Dicionários e conjuntos. Soluções aos exercícios

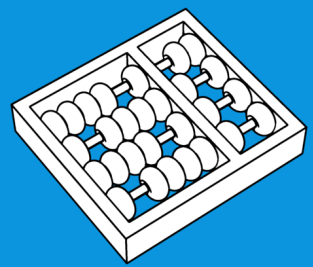
MC102 - Algoritmos e Programação de Computadores

04/25

14

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP



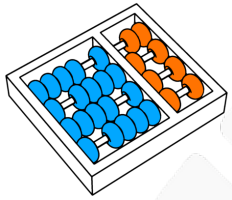
Al-go-rithm
(noun)

word used by programmers when they do not want to explain what they did.

Pro-gram-mer
(noun)

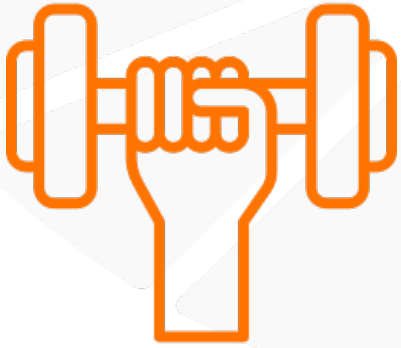
someone who solves a problem you didn't know you had in a way you don't understand.

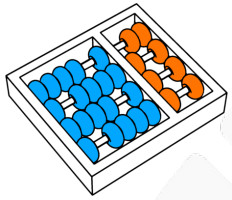
Exercícios



Dicionários e conjuntos

Soluções para os exercícios!

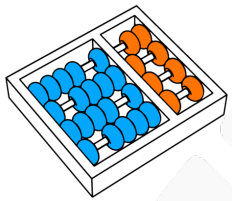




Exercícios

Faça os seguintes exercícios sem uso de dicionário e depois com uso de dicionários.

1. Leia uma quantidade **n** de pares de nome/RA de alunos. Faça um programa que permita buscar o RA de um aluno a partir do seu nome.
2. Faça um programa, que dado uma lista de número inteiros em $[0, 10)$, imprime um histograma para tais números.
3. Como modificar o programa anterior para fazer um histograma de nomes de alunos?



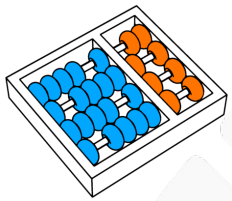
Soluções sem dicionários

Busca de RA por nome:

```
1  n = int(input())
2  nomes = []
3  ras = []
4  for i in range(n):
5      print('Indique o nome e RA do aluno', i+1)
6      linha = input().split()
7      nomes[i] = linha[0]
8      ras[i] = linha[1]
9  print('Indique o nome do aluno para encontrar o RA')
10 nome = input()
11 for i in range(n):
12     if nome == nomes[i]:
13         print('O RA de', nome, 'é', ras[i])
```

Histograma de números:

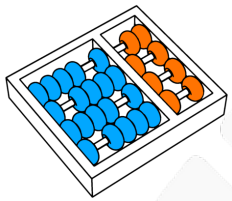
```
1  lista = list(map(int, input().split()))
2  histograma = [0] * 10
3  for numero in lista:
4      histograma[numero] += 1
5  for i in range(10):
6      print(i,':', '*' * histograma[i])
```



Soluções sem dicionários. Continuação

Histograma de alunos:

```
1  lista = input().split())
2  histograma = []
3  nomes = []
4  for nome in lista:
5      i = 0
6      # procura a posição do nome na lista, se não estiver a posição fica == len
7      while i < len(nomes) and nomes[i] != nome:
8          i += 1
9      if i < len(nomes):
10         histograma[i] += 1
11     else:
12         nomes.append(nome)
13         histograma.append(1)
14 for i in range(len(nomes)):
15     print(nomes[i],':', '*' * histograma[i])
```



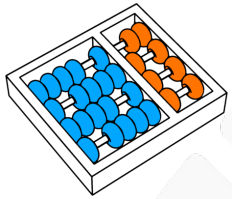
Soluções com dicionários

Busca de RA por nome:

```
1  n = int(input())
2  nomes_ras = {}
3  for i in range(n):
4      print('Indique o nome e RA do aluno', i+1)
5      linha = input().split()
6      nomes_ras[linha[0]] = linha[1]
7  print('Indique o nome do aluno para encontrar o RA')
8  nome = input()
9  print('O RA de', nome, 'é', nomes_ras[nome])
```

Histograma de números:

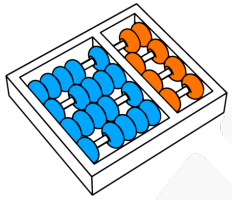
```
1  lista = list(map(int, input().split()))
2  histograma = {0:0, 1:0, 2:0, 3:0, 4:0, 5:0, 6:0, 7:0, 8:0, 9:0}
3  for numero in lista:
4      histograma[numero] += 1
5  for i in range(10):
6      print(i,':', '*' * histograma[i])
```



Soluções com dicionários. Continuação

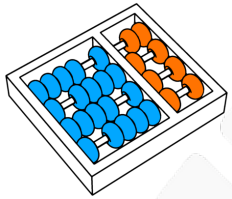
Histograma de alunos:

```
1  lista = input().split())
2  histograma = {}
3  for nome in lista:
4      if nome in histograma:
5          histograma[nome] += 1
6      else:
7          histograma[nome] = 1
8  for nome, hist in histograma.items():
9      print(nome,':', '*' * hist)
```



Exercícios para implementar operações de conjuntos

1. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a união de **s1** e **s2**.
2. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a interseção de **s1** e **s2**.
3. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a subtração de **s2** em **s1**.
4. Faça uma função que, dados conjuntos **s1** e **s2**, devolve se **s1** é subconjunto de **s2**.
5. Faça uma função que, dada uma lista, devolve uma nova lista sem elementos repetidos. Dica: use conjuntos!



Soluções

União:

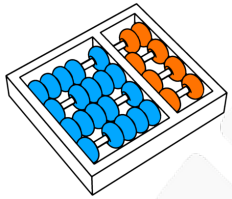
```
1  def uniao(s1, s2):
2      s3 = set()
3      for e in s1:
4          s3.add(e)
5      for e in s2:
6          s3.add(e)
7      return s3
```

Interseção:

```
1  def intersecao(s1, s2):
2      s3 = set()
3      for e in s1:
4          if e in s2:
5              s3.add(e)
6      return s3
```

Subtração:

```
1  def subtracao(s1, s2):
2      s3 = set()
3      for e in s1:
4          if e not in s2:
5              s3.add(e)
6      return s3
```



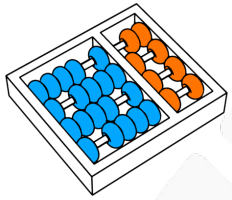
Soluções. Continuação

Subconjunto:

```
1 def subconjunto(s1, s2):
2     for e in s1:
3         if e not in s2:
4             return False
5     return True
```

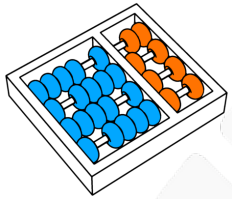
Sem repetições:

```
1 def sem_rep(lista):
2     return list(set(lista))
```



Exercícios de conjuntos

1. Faça um programa que encontre o mínimo e o máximo em um dado conjunto de números.
2. Faça uma função que dados dois conjuntos de números inteiros, retorna dois novos conjuntos: um com os dobros dos elementos repetidos e outro com a metade (inteira) dos elementos não repetidos.
3. Faça uma função que dados dois conjuntos, retorna um conjunto contendo qualquer valor que pode ser obtido como soma de um elemento do primeiro com um elemento do segundo.
4. Faça uma função que recebe um conjunto de números e retorna o maior produto entre dois números do conjunto.
5. Faça uma função que recebe um conjunto de inteiros **S** e um inteiro **x** e retorna um conjunto contendo todos os pares de valores de **S** que somados resultam em **x**.
6. Faça uma função que recebe um conjunto de strings e retorna o maior prefixo de todas as strings do conjunto.



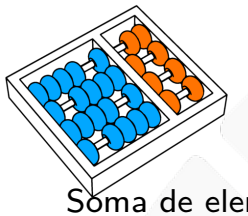
Soluções

Mínimo e máximo:

```
1  def min_max(s):
2      mini, maxi = None, None
3      for e in s:
4          if mini is None:
5              mini = e
6              maxi = e
7          elif mini > e:
8              mini = e
9          elif maxi < e:
10             maxi = e
11     return mini, maxi
```

Dobros e metades:

```
1  def dobros_metades(s1, s2):
2      dobros, metades = set(), set()
3      uniao = s1.union(s2)
4      for e in uniao:
5          if e in s1 and e in s2:
6              dobros.add(2 * e)
7          else:
8              metades.add(e // 2)
9      return dobros, metades
```



Soluções. Continuação

Soma de elementos:

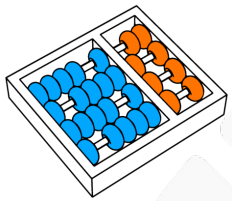
```
1  def soma(s1, s2):
2      s3 = set()
3      for e1 in s1:
4          for e2 in s2:
5              s3.add(e1 + e2)
6      return s3
```

Maior produto:

```
1  def maior_produto(s):
2      produto = None
3      for e1 in s:
4          for e2 in s:
5              if e1 != e2 and (produto == None or produto < e1 * e2):
6                  produto = e1 * e2
7      return produto
```

Pares que somam:

```
1  def pares_somam(s, x):
2      res = set()
3      for e1 in s:
4          for e2 in s:
5              if e1 < e2 and e1 + e2 == x:
6                  res.add(str(e1) + '+' + str(e2))
7      return res
```



Soluções. Continuação

Maior prefixo:

```
1  def maior_prefixo(strings):
2      prefixo = None
3      for s in strings:
4          if prefixo is None:
5              prefixo = s
6          else:
7              i = 0
8              # enquanto prefixo[:i] seja um prefixo válido de s aumentamos o i
9              while i < len(prefixo) and i < len(s) and prefixo[i] == s[i]:
10                 i += 1
11             prefixo = prefixo[:i]
12     return prefixo
```

Nesta solução, o maior prefixo comum deve ser prefixo de todas as strings, ou seja, deve ser prefixo da i-ésima string e também do maior prefixo comum das $i - 1$ strings anteriores. Assim, basta manter o maior prefixo comum até o momento e, a cada iteração, buscar o maior prefixo comum entre ele e a string atual.

Dicionários e conjuntos. Soluções aos exercícios

MC102 - Algoritmos e Programação de Computadores

04/25

14

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP