CÁC KIỂU DỮ LIỆU TRỪU TƯỢNG CƠ BẢN

TẬP HỢP

Đỗ Thanh Nghị

dtnghi@cit.ctu.edu.vn

NỘI DUNG

- Khái niệm tập hợp
- Phép toán trên tập hợp
- Cài đặt tập hợp
- Từ điển
- Bảng băm

KHÁI NIỆM TẬP HỢP

- Là tập hợp các thành viên hoặc phần tử
- Các phần tử của tập hợp phải khác nhau
- Các phần tử của tập hợp có quan hệ tuyến tính
- Liệt kê các phần tử trong cặp dấu ngoặc {}

$x \in S$: x là một thành viên của tập hợp S

$x \notin S$: x không là một thành viên của tập hợp S

$\varnothing$: tập hợp rỗng, không có thành viên

VD: A={1,2} B= {1,2,3}

BIỂU DIỄN TẬP HỢP

- Cho hai tập hợp A và B:
  - A là 1 bộ phận của B, kí hiệu $A \subseteq B$: nếu mọi thành viên của A đều là thành viên của B
    - VD: $A \subseteq B$
  - Tập hợp A và B bằng nhau, kí hiệu $A = B$: nếu $A \subseteq B$ và $B \subseteq A$
  - Hợp của hai tập hợp: $A \cup B = \{x \mid x \subseteq A \text{ hoặc } x \in B\}$
  - Giao của hai tập hợp: $A \cap B = \{x \mid x \in A \text{ và } x \in B\}$
  - Hiệu của hai tập hợp: $A \setminus B = \{x \mid x \in A \text{ và } x \notin B\}$

PHÉP TOÁN TẬP HỢP

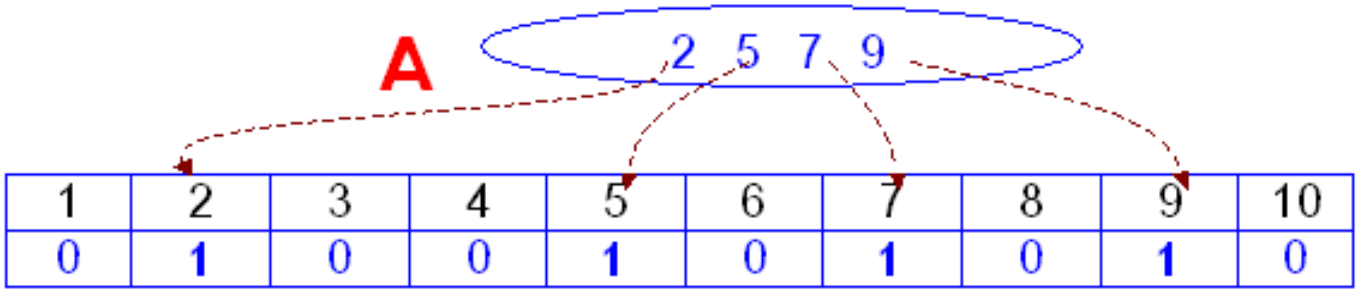
Tên hàm/thủ tục	Diễn giải
MAKENULLSET(*A*)	Tạo tập A rỗng
EMPTY(*A*)	Kiểm tra xem tập A có rỗng?
MEMBER(*x, A*)	Kiểm tra xem x có thuộc A?
INSERTSET(x, *A*)	Thêm x vào tập A
DELETESET(*x, A*)	Xóa x khỏi tập A
ASSIGN(*A, B*)	Gán B=A
MIN(*A*)	Trả về phần tử nhỏ nhất trong tập hợp
EQUAL(*A,B*)	Trả về TRUE nếu A=B
UNION(*A,B,C*)	C=A∪B
INTERSECTION(*A,B,C*)	C=A∩B
DIFFERENCE(*A,B,C*)	C=A\B
MERGE(*A,B,C*)	C=A∪B, nhưng có quan tâm thứ tự

CÀI ĐẶT TẬP HỢP

- CÀI ĐẶT BẰNG VECTƠ BIT
- CÀI ĐẶT BẰNG DANH SÁCH LIÊN KẾT
- CÀI ĐẶT BẰNG TỪ ĐIỂN
- CÀI ĐẶT BẰNG BẢNG BĂM

CÀI ĐẶT TẬP HỢP BẰNG VECTƠ BIT (1)

- Thường được dùng khi tập hợp của ta là 1 tập con của tập số nguyên, có giá trị từ 1..n. Khi đó ta sẽ dùng 1 mảng kiểu boolean có kích thước n để lưu trữ tập hợp
- Phần tử thứ i của mảng có giá trị TRUE nếu i thuộc tập hợp
- VD: muốn lưu trữ các tập có giá trị phần tử từ 1..10. Ta dùng mảng có tối đa 10 phần tử.
- Mô hình cho A={2,5,7,9} là:



CÀI TẬP HỢP ĐẶT BẰNG VECTƠ BIT (2)

- **Khai báo**

```
#define maxlength ...; // giá trị phần tử lớn nhất
typedef  int  SET[maxlength];
```

- **Tạo tập hợp rỗng:**

```
void MakeNull(SET a) {
      int i;
      for(i=0; i<maxlength; i++) a[i]=0;
}
```

CÀI TẬP HỢP ĐẶT BẰNG VECTƠ BIT (2)

- **Kiểm tra thành viên**

```
int Member(int x, SET a) {
        return (a[x] == 1)
}
```

- **Phép hợp**

```
void Union(SET a, SET b, SET c) {
        int i;
        for (i=0;i<maxlength;i++)
            if(Member(i, a) || Member(i, b)) c[i]=1;
            else c[i]=0;
}
```

CÀI ĐẶT TẬP HỢP BẰNG VECTƠ BIT (3)

- **Phép giao**

```
void Intersection(SET a, SET b, SET c) {
   int i;
   for (i=0;i<maxlength;i++)
      if(Member(i, a) && Member(i, b)) c[i]=1;
      else c[i]=0;
}
```

- **Phép hiệu**

```
void Difference(SET a, SET b, SET c) {
   int i;
   for (i=0;i<maxlength;i++)
       if(Member(i, a) && !Member(i, b)) c[i]=1;
       else c[i]=0;
}
```

CÀI ĐẶT TẬP HỢP BẰNG DSLK(1)

- Khai báo

```
typedef int ElementType;
typedef struct Node* NodeType;
struct Node {
   ElementType Data;
   NodeType Next;
};
typedef NodeType Position;
typedef Position SET;
```

CÀI ĐẶT TẬP HỢP BẰNG DSLK(2)

- Khởi tạo tập hợp rỗng

```
void MakeNull(SET *A) {
     (*A)=(NodeType)malloc(sizeof(struct Node));
     (*A)->Next= NULL;
}
```

- Các phép toán cơ bản như DSLK:

Retrieve(p, A), First(A), End(A), Next(p, A), ...

CÀI ĐẶT TẬP HỢP BẰNG DSLK (3)

- Kiểm tra X có thuộc tập A không?

```
int Member(ElementType X, SET A) {
    Position P;
    int Found = 0;
    P = First(A);
    while ((P != End(A)) && (Found == 0))
          if (Retrieve(P, A) == X) Found = 1;
          else P = Next(P, A);
    return Found;
}
```

CÀI ĐẶT TẬP HỢP BẰNG DSLK(4)

- Thêm một phần tử X vào đầu tập A

```
void Insert(ElementType X, SET *A)  {
        Position T;
        T=(NodeType)malloc(sizeof(struct Node));
        T->Data=X;
        T->Next=(*A)->Next;
        (*A)->Next=T;
}
```

CÀI ĐẶT TẬP HỢP BẰNG DSLK(4)

Phép hợp

```
void Union(SET A, SET B, SET *C) {
     Position p;
    MakeNull(C);
    p=First(A);
    while (p!=End(A)) {
       Insert(Retrieve(p, A), C);
       p=Next(p,A);
    }
    p=First(B);
    while (p!=End(B)) {
          if (!Member(Retrieve(p, B), *C))
               Insert(Retrieve(p, B), C);
          p=Next(p,B);
    }
}
```

CÀI ĐẶT TẬP HỢP BẰNG DSLK(5)

- Phép giao

```
void Intersection(SET A, SET B, SET *C) {
  Position p;
  MakeNull(C);
  p=First(A);
  while (p!=End(A)) {
    if (Member(Retrieve(p,A),B))
        Insert(Retrieve(p,A), C);
    p=Next(p,A);
  }
}
```

CÀI ĐẶT TẬP HỢP BẰNG DSLK(6)

- Phép hiệu

```
void Difference(SET A, SET B, SET *C) {
    Position p;
    MakeNull(C);
    p=First(A);
    while (p!=End(A)) {
        if (!Member(Retrieve(p,A),B))
            Insert(Retrieve(p,A), C);
        p=Next(p,A);
    }
}
```

TỪ ĐIỂN

- Khái niệm: là một tập hợp đơn giản với các phép toán INSERT, DELETE và MEMBER
- Có thể cài đặt từ điển bằng:
  - Véctơ-bít
  - Danh sách đặc (mảng)
  - Danh sách liên kết có thứ tự hoặc không thứ tự
  - Mảng có kích thước cố định với con nháy chỉ đến vị trí cuối cùng:
    - Khuyết điểm:
      - kích thước không thể lớn tùy ý
      - xóa một phần tử chậm
      - dùng bộ nhớ không hiệu quả
      - Tương tự cài đặt danh sách bằng mảng

CÀI ĐẶT TỪ ĐIỂN BẰNG MẢNG (1)

- Khai báo

```
#define MaxLength ...   //So phan tu toi da
typedef ... ElementType;   //Kieu du lieu
typedef int Position;
typedef struct {
  ElementType Data[MaxLength];
  Position Last;
} SET;
```

CÀI ĐẶT TỪ ĐIỂN BẰNG MẢNG (2)

- Khởi tạo rỗng

```
void MakeNullSET(SET *A) {
  A->Last=0;
 }
```

- Hàm kiểm tra 1 phần tử có trong từ điển không:

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

CÀI ĐẶT TỪ ĐIỂN BẰNG MẢNG (2)

- Thêm 1 phần tử vào từ điển:

```
void InsertSET(ElementType X, SET *A) {
  if (FullSET(*A))
   printf("Tap hop day");
  else if (Member(X,*A)==0) {
       A->Last++;
       A->Data[A->Last]=X;
  } else
    printf("\nPhan tu da ton tai trong tu dien");
}
```

CÀI ĐẶT TỪ ĐIỂN BẰNG MẢNG (3)

- Xóa 1 phần tử khỏi từ điển:

```
void DeleteSET(ElementType X, SET *A) {
  if (EmptySET(*A))
     printf("Tap hop rong!");
  else {
     Position Q=1;
     while((Q<= A->Last)&&(A->Data[Q]!=X))
        Q++;
     if (A->Data[Q]==X) {
        //int i;
        //for (i=Q; i<A->Last; i++)
        //   A->Data[i]=A->Data[i+1];
          A->Data[Q] = A->Data[A->Last];
          A->Last--;
     }
  }
```

CÀI ĐẶT TỪ ĐIỂN BẰNG BẢNG BĂM (4)

- BĂM ĐÓNG
- BĂM MỞ

BĂM ĐÓNG (1)

- Khai báo

```
#define B 100
#define Deleted -1000
#define Empty 1000
typedef int ElementType;
typedef int Position;
typedef ElementType Dictionary[B];
```

BĂM ĐÓNG (2)

- **Tạo tự điển rỗng**

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

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

```
int Member(ElementType X, Dictionary D) {
  Position P = H(X); // ham bam
  int i=0, Found = 0;
  while ((i < B) && (D[P]!=Empty) &&
         (!Found))
     if ((D[P]) == X) Found = 1;
     else {P=(P+1)%B; i++;}
  return Found;
}
```

BĂM ĐÓNG (3)

- **Thêm phần tử vào tự điển**

```
void InsertDic(ElementType X, Dictionary D){
      Position P;
      if (FullDic(D))
           printf("Bang bam day");
      else if (!Member(X,D)){
          P = H(X);
          int i = 0;
          while((i<B)&& (D[P]!=Empty)&&
                (D[P]!=Deleted)) {i++; P=(P+1)%B;}
          D[P]=X;
      } else
          printf("\nPhan tu da ton tai");
  }
```

BĂM ĐÓNG (4)

- **Xóa từ ra khỏi tự điển**

```
void DeleteDic(ElementType X, Dictionary D) {
        if (EmptyDic(D))
           printf("\nBang bam rong!");
        else {
             int i=0;
             Position P = H(X);
              while ((i<B) && (D[P]!=X) &&
                     (D[P]!=Empty))
                  {i++; P=(P+1)%B;}
             if (D[P]==X)
                 D[P]=Deleted;
        }
    }
```

BĂM MỞ (1)

- Khai báo

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

BĂM MỞ (2)

- **Khởi tạo bảng băm mở rỗng**

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

BĂM MỞ (3)

- **Kiểm tra một thành viên trong từ điển**

```
int Member(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 (4)

- **Thêm một phần tử vào từ điển**

```
void InsertSet(ElementType X, Dictionary *D){
    if (!Member(X,*D)){
      NodeType temp;
      temp =(NodeType)malloc(sizeof(struct Node));
      temp->Data = X;
      temp-> Next = (*D)[H(X)];
      (*D)[H(X)] = temp;
    }
}
```

BĂM MỞ (5)

- **Xoá một phần tử trong từ điển**

```
void DeleteSet(ElementType X, Dictionary *D) {
   Position P, Q;
   if((*D)[H(X)]!=NULL) {
     if ((*D)[H(X)]->Data==X) {
           Q=(*D)[H(X)];
          (*D)[H(X)]=(*D)[H(X)]->Next;
          free(Q);
      } else {
         int Found = 0;
        P=(*D)[H(X)];
         while ((P->Next!=NULL) && (!Found))
             if (P->Next->Data==X) Found=1;
             else P=P->Next;
         if(Found) {
              Q=P->Next;
              P->Next=Q->Next;
              free(Q);
          }}}}
```