# 第１４週
# 木構造と探索（３）

# ハッシュ法

定義

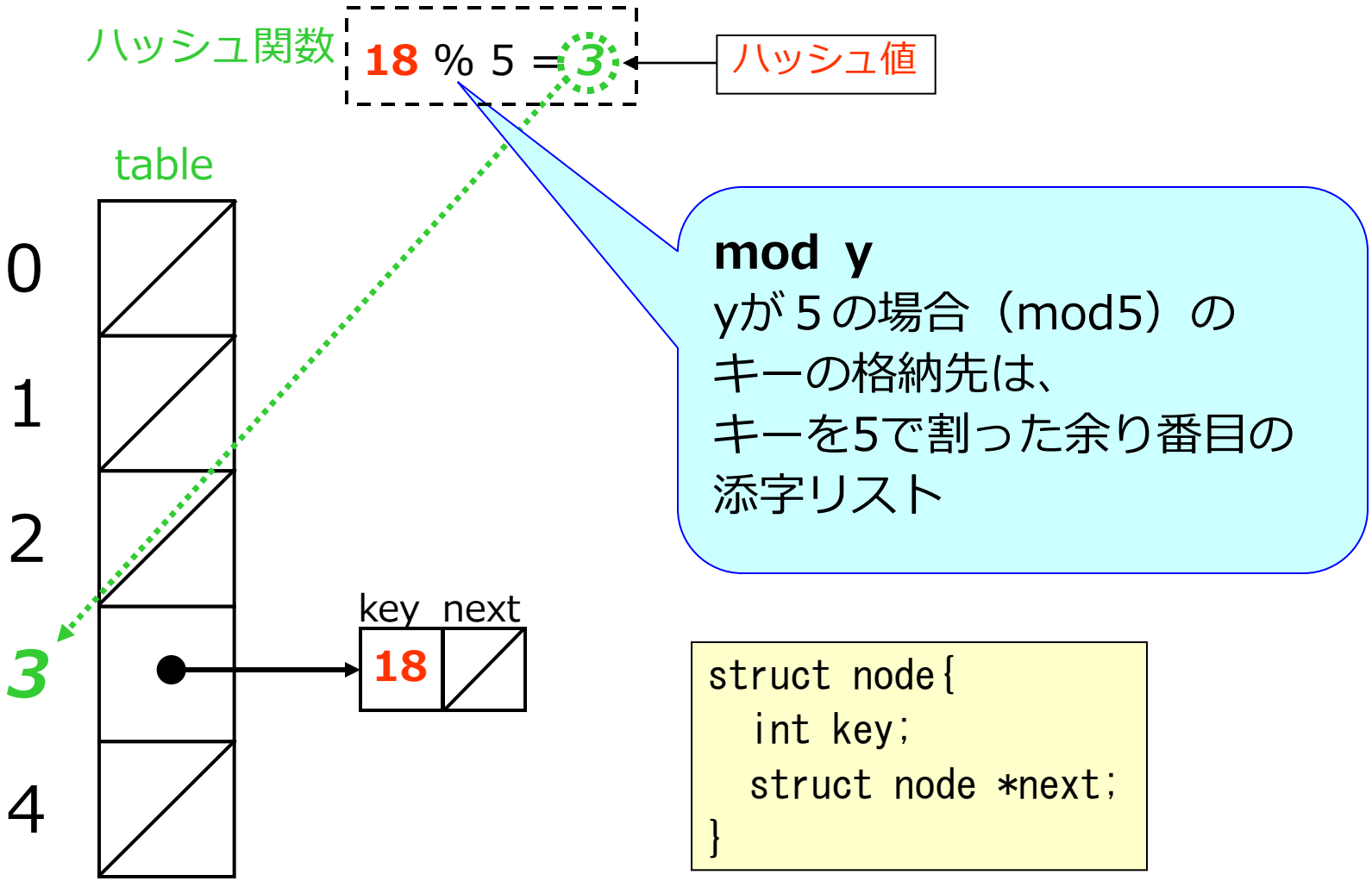
ハッシングに基づく探索法

ハッシングとは

ハッシュ関数に基づき計算されたハッシュ値によってキーの格納先を決定

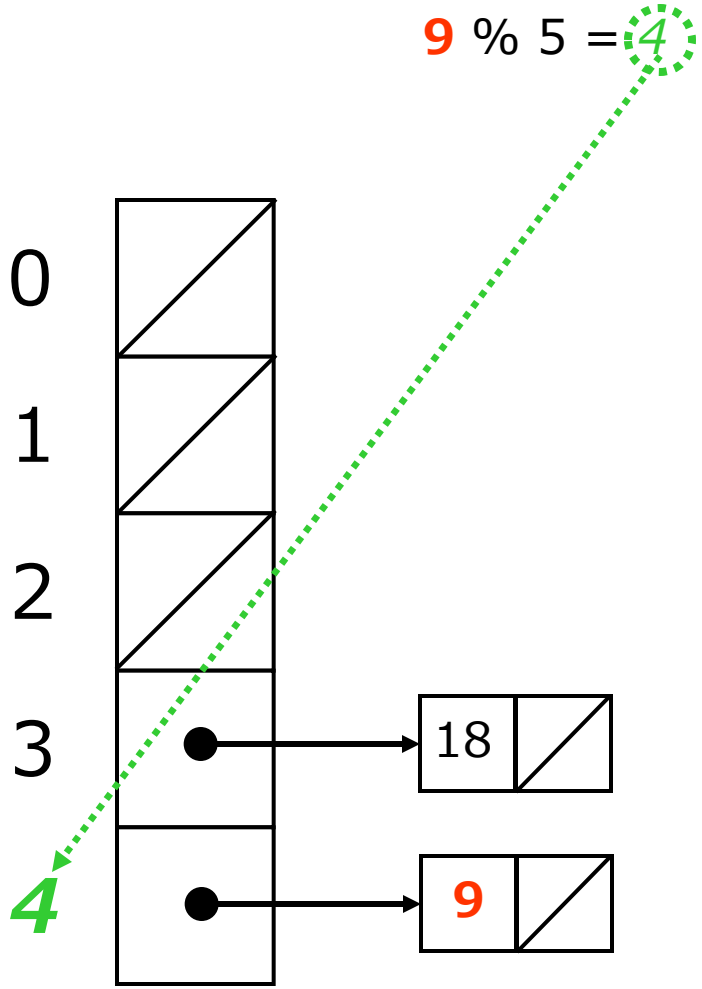
# ハッシュ関数： mod5の例 1/5

S={**18**, 9, 22, 4, 21, 12, 15, 31, 7, 25}



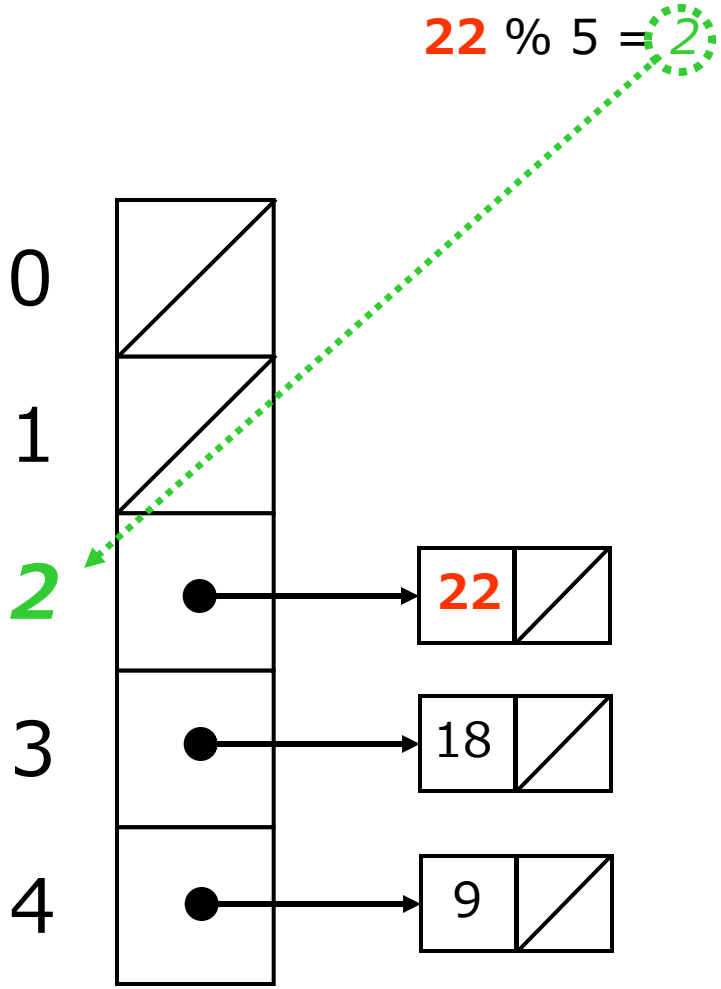
# ハッシュ関数： mod5の例 2/5

S={18, **9**, 22, 4, 21, 12, 15, 31, 7, 25}



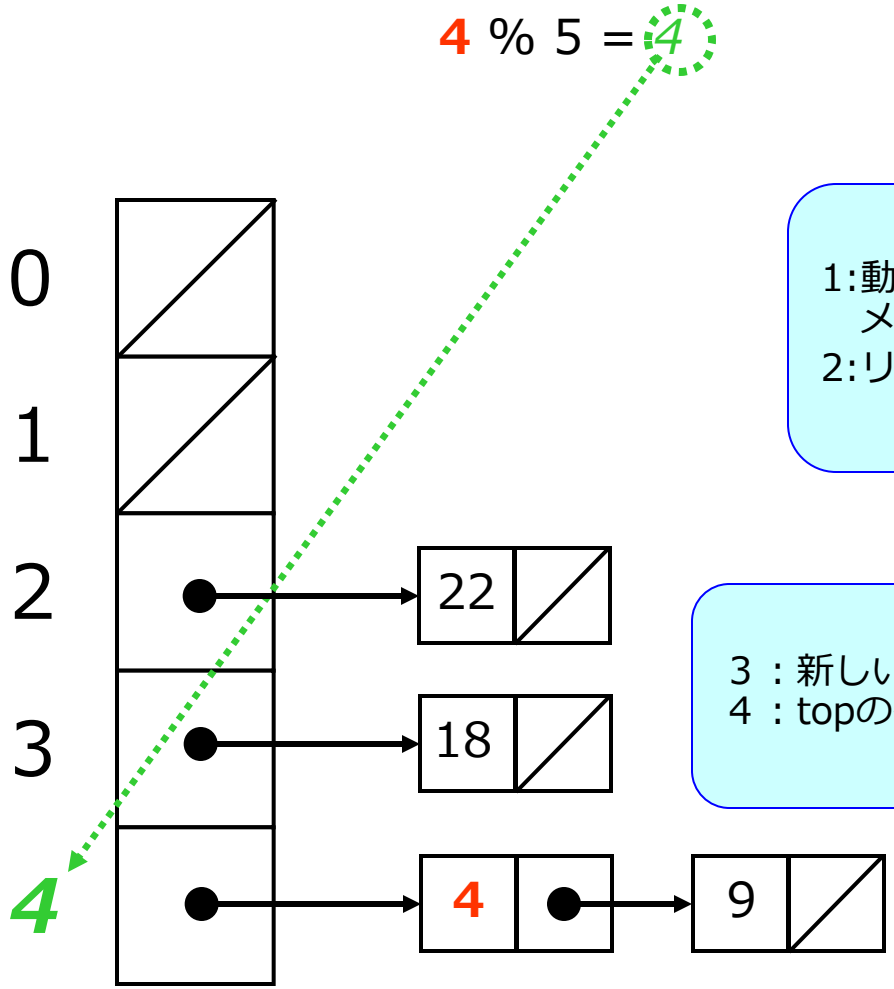
# ハッシュ関数： mod5の例 3/5

S={18, 9, **22**, 4, 21, 12, 15, 31, 7, 25}



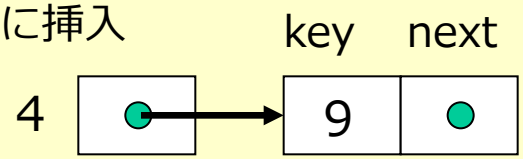
# ハッシュ関数： mod5の例 4/5

S={18, 9, 22, **4**, 21, 12, 15, 31, 7, 25}

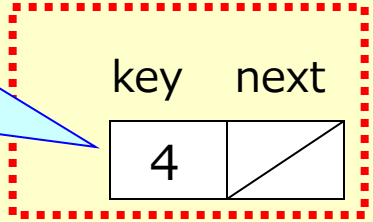



8週目の復習

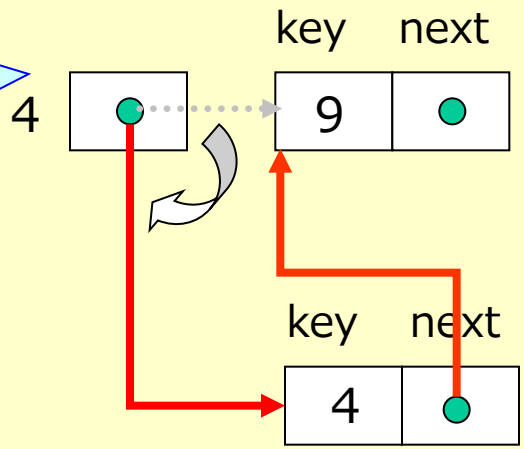
リストの先頭に挿入




1:動的メモリ確保（malloc）で
メモリを確保する
2:リストの要素の値を設定する

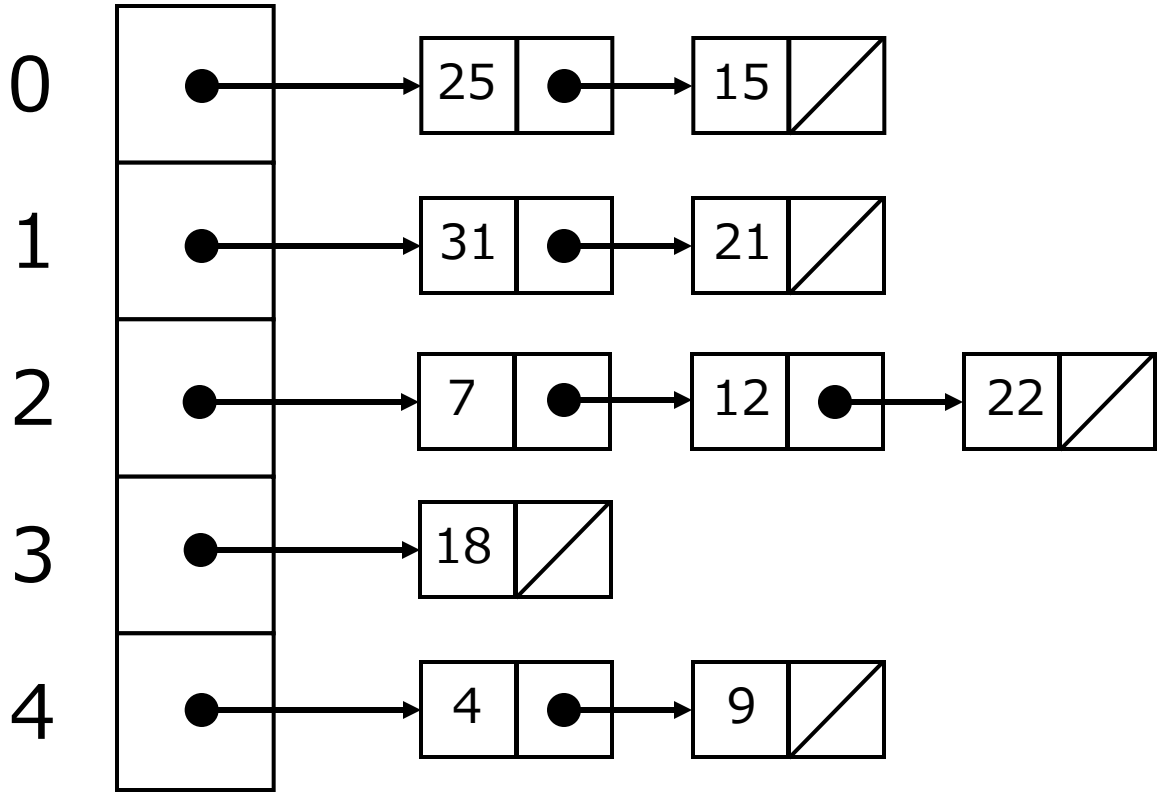



3 ：新しい要素のnextを設定
4 ：topの指す先を変更



# ハッシュ関数： mod5の例 5/5

S={18, 9, 22, 4, 21, 12, 15, 31, 7, 25}
順番に挿入していくと下記のような構造になる



# オプション課題１４－１：print_hashの出力例

print_hash
前ページのハッシュの内容を出力すると

```
0:   25 -> 15
1:   31 -> 21
2:    7 -> 12 -> 22
3:   18
4:    4 ->  9
```

# オプション課題１４－１：参考プログラム

```
int main(){
   struct node *table[5] = {NULL,NULL,NULL,NULL,NULL};
    /* 略：変数宣言*/
   // ハッシュテーブルの作成

   // ハッシュテーブルの表示
   print_hash(table);
   // ハッシュテーブルの消去
   return 0;
}
```

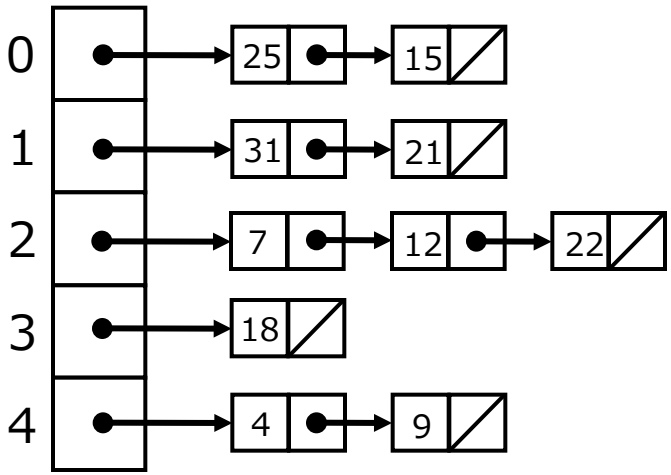
ハッシュテーブルの宣言

ハッシュを作成

教科書p179図6.48に示されるリスト集合をプログラム上で構成する
教科書は双方向リストになっていることに注意

14-1で作成するprint_hash

図6.48



# オプション課題14-２：member

struct node *member (struct node **table, int key)

取り出されたノード　　ハッシュテーブル　　検索したいkey

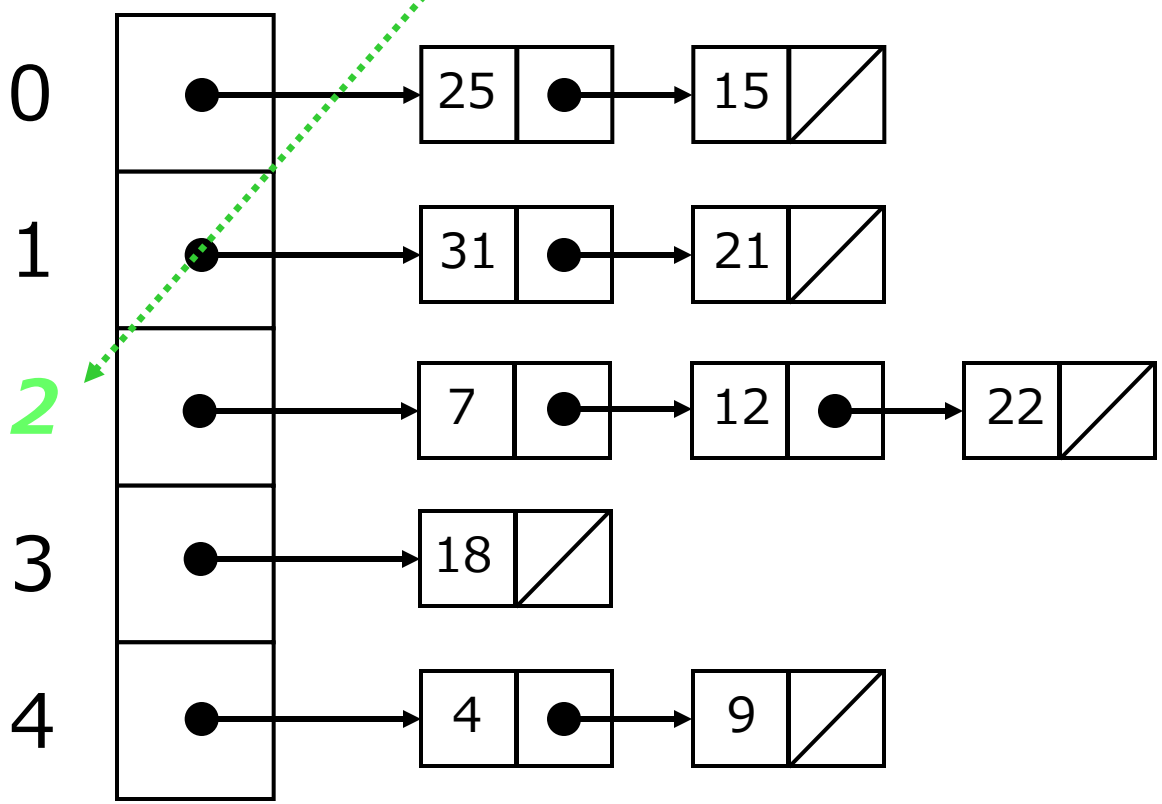
ハッシュがkeyを含むかどうかを検索する関数

ＳＴＥＰ１：
検索するリストを決める

ＳＴＥＰ２：
先頭からリストを辿りキーを探す
・リストの末尾まで見てキーがなかったら：
NULLを返す
・キーがあった場合：
そのノードを返す

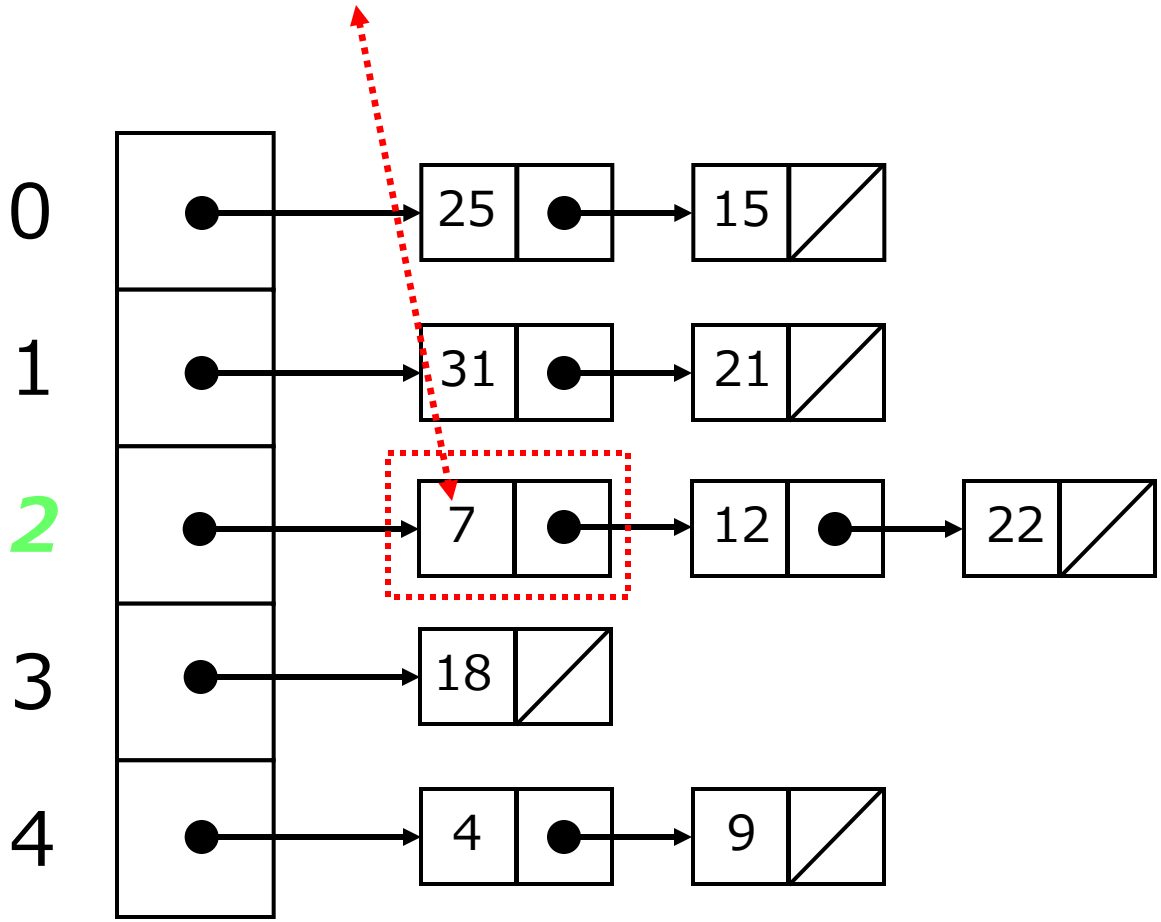
# memberの処理例１：１／３

12を検索する場合
ＳＴＥＰ１：キーを探す添字を決める
12%5 = 2



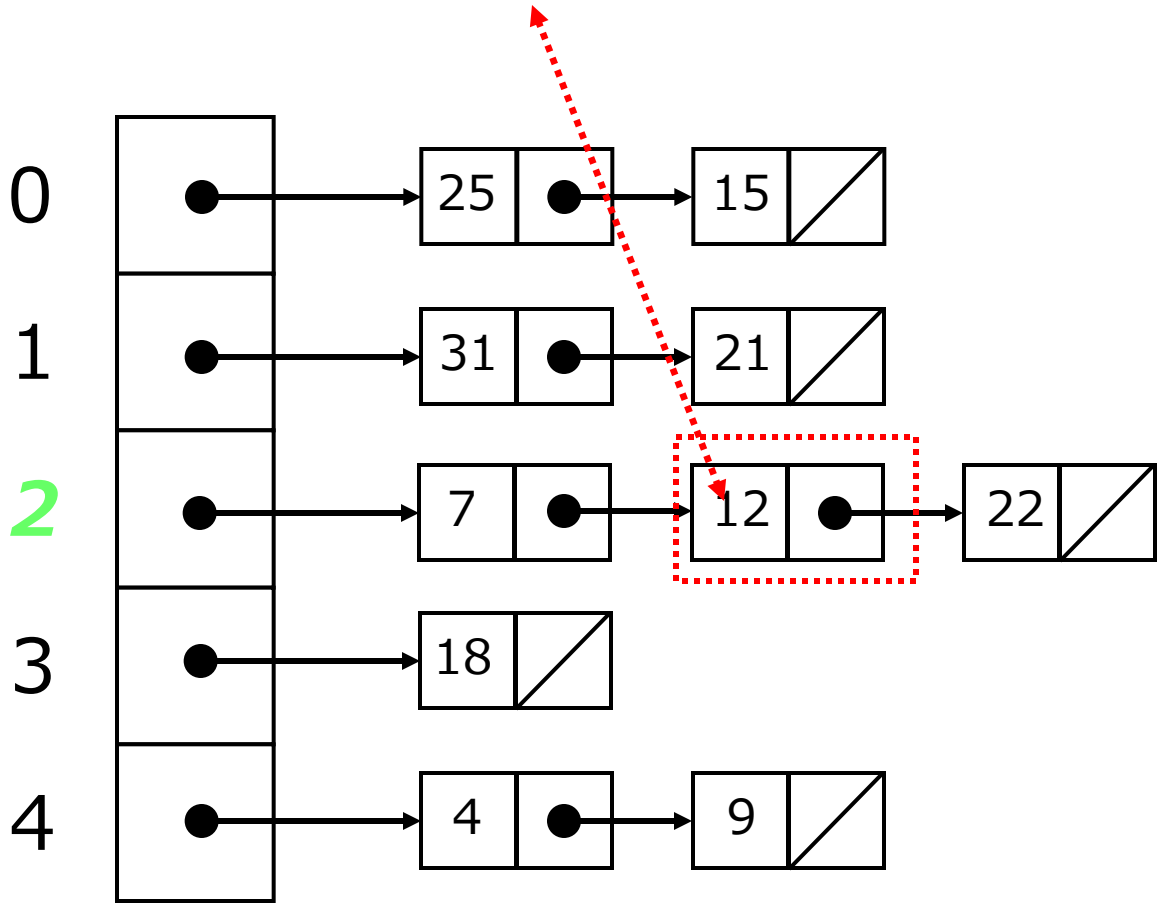
# memberの処理例１：２／３

12を検索する場合
ＳＴＥＰ２：リストを辿りキーを比較
キー（１２）が違うので、次のリストを辿る



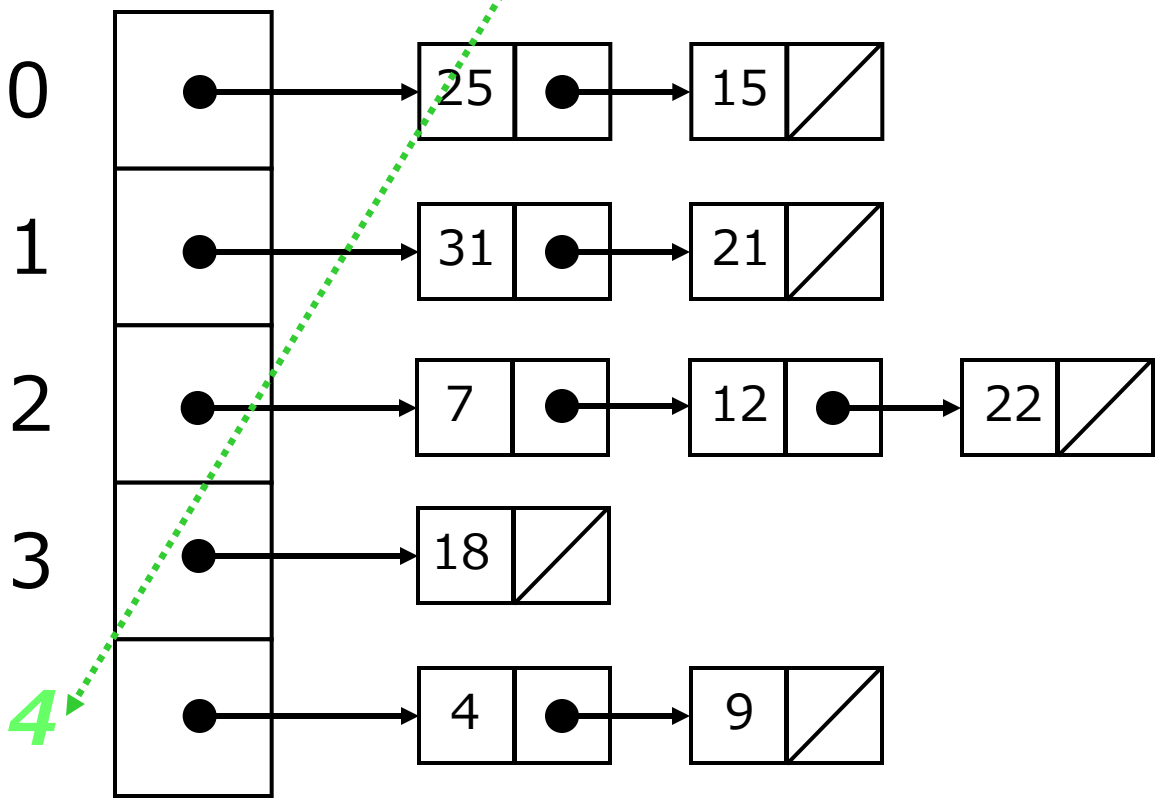
# memberの処理例１：３／３

12を検索する場合
ＳＴＥＰ２：リストを辿りkeyを比較
キー（１２）を見つけたのでノードを返す



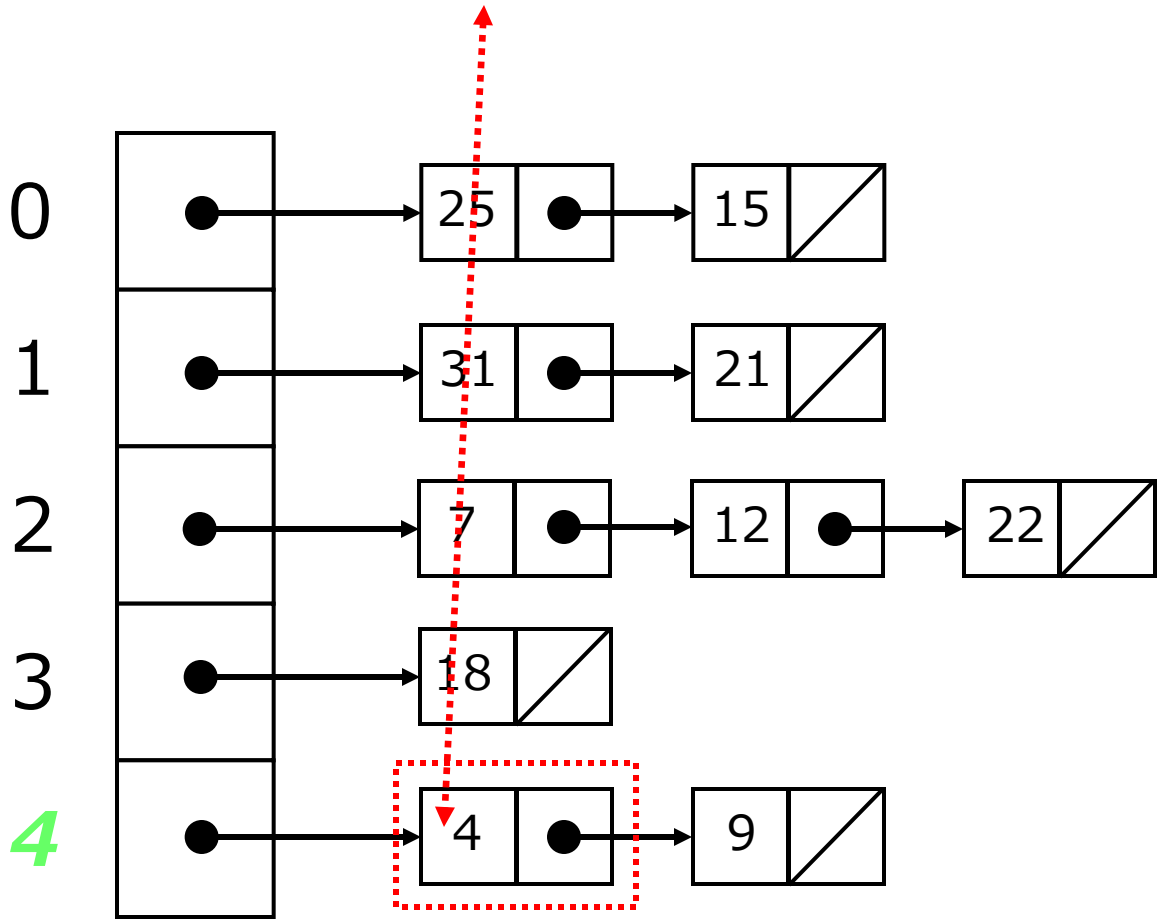
# memberの処理例２：１／３

１９を検索する場合
ＳＴＥＰ１：キーを探す添字を決める
１９%5 = 4



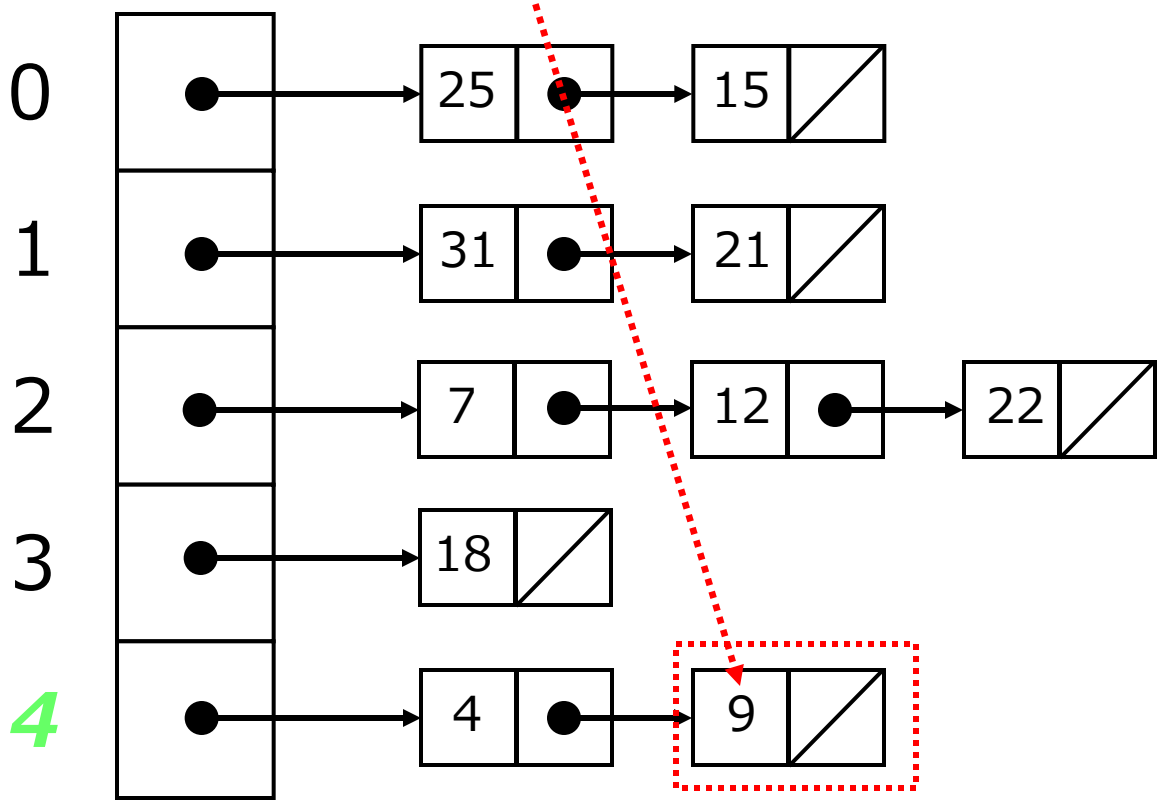
# memberの処理例２：２／３

19を検索する場合
ＳＴＥＰ２：リストを辿りキーを比較
キー（１９）が違うので、次のリストを辿る



# memberの処理例２：３／３

19を検索する場合
ＳＴＥＰ２：リストを辿りkeyを比較
キー（１９）が違い、リスト末尾なので
ＮＵＬＬを返す（見つからなかった）



# オプション課題１４−２：参考プログラム

```
int main(){
   struct node *table[5] = {NULL,NULL,NULL,NULL,NULL};
   struct node *tmp;
    /* 略：変数宣言*/
   // ハッシュの作成
```

ハッシュテーブルの宣言

```
```

ハッシュを作成

```
    // 標準入力から探索したい値を入力
    printf("探索する値を入力してください: ");
    scanf("%d",&key);
    // 探索を行う
    tmp = member(table,key);
    // 検索結果の出力
    if(tmp != NULL){
        printf("%dは見つかりました¥n",tmp->key);
    }else{
        printf("%dは見つかりませんでした¥n",key);
    }
    return 0;
}
```

関数memberを利用して、
keyがハッシュに含まれるかを検索する

# 著者リスト

1. 安積 卓也（情報システム学科）
2. 泉　朋子　（情報コミュニケーション学科）