UNIVERSIDADE FEDERAL DO CEARÁ

CK0109 2019.2 T02 - ESTRUTURAS DE DADOS

AULA 12 - 06/09/2019

CONJUNTOS DINÂMICOS

1. INTRODUÇÃO: O TAD CONJUNTO DINÂMICO É TAMBÉM UM CONTÊINER, ASSIM COMO PILHAS E FILAS. UM CONJUNTO DINÂMICO SE DIFERE DOS CONJUNTOS "MATEMÁTICOS" POR ESTES ÚLTIMOS SEREM ESTÁTICOS, NO SENTIDO DE QUE NÃO TÊM SEUS ELEMENTOS ALTERADOS (ASSIM, POR EXEMPLO, NA TEORIA DOS CONJUNTOS, SE $C = A \cup \{x\}$ E $x \notin A$, ENTÃO C, QUE É O RESULTADO DA "INSERÇÃO" DE x EM A, É OUTRO CONJUNTO: $C \neq A$).

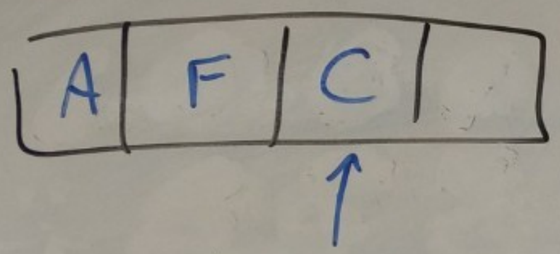
CONJUNTOS DINÂMICOS TÊM UMA DIFERENÇA CRUCIAL EM RELAÇÃO A PILHAS E FILAS: NELES, QUALQUER ELEMENTO ARMAZENADO PODE SER REMOVIDO. ALÉM DISSO, É POSSÍVEL CONSULTAR SE UM ELEMENTO QUALQUER ESTÁ OU NÃO NO CONJUNTO.

AS SEGUINTES OPERAÇÕES SÃO, ENTÃO, ESSENCIAIS EM CONJUNTOS DINÂMICOS:

- INSERIR(C, e): INSERE O ELEMENTO e NO CONJUNTO C.
- PERTENCE(C, e): INFORMA SE $e \in C$ OU NÃO.
- REMOVER(C, e): REMOVE e DE C.

O PERCURSO DOS ELEMENTOS DE UM CONJUNTO SERÁ ESTUDADO NA PRÓXIMA AULA; OUTRAS OPERAÇÕES, COMO A UNIÃO E A INTERSEÇÃO, SÃO TÍPICAS, MAS OMITIREMOS.

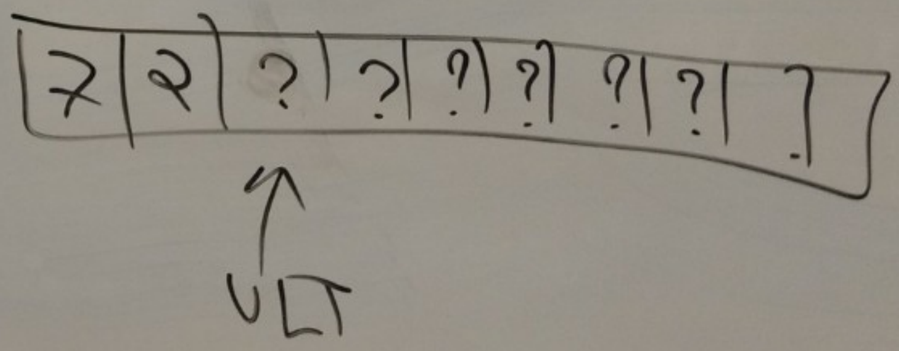
2. Representação via Vetor:



3. Uma implementação:

a) Especificação:

```
TEMPLATE <TYPENAME T>

STRUCT CONJUNTO
  {
  ... CAMPOS/DADOS...

  BOOL INICIALIZAR() {...}
    ↳ TRUE ↔ ERRO DE ALOCAÇÃO

  //CONJUNTO

  //PRÉ-COND.: "E" NÃO PERTENCE.
  BOOL INSERIR (T E) {...}
    ↳ TRUE ↔ ERRO DE ALOC.

  BOOL PERTENCE (T E) {..}
    ↳ TRUE ↔ PERTENCE

  //PRÉ-COND.: "E" PERTENCE.
  BOOL REMOVER (T E) {...}

  //VOCÊ VAI PRECISAR DE "REDIMENSIONAR"...
  };
```



b) Implementação:

```
#include <new>
template <typename T>

struct Conjunto
  {
    T *v; int tam_v, ult;

    bool iniciazizar ( )
      {
        v = new(std::nothrow) T[1];
        if(v==nullptr) return true;
        tam_v=1; ult = -1;
        return false;
      }

    // redimensionar...
```

```
    //Conjunto
    //pré-cond.: não pertence.
    bool inserir (T e)
      {
        if(ult == tam_v -1)
          {
            if( redimensionar(2*tam_v))
              { return true; }
          }
        ++ult; v[ult]=e;
        return false;
      }
    bool vazio ()
      {
        return (ult==-1);
      }
```

```
//CONJUNTO

BOOL PERTENCE (T E)
  {
    FOR(INT i=0; i<=ULT; ++i)
       {
         if(v[i]==E) RETURN TRUE;
       }
    RETURN FALSE;
  }

BOOL REMOVER (T E) // PRÉ-COND.: PERTENCE
  {
    INT i=0;
    WHILE(v[i]!=E) ++i;
    v[i]=v[ULT]; --ULT;
    if(!VAZIO() && 4*(ULT+1)<=TAM_v)
       {
         if(REDIMENSIONAR(TAM_v/2))
            { RETURN TRUE; }
       }
    RETURN FALSE;
  }
```