Algoritmos de operação de conjuntos utilizando python

Alunos: Ezequiel, Kennedy, Taylor e Alexandre Matos

Algoritmo 1: Intersecção(A, B)

> Entradas: A e B são conjuntos
> Saída: o conjunto resultante da intersecção

1. $C \leftarrow \emptyset$
2. **para cada** $a \in A$ **faça**
3. **se** $a \in B$ **faça**
4. $C \leftarrow a$
5. **fim**
6. **fim**
7. **retorne** C

Explicação:

O algoritmo interseccao demonstra como funcionaria um programa que faz a intersecção entre dois conjuntos. Como argumentos para o método têm-se duas variáveis ambas conjuntos numéricos. Na linha 1 é atribuído um conjunto vazio a C. Depois o primeiro conjunto passado como parâmetro (A) é percorrido e, a cada elemento deste, faz-se uma comparação onde o objetivo é verificar se o elemento atual em análise também é um elemento do segundo conjunto, também passado como parâmetro (B). Caso essa comparação retorne verdade, o elemento é atribuído ao conjunto - antes vazio - C. Por fim, o algoritmo retorna todo o conjunto C, onde os elementos contidos nele são os elementos que fazem intersecção entre os dois conjuntos analisados.

Código em python:

```
def intersecao(lista1, lista2):
    lista3 = {}
    for n in lista1:
        if n in lista2:
            lista3.append(n)
    return lista3
```

Demonstração:

Entrada:
```
lista1=[1, 5, 6, 8]
lista2=[1, 10,8, 40]
```

Saída:
```
Conjunto intersecção:  [1, 8]
```

Algoritmo 2: União(A, B)

> Entradas: A e B são conjuntos**
> Saída: o conjunto resultante da união
> Insere( ): insere um elemento ao conjunto

1. C $\leftarrow \emptyset$
2. **para cada** $a \in A$ **faça**
3. Insere(C, a)
4. **fim**
5. **para cada** $b \in B$ **faça**
6. Insere(C, b)
7. **fim**
8. **retorne** C

Explicação:

O algoritmo de união realiza a união entre dois conjuntos. Ele recebe como parâmetros dois conjuntos sob os quais a operação de união será realizada (A e B). Utilizando algumas iterações o conjunto A é percorrido, e cada um dos seus elementos é adicionado a um conjunto resultante da união (C). O conjunto B recebido como parâmetro também é percorrido da mesma forma que o A foi, e, para cada elemento do conjunto B, é verificado se o elemento já existe no conjunto resultante (C), caso o elemento não exista, ele é adicionado a esse conjunto. Para finalizar a rotina, é retornado o conjunto resultante da união entre os dois conjuntos recebidos como parâmetros na função.

Código python:

```python
def uniao(A, B):
    C = []
    for a in A:
        C.append(a)
    for e in B:
        if e not in C:
```

```
            C.append(e)
    return print(C)
```

Demonstração:

Entradas:
```
A = [1, 2, 3, 4]
B = [3, 4, 5, 6]
```

Saída:
```
Uniao:  [1, 2, 3, 4, 5, 6]
```

Algoritmo 3: Diferença(A, B)

> Entradas: A e B são conjuntos
> Saída: o conjunto resultante da diferença entre os conjuntos.

1. $C \leftarrow \emptyset$
2. **para cada** $a \in A$ **faça**
3. **se** $a \notin B$ **faça**
4. $C \leftarrow a$
5. **fim**
6. **fim**
7. **retorne** C

Explicação:

O algoritmo diferenca retorna a diferença entre dois conjuntos. Ele recebe como parâmetro duas entradas que são conjuntos numéricos, A e B respectivamente. Na outra linha é atribuído um conjunto vazio a C (linha 1).O conjunto A é percorrido e cada elemento dele é comparado, onde o objetivo é descobrir se o elemento em questão não está em B. Caso retorne verdade, o elemento analisado naquele momento será acrescentado no final do conjunto C (ou início, caso seja o primeiro elemento do conjunto). Por fim o conjunto C é retornado, exibindo o conjunto diferença entre os conjuntos analisados.

Código em Python:

```
def diferenca(A, B):
    C = []
    for a in A:
        if a not in B:
            B.append(a)
    return C
```

Demonstração:

Entrada:
```
A=[5, 8, 9,6]
B=[5, 10 ,11 , 7]
```

Saída:
```
Conjunto da diferença:  [8, 9, 6]
```

Algoritmo 4: Plano cartesiano(A, B)

> Entradas: A e B são conjuntos
> Saída: Produto Cartesiano dos dois conjuntos.

1. $C \leftarrow \emptyset$
2. **para cada** $x \in A$ **faça**
3. **para cada** $y \in B$ **faça**
4. imprima(x, y)
5. **fim**
6. **fim**

Explicação:

O algoritmo de Produto Cartesiano realiza o produto cartesiano entre dois conjuntos. O algoritmo recebe como parâmetro dois conjuntos A e B sob os quais o produto cartesiano será realizado. Utilizando um loop controlado o primeiro conjunto (a) recebido como parâmetro na função é percorrido, dentro do loop controlado do primeiro conjunto (a), o segundo conjunto (b) recebido como parâmetro na função também é percorrido, e para cada um dos seus elementos em suas respectivas posições, a função imprima é responsável por criar o par com o elemento da iteração do primeiro (a) com o segundo (b) conjunto. Ao final do algoritmo, tem-se o produto cartesiano do conjunto A sob o conjunto B.

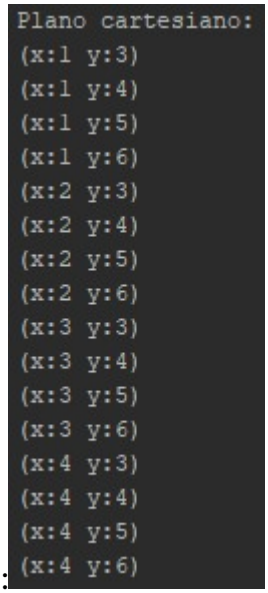
Código em Python

```
def pCartesiano(A, B):
    for x in A:
        for y in B:
            print('(x:{0} y:{1})'.format(x, y))
```

Demonstração:

Entrada:
```
A = [1, 2, 3, 4]
B = [3, 4, 5, 6]
```

Saída:



Algoritmo 5: Complemento();

> Entradas: A e B são conjuntos
> Saída: O conjunto complementar, dos elementos de A, que não estão em B.

1. $C \leftarrow \emptyset$
2. n $\leftarrow \mid C \mid$
3. **para cada** $a \in A$ **faça**
4. **se** $a \notin B$ **faça**
5. $C_{n+1} \leftarrow a$
6. **fim**
7. **fim**
8. **retorne** C

O algoritmo 5 Complemento, recebe dois conjuntos A e B como parâmetro e retorna um conjunto C com o complemento entre os dois conjuntos de entrada. Na linha 1 temos a definição do conjunto C, na linha 2 temos a variável n que recebe o tamanho do conjunto C, logo após na linha 3 define-se que para cada elemento do conjunto A, será executado o bloco de repeticao. Na linha 4 é verificado se o elemento em questão não pertence ao conjunto B, sendo verdadeira a condição, o conjunto C recebe o elemento na última posição. Na linha 6 e 7 temos o fim do bloco de repetição e o condicional respectivamente.E por fim na linha 8 tem o retorno com o conjunto C, contendo os elementos que são complementos entre os conjuntos A e B.

Código em python:

```
def complemento(A, B):
    C = []
```

```
    for x in A:
        if x not in B:
            C.append(x)
    return C
```

Demonstração:

Entrada:

```
A = [1, 2, 3, 4]
B = [3, 4, 5, 6]
```

Saída:

```
Complemento:  [1, 2]
```

Algoritmo 6: ConjuntoDasPartes

Algoritmo 7: UniãoDisjunta(A, B);

> Entradas: A e B são conjuntos
> Saída: o conjunto resultante da união disjunta entre os conjuntos.

1. $C \leftarrow \emptyset$
2. **para cada** $a \in A$ **faça**
3. inserePar(a, $A.nome$)
4. **fim**
5. **para cada** $b \in B$ **faça**
6. inserePar(b, $B.nome$)
7. **fim**
8. **retorna** C
9. **fim**

Explicação:

O algoritmo 7 é utilizado para retornar um conjunto resultante da união disjunta entre dois conjuntos. Na linha 1 é atribuído a C um conjunto vazio. Na linha 2, A é percorrido onde cada elemento deste conjunto, juntamente com o nome do conjunto são passados por parâmetro para outro método - inserePar(a, idA) - que insere o par contendo o elemento e o nome do conjunto, em outro conjunto. Na linha 5 o outro conjunto é percorrido da mesma forma que o A foi. No final é retornado o conjunto contendo todos os pares, ou seja, a união disjunta entre os dois conjuntos analisados.

Código em python:

```
def uniaoDisjunta(A, B):
    C = []
    for a in A:
        C.append(inserePar(a, 'A'))
    for b in B:
        C.append(inserePar(b, 'B'))
    print(C)
```

Demonstração:

Entrada:
```
A = [1, 3, 4]
B = [3, 5, 6]
```

Saída:
```
Conjunto União Disjunta:  [[(1, 'A')], [(3, 'A')], [(4, 'A')], [(3, 'B')], [(5, 'B')], [(6, 'B')]]
```

Algoritmo 8: InserePar(a, idA)

> Entradas: a: é um valor do conjunto
> idA: a identificação do conjunto que a pertence.
> Saída: conjunto contendo pares formados por um elemento e o nome do conjunto ao qual o elemento pertence

1. $C \leftarrow \emptyset$
2. $C \leftarrow (a, idA)$
3. **fim**

Explicação:
Na primeira linha é atribuído a C o conjunto vazio, onde posteriormente será passado uma tupla que deve ser inserida em C.

Código em python:

```
def inserePar(a, Anome):
    C = []
    C.append((a, Anome))
    return C
```

Algoritmo 9: insere(A, a)

> Entradas: A : é um conjunto a : é um valor que será inserido em A

se $a \notin A$ **então**
$n \leftarrow \mid A \mid$
$A_{n+1} \leftarrow a$
fim