Cấu trúc dữ liệu - CT177

Chương 5

Tự điển

Bộ môn Công Nghệ Phần Mềm

NỘI DUNG

- Khái niệm
- Cài đặt tự điển bằng mảng
- Cài đặt tự điển bằng bảng băm

TỰ ĐIỂN

- Tự điển là một kiểu dữ liệu trừu tượng tập hợp đặc biệt với các phép toán thêm (INSERT), bớt (DELETE) và tìm kiếm (MEMBER) có phần hiệu quả nhất.

TỰ ĐIỂN

- Một số ứng dụng của tự điển
  - Công cụ kiểm tra chính tả
  - Từ điển dữ liệu trong các ứng dụng quản lý cơ sở dữ liệu
  - Bảng ký hiệu được tạo bởi trình biên dịch
  - Bảng định tuyến trong các thành phần mạng (tra cứu hệ thống phân giải tên miềm - DNS lookup)

Key[Website]	Value [IP Address]
www.CareerMonks.com	128.112.136.11
www.AuthorsInn.com	128.112.128.15
www.AuthInn.com	130.132.143.21
www.klm.com	128.103.060.55
www.CareerMonk.com	209.052.165.60

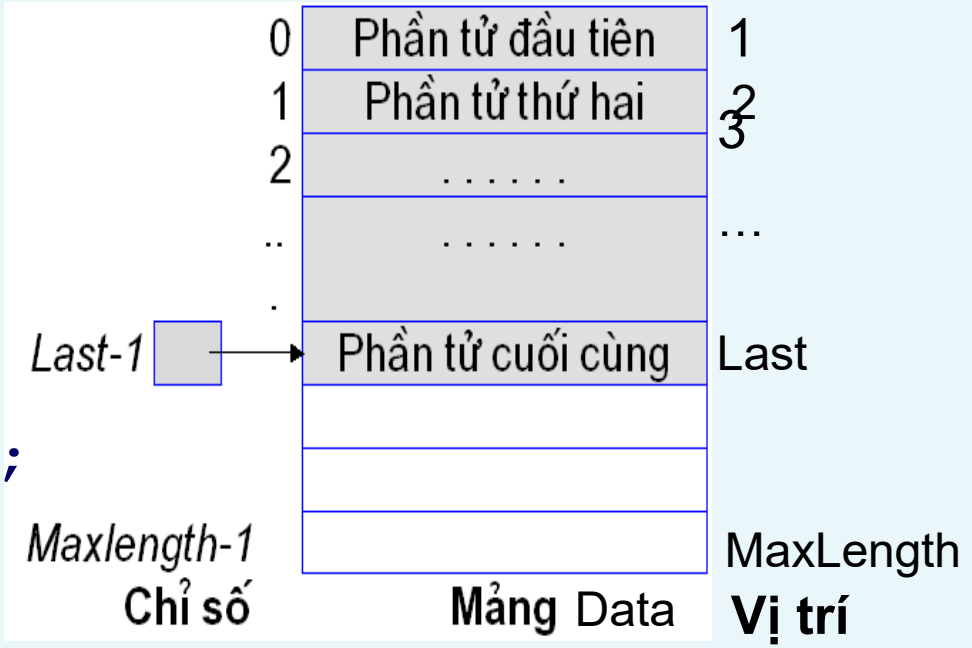
TỰ ĐIỂN

- Tự điển có thể được cài đặt bằng:
  - Danh sách đặc (mảng).
  - Bảng băm.

Implementation	Search	Insert	Delete
Unordered Array	n	n	n
Ordered Array (can be implemented with array binary search)	$logn$	n	n
Unordered List	n	n	n
Ordered List	n	n	n
Binary Search Trees (O($logn$) on average)	$logn$	$logn$	$logn$
Balanced Binary Search Trees (O($logn$) in worst case)	$logn$	$logn$	$logn$
Ternary Search (only change is in logarithms base)	$logn$	$logn$	$logn$
Hashing (O(1) on average)	1	1	1

Data Structures and Algorithms Made Easy Data Structures and Algorithmic Puzzles by Narasimha Karumanchi

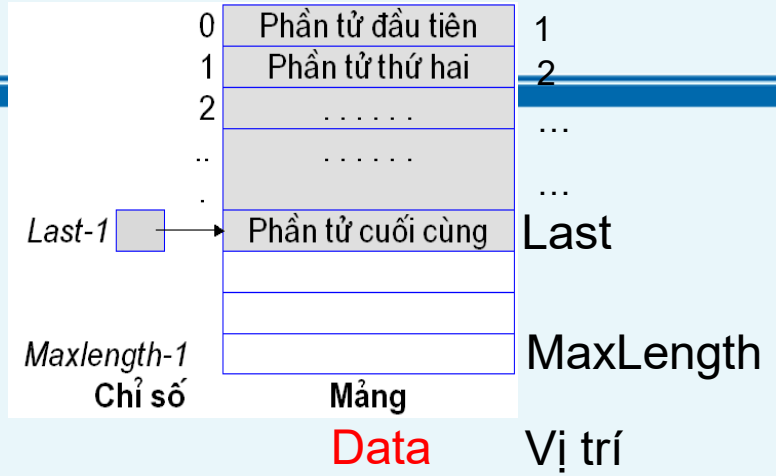
CÀI ĐẶT TỰ ĐIỂN BẰNG MẢNG – Danh sách đặc



- Khai báo

```
#define MaxLength ...
typedef ... ElementType;
typedef int Position;
typedef struct {
  ElementType Data[MaxLength];
  Position    Last;
} SET;
```

CÀI ĐẶT TỰ ĐIỂN BẰNG MẢNG



- Khởi tạo rỗng

```
void makenullSet(SET *pS){
   (*pS).Last=0;
}
```

pS->Last=0

- Kiểm tra 1 phần tử có trong tự điển không

```
int isMember(ElementType x, SET S){
   Position P=1, Found=0;
   while ((P <= S.Last) && (Found == 0))
      if ((S.Data[P-1]) == x) Found = 1;
      else P++;
   return Found;
}
```

!Found

CÀI ĐẶT TỰ ĐIỂN BẰNG MẢNG

- Thêm 1 phần tử vào tự điển

```
void insertSet(ElementType x, SET *pS){
  if (fullSet(*pS))   pS->Last==MaxLength
    printf("Tap hop day");   !isMember(x,*pS)
  else if (isMember(x,*pS)==0)     {
        (*pS).Last++;   pS->Last++
        (*pS).Data[(*pS).Last-1]=x;
                pS->Data[pS->Last-1]
   }
  else
    printf("\nPt da ton tai trong t.dien");
}
```

CÀI ĐẶT TỰ ĐIỂN BẰNG MẢNG

- Xóa 1 phần tử khỏi tự điển

```
void deleteSet(ElementType x, SET *pS){
  if (emptySet(*pS))   pS->Last==0
     printf("Tap hop rong!");
  else{
     Position Q=1;        pS->Last     pS->Data
     while((Q<=(*pS).Last)&&((*pS).Data[Q-1]!=x))
        Q++;
     if ((*pS).Data[Q-1]==x){
        (*pS).Data[Q-1]=(*pS).Data[(*pS).Last-1];
        (*pS).Last--;
     }//if
}
```

CÀI ĐẶT TỰ ĐIỂN BẰNG BẢNG BĂM

- Băm (hashing) là một kỹ thuật rất quan trọng và được dùng rộng rãi để cài đặt tự điển.
- Trong tự điển có n phần tử, cài đặt tự điển bằng bảng băm đòi hỏi trung bình chỉ một hằng thời gian cho mỗi phép toán thêm và tìm kiếm trong khi cài đặt tự điển bằng mảng đòi hỏi tốn n bước cho mỗi phép toán trên.
- Các dạng bảng băm:
  - Băm đóng
  - Băm mở

CÀI ĐẶT TỰ ĐIỂN BẰNG BẢNG BĂM

- Hàm băm là một ánh xạ từ tập dữ liệu A đến các số nguyên 0..B-1. Hàm băm được sử dụng để tìm giá trị băm.
- Các phương pháp xác định hàm băm
  - Phương pháp chia

x	H(x)= x mod B với B=10
34/84	4
19	9

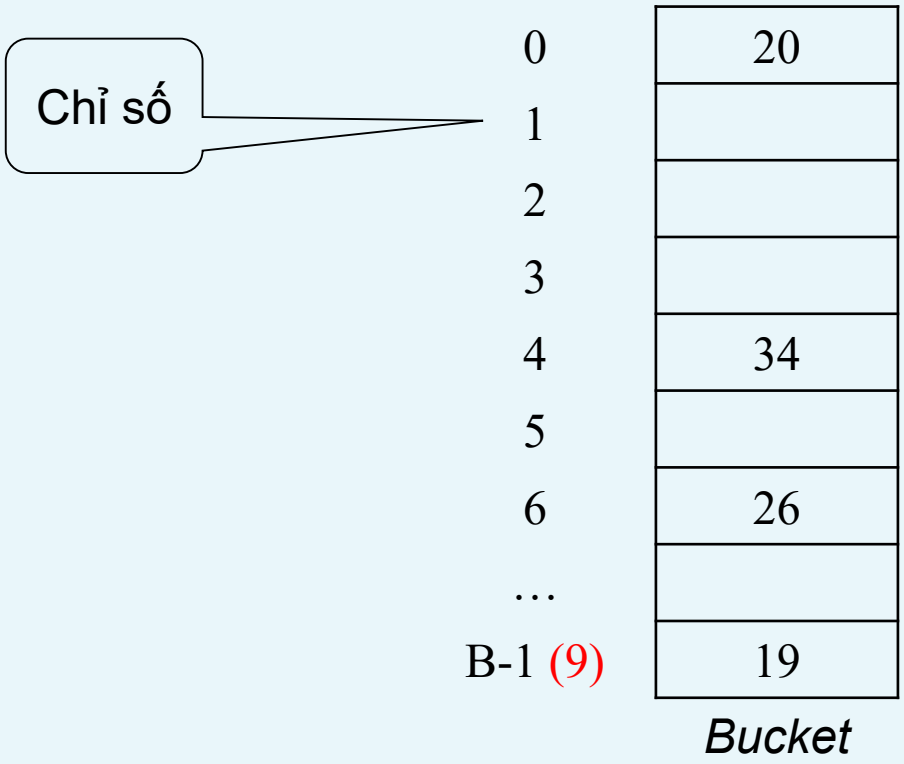
  - Phương pháp nhân

x	x^2	H(x) gồm 3 số ở giữa
5402	29181604	181 hoặc 816
0367	00134689	134 hoặc 346

  - Phương pháp tách
    - Tách khóa. VD 17046329 có giá trị băm (329+046+017)%1000 = 392
    - Gấp khóa. VD 17046329 có giá trị băm (923+046+710)%1000 = 679

BĂM ĐÓNG

- Bảng băm đóng lưu giữ các phần tử của tự điển ngay trong mảng.
- Bucket thứ i chứa phần tử có giá trị băm là i.
- Nếu có nhiều phần tử có cùng giá trị băm, **chiến lược băm lại** (rehash strategy) được sử dụng để giải quyết sự đụng độ.



Ví dụ:

Hàm băm:

H(x)=x%B

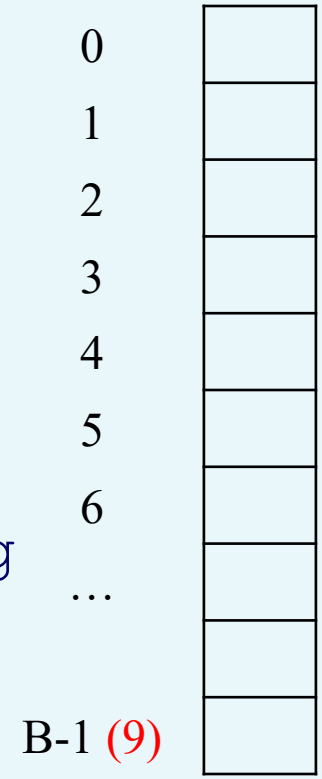
Hàm băm lại tuyến tính

$H_i(x)=(H(x)+i) \bmod B$

BĂM ĐÓNG

- Khai báo

```
#define B 100
#define Deleted -1000
 //Gia dinh gia tri cho o da bi xoa
#define Empty 1000
//Gia dinh gia tri cho o chua su dung
typedef int ElementType;
typedef ElementType Dictionary[B];
```



BĂM ĐÓNG

- Tạo tự điển rỗng

```
void makenullDic(Dictionary D){
 for (int i=0 ;i<B; i++)
      D[i]=Empty;
}
```

0	E
1	E
2	E
3	E
4	E
5	E
6	E
…	E
	E
B-1 (9)	E

E: Empty

BĂM ĐÓNG

- Tạo tự điển rỗng

```
void makenullDic(Dictionary D){
 for (int i=0 ;i<B; i++)
      D[i]=Empty;
}
```

- Hàm băm

```
int H(ElementType x){
 return x%B;
}
```

0	E
1	E
2	E
3	E
4	E
5	E
6	E
…	E
	E
B-1 (9)	E

E: Empty

BĂM ĐÓNG

- Ví dụ: Cho bảng băm với B=10 như hình dưới đây, kiểm tra xem giá trị 29, 39 có trong bảng băm?

Chỉ số

0	20
1	29
2	49
3	E
4	E
5	E
6	26
7	E
8	E
B-1 (9)	19

Ví dụ

Hàm băm

H(x)=x%B

H(29)=29%10=9

Hàm băm lại tuyến tính

$H_i(x)=(H(x)+i)$ mod B

i=0➔ $H_0(29)=(9+0)\%10=9$

i=1➔ $H_1(29)=(9+1)\%10=0$

i=2➔ $H_2(29)=(9+2)\%10=1$

BĂM ĐÓNG

- Kiểm tra sự tồn tại của phần tử trong tự điển

```
int isMember(ElementType x, Dictionary D){
        int i=0,init=H(x);
        while((i<B)&&(D[(i+init)%B]!=Empty)
                        &&(D[(i+init)%B]!=x))
                i++;
        return (D[(i+init)%B]==x);
}
```

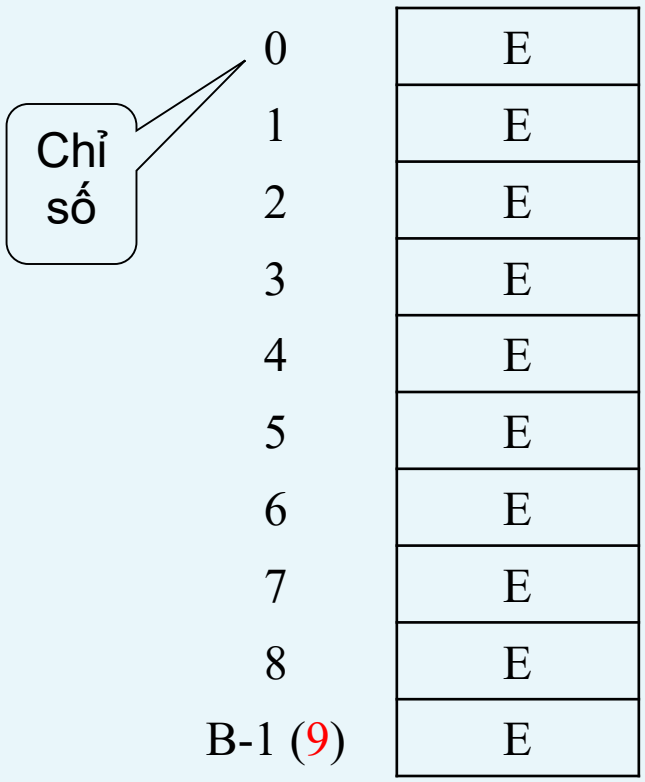
Ví dụ

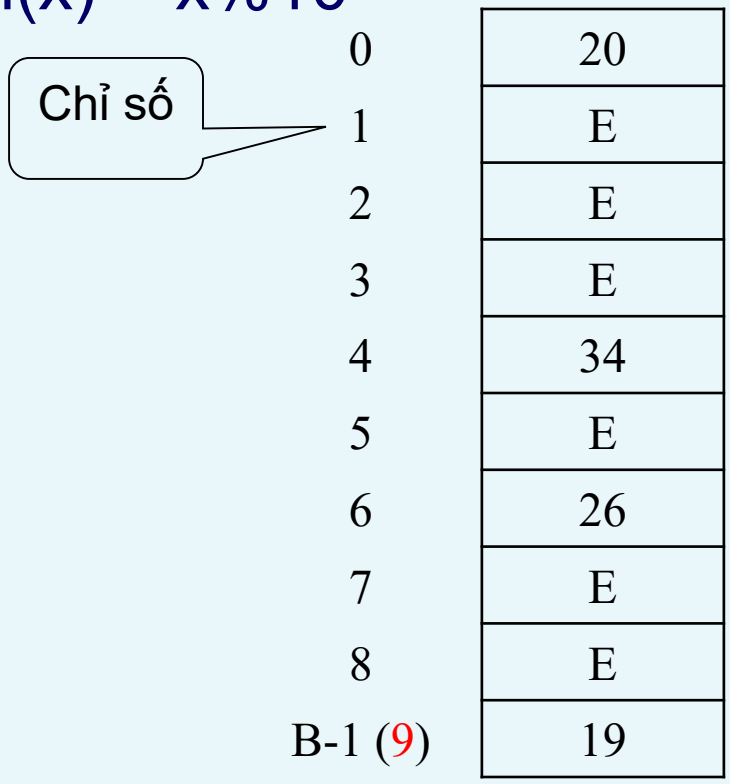
Hàm băm: **H(x)=x%B**

Hàm băm lại tuyến tính: **$H_i(x)=(H(x)+i) \bmod B$**

BĂM ĐÓNG

- Ví dụ: Ta cần lưu trữ các số nguyên 34, 20, 26 và 19 vào trong bảng băm có số bucket B = 10 và sử dụng hàm băm h(x) = x%10





BĂM ĐÓNG

- Ví dụ: Thêm giá trị 29 vào bảng băm có B=10 và sử dụng hàm băm lại $H_i(x)=(H(x)+i) \bmod B$ để giải quyết trường hợp đụng độ.

0	20
1	E
2	E
3	E
4	34
5	E
6	26
7	E
8	E
B-1 (9)	19

0	20
1	29
2	E
3	E
4	34
5	E
6	26
7	E
8	E
B-1 (9)	19

Ví dụ

Hàm băm

H(x)=x%B

H(29)=29%10=9

Hàm băm lại tuyến tính

$H_i(x)=(H(x)+i) \bmod B$

i=0➔ $H_0(29)=(9+0)\%10=9$

i=1➔ $H_1(29)=(9+1)\%10=0$

i=2➔ $H_2(29)=(9+2)\%10=1$

BĂM ĐÓNG

- Ví dụ: Thêm vào giá trị 30

0	20
1	29
2	12
3	D/E
4	34
5	E
6	26
7	E
8	E
B-1 (9)	19

0	20
1	29
2	12
3	30
4	34
5	E
6	26
7	E
8	E
B-1 (9)	19

Ví dụ

Hàm băm

H(x)=x%B

H(30)=30%10=0

Hàm băm lại tuyến tính

$H_i(x)=(H(x)+i) \bmod B$

i=0➔ $H_0(30)=(0+0)\%10=0$

i=1➔ $H_1(30)=(0+1)\%10=1$

i=2➔ $H_2(30)=(0+2)\%10=2$

i=3➔ $H_3(30)=(0+3)\%10=3$

BĂM ĐÓNG

- Thêm phần tử vào tự điển

```
void insertDic(ElementType x, Dictionary D){
     int i=0,init;
     if (fullDic(D))
          printf("Bang bam day");
     else if (isMember(x,D)==0){
          init=H(x);
          while((i<B)&&(D[(i+init)%B]!=Empty)
                       &&(D[(i+init)%B]!=Deleted))
               i++;
          D[(i+init)%B]=x;
     }
     else
          printf("\nPhan tu da ton tai");
}
```

BĂM ĐÓNG

- Kiểm tra tự điển có đầy

```
void fullDic(Dictionary D){
  int i=0;
  int full=1;
  while ((i<B) && (full == 1))
     if((D[i] == Deleted) || (D[i] == Empty))
        full = 0;
     else
        i++;
  return full;
}
```

BĂM ĐÓNG

- Bài tập: Giả sử bảng băm có 7 bucket, hàm băm là h(x)= x mod 7. Hãy vẽ hình biểu diễn bảng băm khi ta lần lượt đưa vào bảng băm rỗng các khoá 1,8, 27, 64, 125, 216, 343 trong trường hợp dùng bảng băm đóng với chiến lược giải quyết đụng độ là phép thử tuyến tính?

0	E
1	E
2	E
3	E
4	E
5	E
6	E

BĂM ĐÓNG

- Ví dụ: Xóa giá trị 30

0	20
1	29
2	12
3	30
4	34
5	E
6	26
7	E
8	E
9 (B-1)	19

0	20
1	29
2	12
3	Deleted
4	34
5	E
6	26
7	E
8	E
9 (B-1)	19

Hàm băm

H(x)=x%B

H(30)=30%10=0

Hàm băm lại tuyến tính

$H_i(x)=(H(x)+i) \bmod B$

i=0➔ $H_0(30)=(0+0)\%10=0$

i=1➔ $H_1(30)=(0+1)\%10=1$

i=2➔ $H_2(30)=(0+2)\%10=2$

i=3➔ $H_3(30)=(0+3)\%10=3$

BĂM ĐÓNG

- Xóa phần tử ra khỏi tự điển

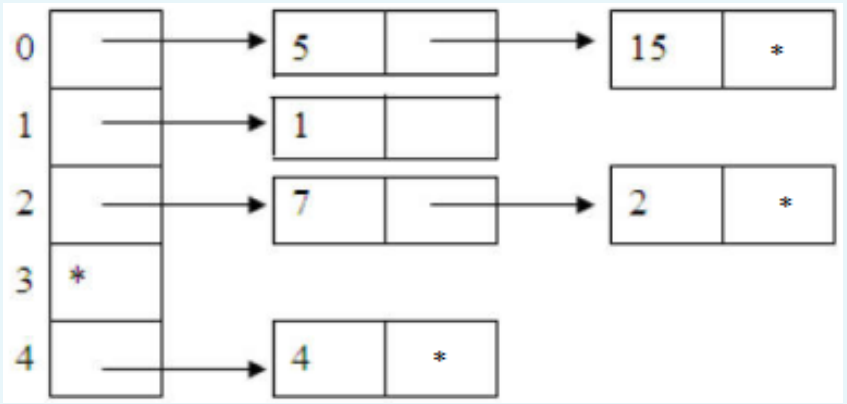
```
void deleteDic(ElementType x, Dictionary D) {
    if (emptyDic(D))
        printf("\nBang bam rong!");
    else {
        int i=0,init=H(x);
        while ((i<B)&&(D[(i+init)%B]!=x)
                    &&(D[(i+init)%B]!=Empty))
            i++;
        if (D[(i+init)%B]==x)
            D[(i+init)%B]=Deleted;
    }
}
```

BĂM ĐÓNG

- Kiểm tra tự điển có rỗng

```
void emptyDic(Dictionary D){
  int i=0;
  int empty=1;
  while ((i<B) && (empty == 1))
     if((D[i] != Deleted) && (D[i] != Empty))
        empty = 0;
     else
        i++;
  return empty;
}
```

BĂM MỞ



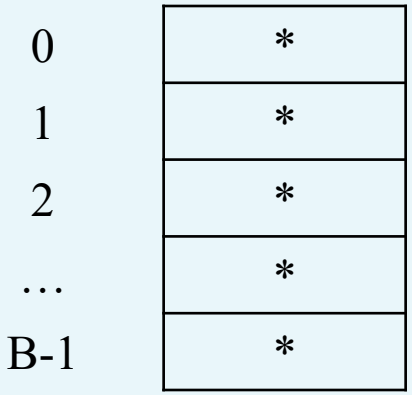
- Khai báo

```
#define B ...
typedef ... ElementType;
struct Node {
  ElementType  Data;
  struct Node* Next;
};
typedef struct Node* Position;
typedef Position Dictionary[B];
```

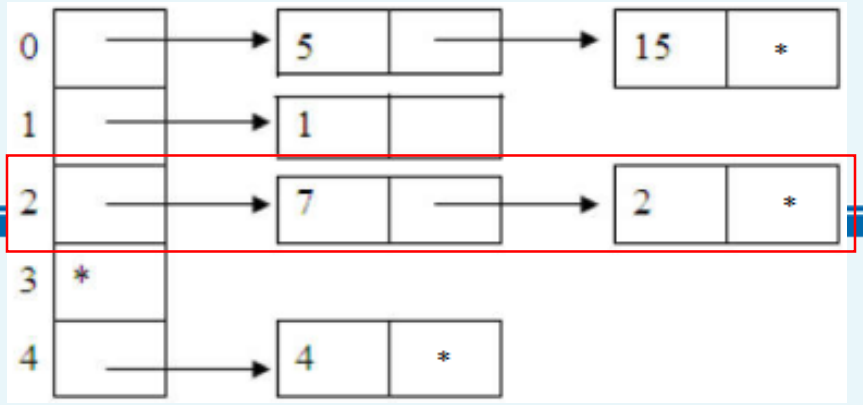
BĂM MỞ

- Tạo tự điển rỗng

```
void makenullDic(Dictionary *D){
  for(int i=0;i<B;i++)
     (*D)[i]=NULL;
}
```



BĂM MỞ



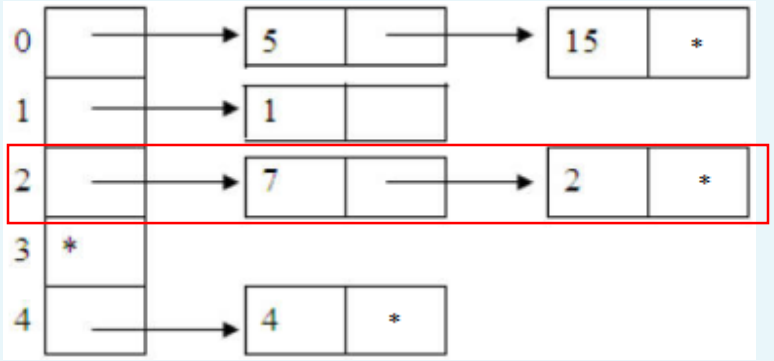
- Kiểm tra sự tồn tại của một phần tử trong tự điển

```
int isMember(ElementType x, Dictionary D){
    Position P;
    int Found=0;
    P=D[H(x)]; //Tim o muc H(x)
    //Duyet tren ds thu H(x)
    while((P!=NULL) && (!Found))
        if (P->Data==x) Found=1;
        else P=P->Next;
    return Found;
}
```

BĂM MỞ

- Thêm phần tử vào tự điển

```
void insertDic(ElementType x,Dictionary *D){
    int Bucket;
    Position P;
    if (!isMember(x,*D)){
     Bucket=H(x);
     P=(*D)[Bucket];
     //Cap phat o nho moi cho (*D)[Bucket]
     (*D)[Bucket]=(struct Node*)malloc(sizeof(struct Node));
     (*D)[Bucket]->Data=x;
     (*D)[Bucket]->Next=P;
    }
}
```



BĂM MỞ

- Bài tập: Giả sử bảng băm có 7 bucket, hàm băm là h(x)= x mod 7. Hãy vẽ hình biểu diễn bảng băm khi ta lần lượt đưa vào bảng băm rỗng các khoá 1,8, 27, 64, 125, 216, 343 trong trường hợp dùng bảng băm mở?

0	*
1	*
2	*
3	*
4	*
5	*
6	*

Xoá phần tử ra khỏi tự điển

```
void deleteDic(ElementType x,Dictionary *D){
    int Bucket, Done;
    Position P,Q;
    Bucket=H(x);
    if ((*D)[Bucket]!=NULL) { //danh sach ton tai
       if ((*D)[Bucket]->Data==x) { //x dau dsach
          Q=(*D)[Bucket];
          (*D)[Bucket]=(*D)[Bucket]->Next;
          free(Q);
        }
       else { // Tim x
          Done=0;
          P=(*D)[Bucket];
          while ((P->Next!=NULL) && (!Done))
             if (P->Next->Data==x) Done=1; else P=P->Next;
          if (Done) {      // Neu tim thay
             Q=P->Next; //Xoa P->Next
             P->Next=Q->Next;
             free(Q);
}}}}
```

BĂM MỞ

HẾT