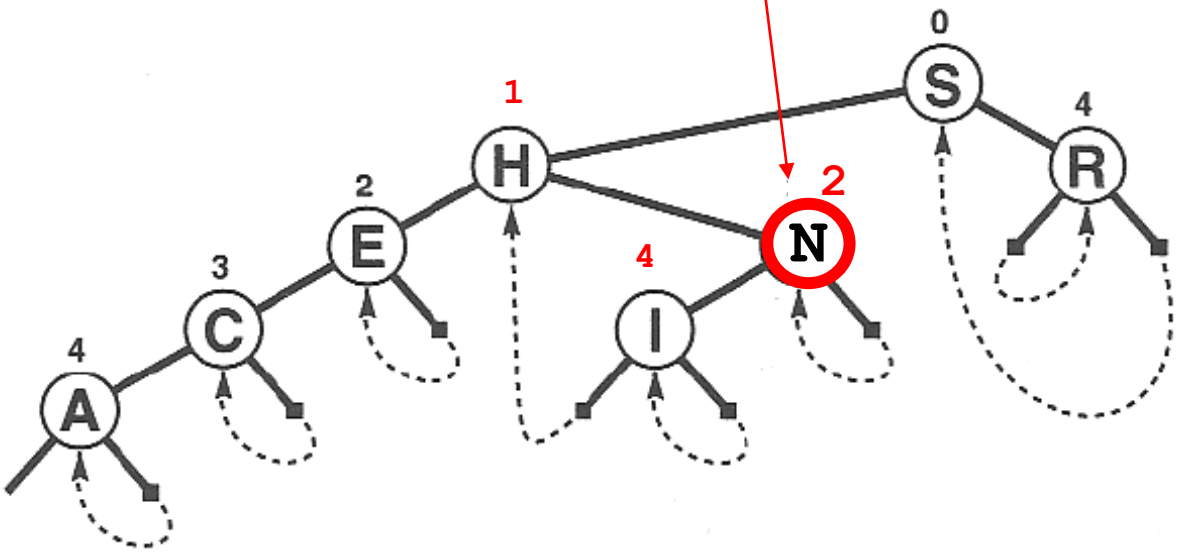


挿入2

N=01110の挿入

N=01110
H=01000

NとHとで最初に異なるのは2ビット目



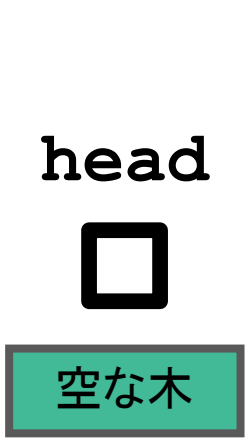
パトリシアの生成

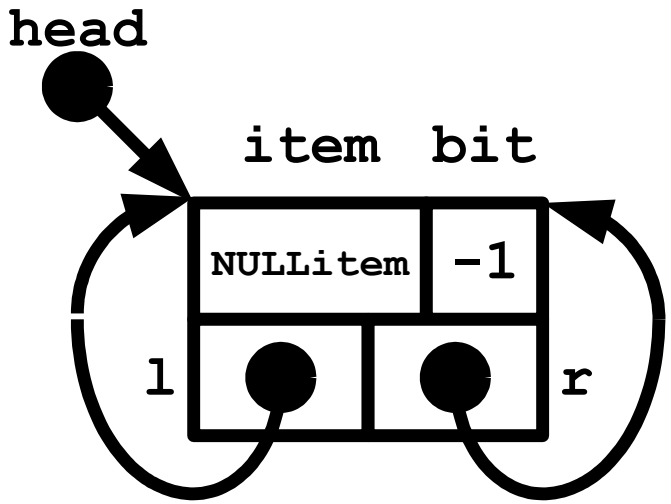
空な木

添字bit=-1

```
void STinit()
  { head = NEW(NULLitem, 0, 0, -1);
    head->l = head; head->r = head; }
```

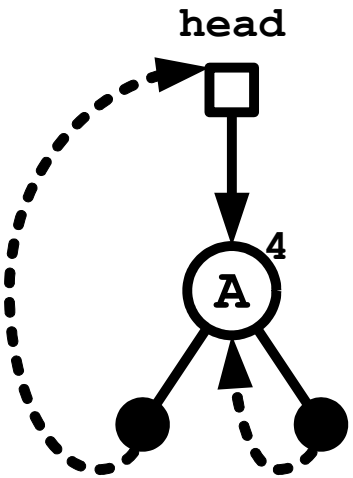
右の部分木も左の部分木も自分自身を指す

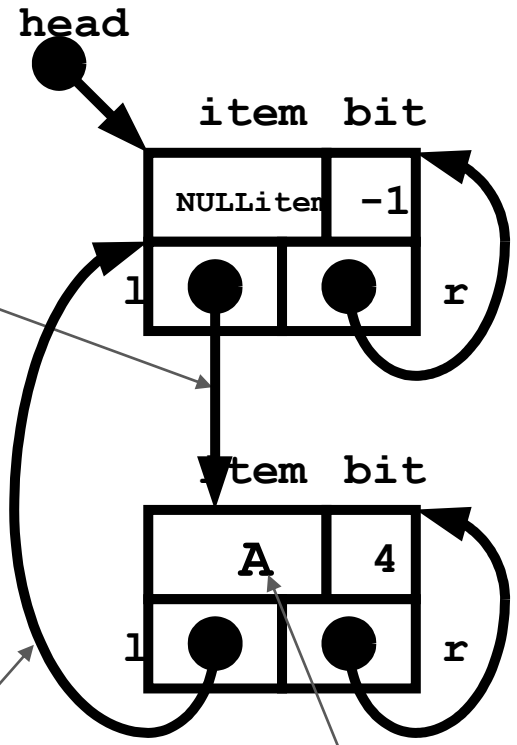




パトリシアの生成

葉が1つの木





head->lが，木の根を指す

A=00001

4ビット目で初めて1が出る

最も左の葉の左リンクはヘッドを指す

キーがAのitem

パトリシアの挿入

```
void STinsert(Item item)
  { int i;
    Key v = key(item);
    Key t = key(searchR(head->l, v, -1));
    if (v == t) return;
    for (i = 0; digit(v, i) == digit(t, i); i++) ;
    head->l = insertR(head->l, item, i, head);
  }
```

パトリシアでキーvを探索する

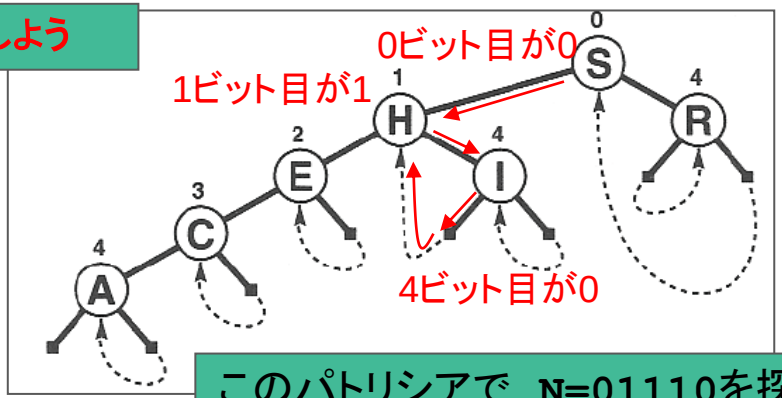
vとtとで，最初に異なるビットを発見

このパトリシアにNを挿入しよう

N=01110の挿入

N=01110
H=01000

NとHとで最初に異なるのは2ビット目
→ i = 2



このパトリシアで，N=01110を探索すると，Hが見付かる

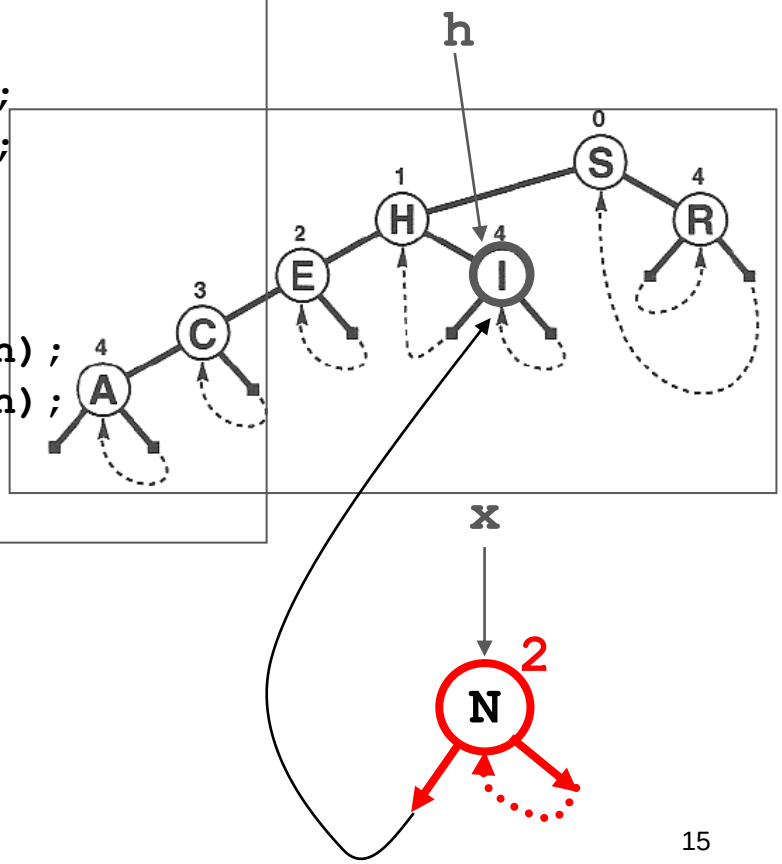
t = 'H'

パトリシアの挿入

w=2はこの再帰処理の中で不変

```
link insertR(link h, Item item, int w, link p)
  { link x; Key v = key(item);
    if ((h->bit >= w) || (h->bit <= p->bit))
      {
        x = NEW(item, 0, 0, w);
        x->l = digit(v, x->bit) ? h : x;
        x->r = digit(v, x->bit) ? x : h;
        return x;
      }
    if (digit(v, h->bit) == 0)
         h->l = insertR(h->l, item, w, h);
    else h->r = insertR(h->r, item, w, h);
    return h;
  }
```

hが'I'を指したとき，この条件が成立



N=01110の挿入

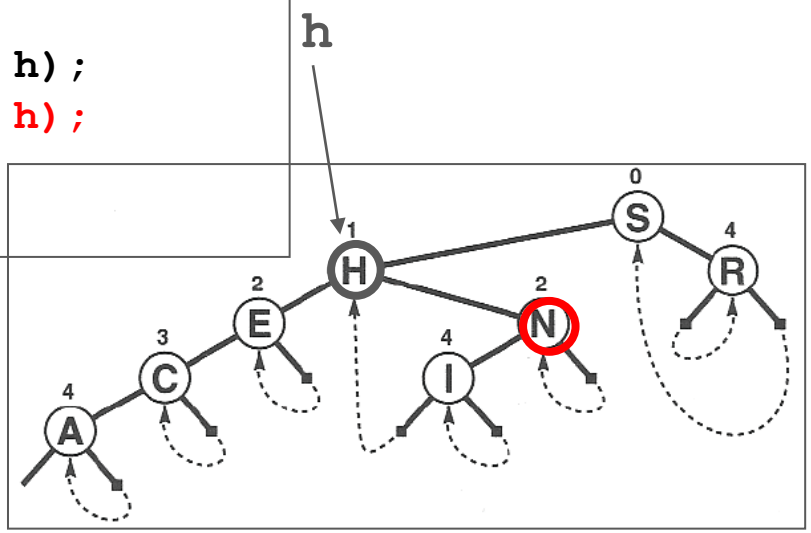
N=01110

Nの2ビット目を見ると

digit('N', 2)=1

パトリシアの挿入

```
link insertR(link h, Item item, int w, link p)
  { link x; Key v = key(item);
    if ((h->bit >= w) || (h->bit <= p->bit))
      {
        x = NEW(item, 0, 0, w);
        x->l = digit(v, x->bit) ? h : x;
        x->r = digit(v, x->bit) ? x : h;
        return x;
      }
    if (digit(v, h->bit) == 0)
         h->l = insertR(h->l, item, w, h);
    else h->r = insertR(h->r, item, w, h);
    return h;
  }
```



再帰から返って来るときに，Hの右にNが接続される

パトリシア

- 性質15.5
  - ランダムなビット列のキーN個から作られるパトリシアにおける探索や挿入は，1回の探索あたり平均約lgN回のビット比較を行う．ビット比較は，キーの長さ以下である