

## Struttura dati SET implementata come Vettore Booleano

```
bool[] V
int size
int dim

SET Set(int m)
┌   SET t ← new SET
├   t.size ← 0
├   t.dim ← m
├   t.V ← [falso] * m
└   ritorna t

SET contains(int x)
┌   se  $1 \leq x \leq \text{dim}$  allora
├   │   ritorna V[x]
├   allora
├   │   ritorna falso
└   ritorna falso

int size
└   ritorna size

insert(int x)
┌   se  $1 \leq x \leq \text{dim}$  allora
├   │   se not V[x] allora
├   │   │   size++
├   │   │   V[x] ← vero
├   se not  $1 \leq x \leq \text{dim}$  allora
├   │   errore
└   fine

remove(int x)
┌   se  $1 \leq x \leq \text{dim}$  allora
├   │   se V[x] allora
├   │   │   size --
├   │   │   V[x] ← falso
├   se not  $1 \leq x \leq \text{dim}$  allora
├   │   errore
└   fine
```

```
SET union(SET A, SET B)
┌   // crea un insieme della capacità max
├   SET C ← Set(max(A.dim, A.dim))
├   // inserisci gli elementi di A
├   da i ← 1 fino a A.dim fai
├   │   se A.contains(i) allora
├   │   │   C.insert(i)
├   // inserisci gli elementi di B
├   da i ← 1 fino a B.dim fai
├   │   se A.contains(i) allora
├   │   │   C.insert(i)
└   ritorna C

SET intersection(SET A, SET B)
┌   // crea un insieme della capacità min
├   SET C ← Set(min(A.dim, A.dim))
├   da i ← 1 fino a min(A.dim, A.dim) fai
├   │   // se è contenuto in entrambi
├   │   se A.contains(i) and B.contains(i) allora
├   │   │   C.insert(i) // aggiungilo
└   ritorna C

SET difference(SET A, SET B)
┌   SET C ← Set(A.dim)
├   da i ← 1 fino a A.dim fai
├   │   // se è contenuto A e non in B
├   │   se A.contains(i) and not B.contains(i) allora
├   │   │   C.insert(i) // aggiungilo
└   ritorna C
```