

Matemática

Teoria dos Conjuntos

Prof. Edson Alves

Faculdade UnB Gama

Teoria dos Conjuntos

Axioma Fundamental

O principal axioma da Teoria dos Conjuntos, que relaciona os termos primitivos **elemento** e **conjunto**, diz que a afirmação “um elemento pertence a um conjunto” é uma proposição.

A simplicidade aparente deste axioma esconde dois importante fatos:

1. a pertinência estabelece a relação entre elementos e conjuntos: dado um elemento e um conjunto quaisquer, este elemento pertence (ou não) ao conjunto
2. a Teoria dos Conjuntos fica edificada sobre a Lógica Proposicional Booleana, uma vez que o que vale para proposições valerá para este axioma também

- Em geral, elementos são representados por letras minúsculas ou símbolos (por exemplo, a, b, x, π, \dots)
- Os conjuntos são representados, em geral, por letras maiúsculas, possivelmente cursivas ou estilizadas (por exemplo, $A, B, \mathbb{N}, \mathcal{F}, \dots$)
- A notação $x \in A$ indica que o elemento x pertence ao conjunto A
- Caso x não pertença ao conjunto A , a notação é $x \notin A$

Subconjuntos

- Um conjunto B é **subconjunto** do conjunto A se, para qualquer elemento $b \in B$, vale que $b \in A$
- A notação para subconjuntos é $B \subset A$, a qual também pode ser lida como “ B está contido em A ”
- Dizer que “o elemento x está contido no conjunto A ” ou que “o conjunto A pertence ao conjunto B ” não só é impreciso como é logicamente falso
- A relação de pertinência se dá entre conjuntos e elementos
- A relação de subconjunto, associada ao termo “contido”, se dá entre conjuntos
- Elementos se relacionam entre si por relação de igualdade

O Conjunto Vazio

- O axioma fundamental permite definir precisamente um conjunto especial, denominado **conjunto vazio**
- \emptyset é o conjunto vazio se, para qualquer elemento x , $x \notin \emptyset$
- Veja que esta definição não é baseada na ideia de cardinalidade (número de elementos de um conjunto)
- Ainda assim, ela permite provar fatos importantes, como o fato de que o conjunto vazio é único ou que qualquer conjunto contém o conjunto vazio

Caracterização de Conjuntos

Duas possíveis formas de se caracterizar um conjunto são:

1. a enumeração de todos os seus elementos
2. descrição das propriedades comuns a todos os elementos do conjunto

Formalmente, se $P(x)$ é uma sentença aberta em x (isto é, uma sentença tal que, uma vez atribuído um valor específico para a variável x , tal sentença se torna uma proposição), então

$$\{ x \in X \mid P(x) \text{ é verdadeira} \}$$

é um conjunto, onde X é o conjunto de todos os possíveis valores de x .

Exemplos de conjuntos

1. Conjunto de constantes notáveis

$$C = \{ e, \pi, 0, -1 \}$$

2. Conjunto dos números ímpares

$$I = \{ 2x + 1 \mid x \in \mathbb{Z} \}$$

3. Conjunto dos números primos

$$P = \{ p \in \mathbb{N} \mid p \text{ é primo} \}$$

Dados dois conjuntos A e B , é possível definir três novos conjuntos:

1. conjunto **união** $A \cup B = \{ x \mid x \in A \vee x \in B \}$
2. conjunto **interseção** $A \cap B = \{ x \mid x \in A \wedge x \in B \}$
3. conjunto **diferença** $A - B = \{ x \mid x \in A \wedge x \notin B \}$

Operações em Conjuntos

Observe que as três operações em conjuntos são definidas em termos dos conectivos lógicos fundamentais:

- disjunção (união),
- conjunção (interseção), e
- negação (diferença).

Esta relação permite a verificação das propriedades destes operadores e a relação entre eles (como o equivalente das Leis de Morgan para união e interseção).

Conjuntos em C/C++

- Há quatro maneiras de se representar conjuntos em C e C++: as classes `set`, `multiset`, `bitset` e o tipo primitivo `int`, sendo que a última representação também é válida em C
- A biblioteca padrão do C++ traz a implementação da classe `set` (`#include <set>`), que abstrai a ideia de conjuntos
- Esta classe provê operações elementares como inserção e remoção de elementos, através dos métodos `insert()` e `erase()`, ou relações de pertinência, com o método `count()`

Conjuntos em C/C++

- Na classe `set` os elementos são únicos e armazenados ordenadamente (uma travessia padrão é feita do menor para o maior elemento)
- A classe `multiset` permite a repetição de um mesmo elemento, porém o processo de remoção deve ser feito de forma mais cuidadosa
- As operações de união, interseção e diferença de conjuntos podem ser feitas em qualquer contêiner ordenado, através das funções `set_union()`, `set_intersection()` e `set_difference()`, disponíveis na biblioteca de algoritmos (`#include <algorithm>`)

Exemplo de uso da classe set

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 int main()
6 {
7     vector<int> A { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };           // Conjunto A
8     vector<int> B { 2, 3, 5, 7, 11, 13 };                     // Conjunto B
9     vector<int> C;
10
11     set_union(A.begin(), A.end(), B.begin(), B.end(), back_inserter(C));
12
13     // C = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13 }
14     cout << "union = ";
15     for (size_t i = 0; i < C.size(); ++i)
16         cout << C[i] << (i + 1 == C.size() ? '\n' : ' ');
17
18     C.clear();
19     set_intersection(A.begin(), A.end(), B.begin(), B.end(), back_inserter(C));
```

Exemplo de uso da classe set

```
21  cout << "intersection = ";
22  for (size_t i = 0; i < C.size(); ++i)
23      cout << C[i] << (i + 1 == C.size() ? '\n' : ' ');    // C = { 2, 3, 5, 7 }
24
25  C.clear();
26  set_difference(A.begin(), A.end(), B.begin(), B.end(), back_inserter(C));
27
28  cout << "difference = ";
29  for (size_t i = 0; i < C.size(); ++i)
30      cout << C[i] << (i + 1 == C.size() ? '\n' : ' ');    // C = { 1, 4, 6, 8, 9, 10 }
31
32  return 0;
33 }
```

- O tipo de dados primitivo **int** (ou sua variante **long long**, com maior capacidade de armazenamento) também pode ser usado para uma representação compacta e eficiente de conjuntos
- Em geral o tipo **int** tem 32 *bits* de tamanho
- É possível associar cada elemento do conjunto universo (que contém todos os elementos possíveis, numa quantidade menor ou igual a 32) a cada *bit*, de modo que, se o *bit* está ligado, o elemento pertence ao conjunto; se está desligado, o elemento não pertence ao conjunto

A principal restrição da representação de conjuntos por meio de inteiros é o número de elementos do conjunto universo (limitado pelo número de *bits* do tipo escolhido).

Contudo esta representação tem várias vantagens, dentre elas:

1. ocupa pouco espaço em memória (4 *bytes* a cada 32 elementos);
2. permite responder relações de pertinência em complexidade $O(1)$;
3. permite operações de união, interseção e diferença também em $O(1)$.

- A última vantagem listada se dá por conta da definição de tais operações em termos dos conectivos lógicos
- Lembre que as linguagens C e C++ disponibilizarem tais conectivos tanto em relação à variáveis booleanas quanto em versões *bit a bit*
- A seguir o exemplo anterior é reescrito em termos desta nova representação
- No código, considere o conjunto universo $U = \{1, 2, 3, \dots, 32\}$

Exemplo de representação de conjuntos por meio de inteiros

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int A = 2046;           // A = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }
8     int B = 10412;          // B = { 2, 3, 5, 7, 11, 13 }
9     int C = A | B;          // C = 12286
10
11     cout << "union = ";
12     for (int x = 0; x < 32; ++x)
13         if (C & (1 << x))
14             cout << x << " ";
15     cout << endl;           // C = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13 }
16
17     C = A & B;               // C = 172
18
19     cout << "intersection = ";
```

Exemplo de representação de conjuntos por meio de inteiros

```
19  cout << "intersection = ";
20  for (int x = 0; x < 32; ++x)
21      if (C & (1 << x))
22          cout << x << " ";
23  cout << endl;                                // C = { 2, 3, 5, 7 }
24
25  C = A & ~B;                                    // C = 1874
26
27  cout << "difference = ";
28  for (int x = 0; x < 32; ++x)
29      if (C & (1 << x))
30          cout << x << " ";
31  cout << endl;                                // C = { 1, 4, 6, 8, 9, 10 }
32
33  return 0;
34 }
```

Classe bitset

- Para conjuntos universos com mais de 32 elementos (ou 64, no caso de variáveis **long long**), as alternativas são o uso ou de um vetor de inteiros, ou da classe bitset (`#include <bitset>`)
- Esta classe pode armazenar uma quantidade arbitrária de *bits* (que deve ser conhecida em tempo de compilação)
- Ela traz em sua interface as operações básicas dos conjuntos e suporte para os operadores lógicos *bit a bit*

Exemplo de uso da classe bitset

```
1 #include <bits/stdc++.h>
2
3 int main()
4 {
5     std::bitset<35> A(2046), B(10412);
6
7     std::cout << A.to_string() << '\n';
8     std::cout << B.to_string() << '\n';
9
10    auto C = A | B;
11    std::cout << "union = " << C.to_string() << '\n';
12
13    C = A & B;
14    std::cout << "interseção = " << C.to_string() << '\n';
15
16    C = A & ~B;
17    std::cout << "diferença = " << C.to_string() << '\n';
18
19    return 0;
20 }
```

1. **CppReference**. [std::bitset](#), acesso em 31/12/2020.
2. **CppReference**. [std::multiset](#), acesso em 31/12/2020.
3. **CppReference**. [std::set](#), acesso em 31/12/2020.
4. **HALE**, Margie. *Essentials of Mathematics: Introduction to Theory, Proof, and the Professional Culture*. Mathematical Association of America, 2003.