Instituto de Computação - Unicamp

MC102 - Algoritmos e Programação de Computadores

Laboratório 16 - Conjuntos

Prazo de entrega: **07/06/2019 23:59:59**

Peso: **2**

Professor: Eduardo C. Xavier

Professor: Luiz Fernando Bittencourt

Descrição

A Teoria dos Conjuntos é um ramo da matemática que estuda propriedades de coleções de objetos, e é utilizada para melhor entendermos e modelarmos diversos problemas. Um conjunto é uma coleção de elementos distintos. Por exemplo, os números *2*, *3* e *5* são elementos distintos quando considerados isoladamente, mas quando considerados coletivamente formam o conjunto *{2,3,5}* e, nesse caso, *2*, *3* e *5* são os elementos do conjunto. A cardinalidade ou tamanho de um conjunto é o número de elementos desse conjunto. Note que um conjunto não pode ter mais de um elemento do mesmo valor. Por exemplo *{1,2,1,2,3}* não é um conjunto, mas *{1,2,3}* sim.

Abaixo temos uma lista de definições e operações que podem ser realizadas sobre conjuntos:

- Dado um conjunto A, se x é um de seus elementos então dizemos que x **pertence** ao conjunto A, e denotamos isto por $x \in A$. Caso x não pertença ao conjunto A, denotamos isto por $x \notin A$.
- Dados dois conjuntos A e B, se cada um dos elementos de B pertencer também a A, então dizemos que A **contém** B, ou de forma equivalente B **está contido** em A, e denotamos isto por $B \subseteq A$. Caso B não esteja contido em A, denotamos isto por $B \not\subset A$.

- A operação de **união** de dois conjuntos *A* e *B*, denotada por $A \cup B$, tem como resultado um outro conjunto contendo os elementos que estão em *A* ou *B*. Por exemplo, se $A = \{1,2,3\}$ e $B = \{2,3,4\}$, a operação $A \cup B$ terá como resultado o conjunto $\{1,2,3,4\}$.

- A operação de **interseção** de dois conjuntos *A* e *B*, denotada por $A \cap B$, tem como resultado um outro conjunto contendo os elementos que estão em ambos conjuntos *A* e *B*. Por exemplo, se $A = \{1,2,3\}$ e $B = \{2,3,4\}$, sua interseção $A \cap B$ será $\{2,3\}$.

- A operação **diferença** do conjunto *A* para o conjunto *B*, denotada por $A \setminus B$, tem como resultado um conjunto contendo os elementos que estão em *A* mas não estão em *B*. Por exemplo, se $A = \{1,2,3\}$ e $B = \{2,3,4\}$, a diferença $A \setminus B$, será $\{1\}$, enquanto que a diferença $B \setminus A$ será $\{4\}$.

- A operação **união disjunta** do conjunto *A* e *B*, denotada por $A \setminus B \cup B \setminus A$, tem como resultado um conjunto contendo os elementos de $A \setminus B$ e os elementos de $B \setminus A$. Por exemplo, se $A = \{1,2,3\}$ e $B = \{2,3,4\}$, a operação $A \setminus B \cup B \setminus A$, resultará em $\{1,4\}$.

O objetivo deste laboratório é criar uma biblioteca de funções em Python para realizar operações sobre conjuntos de números naturais. Os conjuntos serão representados utilizando-se listas.

Você deve implementar funções que realizam as seguintes operações:

- **Pertence:** verifica se um elemento pertence ao conjunto especificado, retornando verdadeiro ou falso.

- **Continência:** verifica se um conjunto está contido em outro conjunto retornando verdadeiro ou falso.

- **Adição:** adiciona um elemento em um conjunto, alterando o conjunto com a adição do novo elemento *caso ele já não pertença ao mesmo*.

- **Subtração:** remove um elemento de um conjunto, alterando o conjunto especificado com a remoção do elemento *caso ele pertença ao conjunto*.

- **União:** faz a união de dois conjuntos, criando um conjunto com os elementos dos dois conjuntos de entrada.

- **Interseção:** faz a interseção de dois conjuntos, criando um conjunto com os elementos que pertencem aos dois conjuntos de entrada.

- **Diferença:** Faz a diferença de dois conjuntos, criando um conjunto com os elementos do primeiro que não se encontram no segundo.
- **União Disjunta:** Faz a união disjunta de dois conjuntos, criando um conjunto que corresponde a definição dada anteriormente.

Funções

Observações gerais:

- Os conjuntos armazenam apenas números naturais: 0, 1, 2, etc.
- O tamanho máximo de cada conjunto é 20.
- O tamanho atual de cada conjunto é passado por parâmetro.
- Os vetores que armazenam os conjuntos não estão e não precisam estar ordenados.

A descrição geral dos parâmetros de entrada e saída das funções está descrita nos comentários dos protótipos das funções, que são fornecidos a seguir:

Linguagem Python 3:

```
#!/usr/bin/env python3

# Funcao: pertence
#
# Parametros:
#   conj: vetor contendo o conjunto de entrada
#    num: elemento a ser verificado pertinencia
#
# Retorno:
#   True se num pertence a conj e False caso contrario
#
def pertence(conj, num):
    # Implementar a funcao e trocar o valor de retorno
    return False

# Funcao: contido
#
# Parametros:
#   conj1: vetor contendo um conjunto de entrada
#   conj2: vetor contendo um conjunto de entrada
#
# Retorno:
#   True se conj1 esta contido em conj2 e False caso contrario
#
```

```
def contido(conj1, conj2):
    # Implementar a funcao e trocar o valor de retorno
    return False

# Funcoes: adicao e subtracao
#
# Parametros:
#   conj: vetor contendo o conjunto que tera incluso ou removido o elemento
#    num: elemento a ser adicionado ou removido
#
def adicao(conj, num):
    # Implementar a funcao
    return

def subtracao(conj, num):
    # Implementar a funcao
    return

# Funcoes: uniao, intersecao e diferenca
#
# Parametros:
#     conj1: vetor contendo o conjunto de entrada do primeiro operando
#     conj2: vetor contendo o conjunto de entrada do segundo operando
#
# Retorno:
#   Vetor contendo o conjunto de saida/resultado da operacao
#
def uniao(conj1, conj2):
    # Implementar a funcao e trocar o valor de retorno
    return []

def intersecao(conj1, conj2):
    # Implementar a funcao e trocar o valor de retorno
    return []

def diferenca(conj1, conj2):
    # Implementar a funcao e trocar o valor de retorno
    return []

def uniao_disjunta(conj1, conj2):
    # Implementar a funcao e trocar o valor de retorno
    return []
```

Múltiplos Arquivos e Função Principal

Neste laboratório vamos utilizar o conceito de dividir o código em múltiplos arquivos. Quando se implementa programas grandes é comum separar o código

em vários arquivos com a extensão .py, onde cada arquivo implementa um conjunto de funções relacionadas entre si. Isto facilita a manutenção e a leitura do código.

Para esse laboratório você só deverá implementar as funções descritas acima. A função principal (**main**) será fornecida em um arquivo separado, chamado `lab16_main.py`.
Um link para ele também está disponível na página da tarefa.

Para executar o seu programa em Python 3, basta executar o arquivo `lab16_main.py`. O arquivo `lab16.py` deverá estar na mesma pasta e será chamado pelo primeiro.

```
python3 lab16_main.py
```

A organização do conteúdo de cada arquivo é a seguinte:

- `lab16`:
  - funções auxiliares que você queira escrever;
  - `pertence(...)`;
  - `contido(...)`;
  - `adicao(...)`;
  - `subtracao(...)`;
  - `uniao(...)`;
  - `intersecao(...)`;
  - `diferenca(...)`;
  - `uniao_disjunta(...)`.
- `lab16_main`:
  - funções auxiliares para a main;
  - main.

Também está disponível um protótipo do arquivo que você deve submeter ao *SuSy* (lab16.py). Esse arquivo e o arquivo auxiliar (lab16_main.py) também podem ser encontrados na página da tarefa:

- `lab16.py`
- `lab16_main.py`

Reforçando

Neste laboratório você não precisará se preocupar em ler a entrada a partir da entrada padrão, nem em escrever a saída. Seu trabalho é apenas implementar as funções descritas. A função `main()` que é fornecida no arquivo `lab16_main.py` se encarrega dessa parte.

Você também **não deve** submeter o arquivo `lab16_main.py` para o *SuSy*, somente o arquivo `lab16.py`.

As sessões abaixo, de Entrada e Saída, descrevem os formatos de entrada e saída, mas você não precisa se preocupar com eles.

Entrada

A entrada consiste de operações a serem realizadas sobre dois conjuntos nomeados de A e B. Os conjuntos iniciam vazios e cada linha da entrada descreve uma operação a ser realizada sobre um ou entre os dois conjuntos.

As operações são:

- `C` : Não faça nada no conjunto, Só imprima o seu conteúdo
- `C = {x1, x2, x3, ... ,xn}` : substitui o conteúdo do conjunto `C` com os `N` elementos.
- `x e C` : verifica se o elemento `x` pertence ao conjunto `C`.
- `C1 c C2` : verifica se o conjunto `C1` está contido no conjunto `C2`.
- `C += x` : adiciona o elemento `x` ao conjunto `C`.
- `C -= x` : remove o elemento `x` do conjunto `C`.
- `C1 u C2` : realiza a união entre `C1` e `C2`
- `C1 ^ C2` : realiza a interseção entre `C1` e `C2`
- `C1 \ C2` : realiza a diferença entre `C1` e `C2`
- `q` : Encerra a execução do programa

Onde:

- `C` é um dos conjuntos `A` ou `B`
- `C1` e `C2` são conjuntos distintos `A` e `B` em qualquer ordem.
- `x` é um elemento
- `x1, x2, x3, ... , xn` são `N` elementos.

- `q` é a letra q

Saída

Cada linha da saída do programa contém o resultado da execução de cada operação dada na entrada, de forma que a saída possui uma linha a menos que a quantidade de linhas da entrada.

O retorno das operações podem ter um dos 3 formatos distintos:

- Para as operações `C`, `C = {x1, x2, x3, ... ,xn}`, `C += x`, ou `C -= x`:

  Imprime o conteúdo do conjunto C, com os elementos em ordem crescente.

  Formato da saída: `C = {x1, x2, x3, ..., xn}`

- para as operações `x e C` ou `C1 c C2`:

  Imprime Verdadeiro ou Falso.

  Formato: `verdadeiro` ou `falso`

- Para as operações `C1 u C2`, `C1 ^ C2` ou `C1 \ C2`:

  Imprime o conjunto resultado da operação, com os elementos em ordem crescente.

  Formato: `C1 op C2 = {x1, x2, x3, ..., xn}` *

  * Onde `op` é o operador correspondende da operação de entrada, entre `u`, `^` e `\`.

- Para a operação `q` nada é impresso. Se encerra a execução do programa.

Exemplos

Teste 03

Entrada

```
A = {2, 5, 7}
B = {2, 5}
B c A
```

```
A c B
B += 3
B c A
q
```

Saída

```
A = {2, 5, 7}
B = {2, 5}
verdadeiro
falso
B = {2, 3, 5}
falso
```

Teste 04

Entrada

```
A = {2, 5, 7}
B = {}
B c A
A c B
B = {5}
B c A
B = {1, 5}
B c A
A c B
q
```

Saída

```
A = {2, 5, 7}
B = {}
verdadeiro
falso
B = {5}
verdadeiro
B = {1, 5}
falso
falso
```

Teste 15

Entrada

```
A
B
A c B
B c A
A = {1, 2, 3}
B = {4, 3, 2}
A c B
B c A
1 e A
1 e B
4 e A
4 e B
A += 0
B += 1
A -= 3
B -= 4
A += 1
B += 2
A -= 4
B -= 5
A u B
B u A
A ^ B
B ^ A
A \ B
B \ A
A -= 1
A c B
A -= 0
A c B
B = {}
B c A
A c B
A -= 2
A c B
B c A
A \ B u B \ A
q
```

Saída

```
A = {}
B = {}
verdadeiro
verdadeiro
A = {1, 2, 3}
B = {2, 3, 4}
falso
```

```
falso
verdadeiro
falso
falso
verdadeiro
A = {0, 1, 2, 3}
B = {1, 2, 3, 4}
A = {0, 1, 2}
B = {1, 2, 3}
A = {0, 1, 2}
B = {1, 2, 3}
A = {0, 1, 2}
B = {1, 2, 3}
A u B = {0, 1, 2, 3}
B u A = {0, 1, 2, 3}
A ^ B = {1, 2}
B ^ A = {1, 2}
A \ B = {0}
B \ A = {3}
A = {0, 2}
falso
A = {2}
verdadeiro
B = {}
verdadeiro
falso
A = {}
verdadeiro
verdadeiro
A \ B u B \ A = {}
```

Para mais exemplos, consulte os testes abertos no Susy.

Observações

- Você **não deve** submeter o arquivo `lab16_main.py` para o *SuSy*, somente o arquivo `lab16.py` .
- O número máximo de submissões é **10**.
- Para a realização dos testes do SuSy, a execução do código em Python se dará da seguinte forma: (Linux e OSX)
  `python3 lab16_main.py` .
- Você deve incluir, no início do seu programa, uma breve descrição dos objetivos do programa, da entrada e da saída, além do seu nome e do seu RA.
- Indente corretamente o seu código e inclua comentários no decorrer do seu programa.