EP08

Disponível a partir de: quinta, 17 jun 2021, 00:01
Data de entrega: quarta, 23 jun 2021, 23:59
Arquivos requeridos: amigos.py (Baixar)
Tipo de trabalho: Trabalho individual
Redução por avaliação automática: 0.5 **Avaliações livres**: 15

EP08 - Amigo secreto

Fonte: https://www.pinterest.com/pin/369154500700563398

O **limite de submissões livres** deste EP é 15.

O **prazo de entrega** deste EP é 23h 59m do dia 23/06/2021. O sistema reabre a partir do dia 24/06 às 12h para envio de EPs com atraso, por mais 4 dias, recebendo desconto de 2 pontos por dia.

Sugerimos que depois de escrever e testar cada função você submeta seu EP para avaliação.

Objetivos

Como sempre, praticar o raciocínio aplicado a resolução de problemas computacionais... e agora com o uso de

- Listas em Python
- Funções com listas

Descrição

Ao planejar uma festa de final de ano, um grupo de `N` pessoas decidiu promover um *amigo(a) secreto(a)*. Nessa brincadeira, cada pessoa é identificada por um número (digamos de `0` a `N-1`), e recebe (por sorteio, de forma aleatória) o número de outra pessoa para dar um presente. Considere a lista em Python `amigo_de[]` que representa essa informação. Por exemplo:

```
                              +---+---+---+---+---+
conteúdo da lista amigo_de  --> | 3 | 2 | 1 | 4 | 0 |
                              +---+---+---+---+---+
                        índice    0   1   2   3   4
```

Essa lista revela que o `amigo_de[0]` é a pessoa `3`, o `amigo_de[1]` é a pessoa `2`, e assim por diante.

Durante a festa, a entrega dos presentes se inicia por uma pessoa qualquer, vamos assumir que seja a pessoa `0`.

A entrega dos presentes ocorre então da seguinte forma:

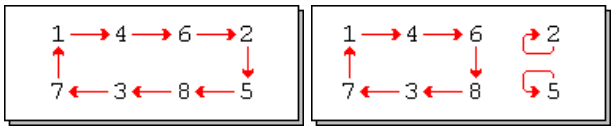
- a pessoa `0` dá seu presente para a pessoa `amigo_de[0]`, no caso para a pessoa `3`;
- a pessoa `amigo_de[0]` dá seu presente para a pessoa `amigo_de[amigo_de[0]]` (`amigo_de[3]` é `4`);
- a pessoa `amigo_de[amigo_de[0]]` dá seu presente para a pessoa `amigo_de[amigo_de[amigo_de[0]]]` (`amigo_de[4]` é `0`), e assim por diante.

Nesse exemplo, após a pessoa `0` receber seu presente, sobraram duas pessoas que ainda precisam participar. Uma dessas pode ser sorteada e o processo se repete com essa pessoa no papel da pessoa `0`.

Vamos ver um outro exemplo, para `N = 6` e a lista `amigo_de = [ 3, 0, 4, 2, 5, 1 ]`. Então

- a pessoa `0` dá seu presente para a pessoa `amigo_de[0] = 3`;
- a pessoa `3` dá seu presente para a pessoa `amigo_de[3] = 2`;
- a pessoa `2` dá seu presente para a pessoa `amigo_de[2] = 4`;
- a pessoa `4` dá seu presente para a pessoa `amigo_de[4] = 5`;
- a pessoa `5` dá seu presente para a pessoa `amigo_de[5] = 1`;
- finalmente, a pessoa `1` dá seu presente para a pessoa `amigo_de[1] = 0` e o processo termina.

Essas pessoas ficaram surpresas (:O_O) ao perceber que a pessoa `0`, que deu seu presente no início, recebeu seu presente somente no final, depois de cada pessoa ter ganhado seu presente.
Esse fenômeno acontece apenas quando a lista `amigo_de[]` é **circular**, ou seja, forma-se um **único ciclo** de pessoas, diferente do primeiro exemplo onde ocorreram dois ciclos. A figura mostra alguns exemplos da Wikipedia.



Fonte: Diagramas ilustrando uma lista *círcular* e uma *não circular* (Wikipedia)

A surpresa foi tão grande que as pessoas resolveram investigar esse fenômeno mais a fundo. Decidiram estimar a probabilidade de acontecer tal "fenômeno *S*" ("*S*" de "surpreendente") ocorrer. Essa probabilidade será denotada por *p*(*N*), pois deve depender do número `N` de amig@s. De fato, como observaremos na reflexão a seguir, a probabilidade *p*(*N*) depende do número `N` de pessoas.

Para `N` = 1 (sim, sabemos que parece não fazer sentido um amigo secreto solitário) temos que a única lista `amigo_de[]` possível é `[0]`, que é circular. Nesse caso, a probabilidade do fenômeno *S* ocorrer é 1.

Quando temos `N` = 2 pessoas no grupo há duas lista `amigo_de[]` possíveis. Essas listas são `[0, 1]` e `[1, 0]`. Nesse caso, apenas a lista `[1, 0]` é circular. Portanto, *p*(2) = 1/2 = 0.5.

No caso de `N` = 3 pessoas no grupo temos 3! = 6 listas `amigo_de[]` possíveis, mas apenas `[1, 2, 0]` e `[2, 0, 1]` são circulares. Logo, *p*(3) = 2/6 ≈ 0.333333.

De maneira similar, para `N = 4` temos 4! = 24 listas `amigo_de[]` possíveis e as únicas circulares são

`[1, 2, 3, 0]`, `[1, 3, 0, 2]`, `[2, 3, 1, 0]`, `[2, 0, 3, 1]`, `[3, 2, 0, 1]` e `[3, 0 , 1, 2]`.

Assim, temos que *p*(4) = 6/24 = 0.25.

Ideia para você implementar

Talvez você já esteja pensando em escrever uma fórmula para calcular a probabilidade. Mas espere, não faça isso ainda. Nesse exercício, ao invés de escrever uma fórmula, você deve aplicar a técnica conhecida pelo nome de Método de Monte Carlo.

Essa é uma técnica bastante poderosa e que poderá lhe ajudar no futuro a entender o comportamento de outros fenômenos além do `amigo secreto` que estamos usando de exemplo nesse exercício. Essa técnica é tão legal que pode lhe ajudar também a comprovar a corretude de uma fórmula analítica sua.

A ideia da técnica é simular, por meio do computador, vários sorteios e medir o resultado. Assim, para cada grupo de `N` amigos, vamos simular vários sorteios (digamos `T` tentativas), gerando `T` listas aleatórias `amigo_de[]` e verificando se cada lista é circular ou não. Digamos que, das `T` tentativas, observamos `S` surpresas, ou seja, `S` listas circulares. Então a razão `S/T` corresponde à estimativa da probabilidade *p*(*N*).

O que você deve fazer

Você deve completar o programa no arquivo `amigos.py` que estima a probabilidade *p*(*N*) para vários valores de `N`.

O arquivo `amigos.py` contém uma função `main()` para lhe ajudar com os testes. Há ainda uma função `sorteie_amigos()` que cria e retorna uma lista *embaralhada*/aleatória. Essa função é fornecida e você pode utilizar como desejar para desenvolver a sua solução.

Vamos dizer que, para vários valores de `N`, a função `main()` realiza um *experimento* a fim de estimar as probabilidades *p*(*N*). Cada experimento consiste de `T` *tentativas*. Em cada tentativa é sorteada uma lista embaralhada de comprimento `N` contendo os valores `0`, `1`, ..., `N-1`.

Você deve escrever duas funções: `circular()` e `experimento()`, como descritas a seguir.

- `circular(amigo_de)`: *recebe* uma lista `amigo_de[]` e *retorna* `True` se a lista é circular e *retorna* `False` em caso contrário. Uma lista é circular se permite que os presentes sejam distribuídos em um único ciclo. Como vimos nos exemplos para `N` = 6 a lista `[ 3, 0, 4, 2, 5, 1 ]` **é** circular e no exemplo para `N` = 5 a lista `[ 3, 2, 1, 4, 0 ]` **não** é circular.

- `experimento(N, T, debug=False)`: recebe dois inteiros, `N` e `T`, e um booleano `debug`. A função realiza `T` sorteios de listas aleatórias de tamanho `N` (número de pessoas no grupo) contendo os inteiros `0`, `1`, ..., `N-1`. A função retorna uma estimativa para probabilidade *p*(*N*) de uma lista de tamanho `N` ser circular. Essa estimativa é razão `S/T` entre o número observado de listas circulares e o número `T` de listas sorteadas. Assim, por exemplo, se `T` = 100 e observamos que dentre as `T` listas sorteadas há *S* = 15 que são circulares, então a estimativa para probabilidade será `S/T = 15/100 = 0.15`.

Quando o parâmetro `debug` é `True`, a função deve imprimir cada uma das listas circulares obtidas durante a realização do experimento. Cada lista deve ser precedida do número de listas circulares observadas até o momento. Veja os exemplos mais adiante.

Edite as funções `main()` e `sorteie_amigos()` do esqueleto

As função `main()` e `sorteie_amigos()` estão disponíveis no arquivo `amigos.py`. No entanto, elas contêm erros ☹ Assim, essas funções devem ser **consertadas**, como descrito abaixo, antes que possam ser utilizadas.

- `main()`: a tabulação de suas linhas foi *removida*. Portanto, antes de utilizá-la, você deverá corrigir essa tabulação.

- `sorteie_amigos(N)`: *cria* e *retorna* uma lista de tamanho `N` contendo uma permutação dos números `0`, `1`, ..., `N-1`. A tabulação de suas linhas foi *removida* e lgumas linhas foram *trocadas de ordem*. Você deve colocar a linhas dessa função na ordem certa e corrigir a tabulação.

Não altere nada além da tabulação da função `main()`. Isso deve fortalecer a sua compreensão sobre *blocos de comandos* (= trecho de código depois de `:`) em `while`s, `if`s, `elif`s, `def`s, etc.

Não altere nada além da ordem das linhas e a tabulação da função `sorteie_amigos()`. Isso deve fortalecer a sua compreensão sobre a *ordem e blocos de comandos*.

Para fazer esses consertos, examine o comportamento dessas funções nos exemplos a seguir.

Exemplos

A seguir estão alguns exemplos de execução da função `main()`. Os valores em **vermelho** foram digitados pelo usuário.

Exemplo 1

Comece testando seu programa com exemplos simples, pequenos, e confira se os resultados das probabilidades estão corretos. As linhas numeradas de 1 a 4, antes de *p*(2) e de *p*(3), contém as listas circulares. As linhas com esses listas foram produzidas pela função `experimento()` quando o argumento de `debug` é `True`. Estude o código da função `main()`.

```
INÍCIO DOS TESTES
Função circular()
circular([1, 2, 3, 4, 0]) = True
circular([1, 2, 0, 4, 3]) = False
Função experimento()
Digite o valor da semente do gerador de números pseudo-aleatórios: 1
Qual o número mínimo de pessoas: 2
Qual o número máximo de pessoas: 5
Qual o passo: 1
Qual o número de tentativas em cada experimento: 10
Você quer ver as listas que são circulares [s/n]: s
1 : [1, 0]
2 : [1, 0]
3 : [1, 0]
4 : [1, 0]
5 : [1, 0]
p(2) = 0.5
1 : [2, 0, 1]
2 : [2, 0, 1]
3 : [2, 0, 1]
4 : [1, 2, 0]
5 : [2, 0, 1]
p(3) = 0.5
1 : [2, 0, 3, 1]
2 : [2, 3, 1, 0]
p(4) = 0.2
1 : [3, 2, 0, 4, 1]
p(5) = 0.1
TESTES ENCERRADOS
```

Exemplo 2

O Exemplo 2 usa os mesmos argumentos que foram digitados pelo usuário no Exemplo 1. Observe que, usando uma semente diferente, as listas circulares sorteadas podem ser diferentes. As estimativas para as probabilidades também podem ser diferentes. Execute em seu computador o programa usando a mesma semente e verifique que os resultados são sempre os mesmos. Modifique a semente para obter outros valores sorteados, como nesse exemplo.

```
INÍCIO DOS TESTES
Função circular()
circular([1, 2, 3, 4, 0]) = True
circular([1, 2, 0, 4, 3]) = False
Função experimento()
Digite o valor da semente do gerador de números pseudo-aleatórios: 21
Qual o número mínimo de pessoas: 2
Qual o número máximo de pessoas: 5
Qual o passo: 1
Qual o número de tentativas em cada experimento: 10
Você quer ver as listas que são circulares [s/n]: s
1 : [1, 0]
2 : [1, 0]
3 : [1, 0]
4 : [1, 0]
5 : [1, 0]
p(2) = 0.5
1 : [1, 2, 0]
2 : [1, 2, 0]
p(3) = 0.2
1 : [1, 3, 0, 2]
2 : [1, 2, 3, 0]
3 : [3, 2, 0, 1]
p(4) = 0.3
1 : [4, 2, 3, 0, 1]
2 : [2, 4, 3, 1, 0]
p(5) = 0.2
TESTES ENCERRADOS
```

Exemplo 3

Exemplo que não mostra as listas com um ciclo.

```
INÍCIO DOS TESTES
Função circular()
circular([1, 2, 3, 4, 0]) = True
circular([1, 2, 0, 4, 3]) = False
Função experimento()
Digite o valor da semente do gerador de números pseudo-aleatórios: 23
Qual o número mínimo de pessoas: 5
Qual o número máximo de pessoas: 100
Qual o passo: 5
Qual o número de tentativas em cada experimento: 1000
Você quer ver as listas que são circulares [s/n]: n
p(5) = 0.195
p(10) = 0.106
p(15) = 0.07
p(20) = 0.055
p(25) = 0.037
p(30) = 0.036
p(35) = 0.029
p(40) = 0.012
p(45) = 0.026
p(50) = 0.02
p(55) = 0.018
p(60) = 0.013
p(65) = 0.015
p(70) = 0.018
p(75) = 0.009
p(80) = 0.007
p(85) = 0.008
p(90) = 0.013
p(95) = 0.01
p(100) = 0.008
TESTES ENCERRADOS
```

Exemplo 4

Usa quase os mesmos argumentos que o Exemplo 3, mas o número de tentativas `T` é bem maior.

```
INÍCIO DOS TESTES
Função circular()
circular([1, 2, 3, 4, 0]) = True
circular([1, 2, 0, 4, 3]) = False
Função experimento()
Digite o valor da semente do gerador de números pseudo-aleatórios: 23
Qual o número mínimo de pessoas: 5
Qual o número máximo de pessoas: 100
Qual o passo: 5
Qual o número de tentativas em cada experimento: 10000
Você quer ver as listas que são circulares [s/n]: n
p(5) = 0.205
p(10) = 0.103
p(15) = 0.0724
p(20) = 0.0474
p(25) = 0.0404
p(30) = 0.0328
p(35) = 0.0284
p(40) = 0.0247
p(45) = 0.0219
p(50) = 0.0198
p(55) = 0.0178
p(60) = 0.0161
p(65) = 0.0161
p(70) = 0.0152
p(75) = 0.013
p(80) = 0.0133
p(85) = 0.0111
p(90) = 0.0134
p(95) = 0.0102
p(100) = 0.0104
TESTES ENCERRADOS
```

Roteiro

Os passos a seguir devem ajudar você a concluir essa tarefa mais rapidamente.

- Baixe o arquivo `amigos.py` para uma pasta do seu computador. Este é o único arquivo que deverá ser depositado nesta página.
- Conserte e estude a função `main()`. Edite a função para que ela tenha o comportamento exibido nos exemplos fornecidos no enunciado. Na `main()` apenas a tabulação foi removida.
- Conserte e estude a função `sorteie_amigos()`. Edite a função para que ela tenha o comportamento exibido nos exemplos fornecidos no enunciado. Na função `sorteie_amigos()` além da tabulação a ordem de algumas linhas foi trocada. Utilize o Python Shell e teste apenas esta função.
  Submeta seu EP para avaliação desta função. Verifique as mensagem do programa `avaliador`. Lembre-se que, para esse exercício, você tem 15 submissões livres.
- Implemente a função `circular(amigo_de)`. Utilize o Python Shell e teste apenas esta função.
  Submeta seu EP para avaliação desta função. Verifique as mensagem do programa `avaliador`.

- Implemente a função `experimento(N, T, debug=False)`. Em seguida, utilizando o Python Shell, teste apenas essa função. Teste-a com `True` como terceiro argumento. Submeta seu EP para avaliação desta função.
  Verifique as mensagem do programa `avaliador`.

- Explore o programa executando-o várias vezes para entender o fenômeno do problema de gerar uma lista de amigo secreto com apenas um ciclo. Por exemplo, use intervalos, passos e `T`s diferentes. Tente formar uma conjectura sobre qual deve ser o valor de $p(N)$ para um dado N. Por exemplo, para um grupo de 10 pessoas, seria mais ou menos provável de encontrarmos uma lista circular que para um grupo de 100 pessoas?

- Observe que, por ter um comportamento aleatório, as probabilidades obtidas em cada execução, mesmo usando os mesmos parâmetros, podem ser diferentes. Inclusive os seus resultados podem ser diferentes dos exemplos no enunciado!.

- Execute o programa `main()` com a opção `s` para ver as listas com um ciclo. Confira seus resultados.

- Deposite o seu EP nesta página. Verifique as mensagem do programa `avaliador`. Caso necessário faça eventuais correções e ressubmeta o seu EP.

Honestidade Acadêmica

Esse é um exercício individual, não em grupo. Isso não significa que você não pode receber ajuda de outras pessoas, inclusive de seus colegas. De uma forma geral, gostaríamos de incentivar as discussões de ideias, conceitos e alternativas de solução. Nossa maior recomendação é evitar olhar o código fonte de uma solução antes de escrever o seu programa. Em caso de dúvida, consulte nossa política de colaboração.

De forma sucinta, evite as seguintes ações que caracterizam desonestidade acadêmica na realização dos trabalhos individuais desta disciplina:

- buscar e obter uma solução, parcial ou completa, correta ou não, de um EP na internet ou qualquer outro meio físico ou virtual, durante o período de submissão do referido EP;
- solicitar ou obter uma cópia, parcial ou completa, correta ou não, da solução de um EP durante o seu período de submissão;
- permitir que um colega acesse uma cópia, parcial ou completa, correta ou não, do seu EP, durante o período de submissão;
- ainda mais grave é o plágio, que se configura pela utilização de qualquer material não visto em aula ou não descrito no enunciado, que não seja de sua autoria, em parte ou ao todo, e entregar, com ou sem edição, como se fosse seu trabalho, para ser avaliado.

Arquivos requeridos

amigos.py

```python
# -*- coding: utf-8 -*-
#---------------------------------------------------------------------
# LEIA E PREENCHA O CABEÇALHO
# NÃO ALTERE OS NOMES DAS FUNÇÕES, MÉTODOS E ATRIBUTOS
#---------------------------------------------------------------------

'''
    Nome:
    NUSP:

    Ao preencher esse cabeçalho com o meu nome e o meu número USP,
    declaro que todas as partes originais desse exercício programa (EP)
    foram desenvolvidas e implementadas por mim e que portanto não
    constituem desonestidade acadêmica ou plágio.
    Declaro também que sou responsável por todas as cópias desse
    programa e que não distribui ou facilitei a sua distribuição.
    Estou ciente que os casos de plágio e desonestidade acadêmica
    serão tratados segundo os critérios divulgados na página da
    disciplina.
    Entendo que EPs sem assinatura devem receber nota zero e, ainda
    assim, poderão ser punidos por desonestidade acadêmica.

    Abaixo descreva qualquer ajuda que você recebeu para fazer este
    EP.  Inclua qualquer ajuda recebida por pessoas (inclusive
    monitores e colegas). Com exceção de material de MAC0110, caso
    você tenha utilizado alguma informação, trecho de código,...
    indique esse fato abaixo para que o seu programa não seja
    considerado plágio ou irregular.

    Exemplo:

        A monitora me explicou que eu devia utilizar a função int() quando
        fazemos leitura de números inteiros.

        A minha função quicksort() foi baseada na descrição encontrada na
        página https://www.ime.usp.br/~pf/algoritmos/aulas/quick.html.

    Descrição de ajuda ou indicação de fonte:

'''
########################################################################
# MÓDULOS A SEREM UTILIZADOS NO PROGRAMA
# random.shuffle(lst) embaralha os elementos da lista lst.
import random

########################################################################
def main():
'''
Essa função auxilia no teste das funções pedidas para o EP08.
Se desejar, escreva mais testes.

Atenção: a tabulação das linhas foi removida e deve ser consertada
antes que a função possa ser utilizada.
'''
print("INÍCIO DOS TESTES")

# testes da função circular()
print("Função circular()")
amigos1 = [1,2,3,4,0]
amigos2 = [1,2,0,4,3]
print(f"circular({amigos1}) = {circular(amigos1)}")
print(f"circular({amigos2}) = {circular(amigos2)}")

# testes da função experimento()
print("Função experimento()")
semente = int(input("Digite o valor da semente do gerador de números pseudo-aleatórios: "))
random.seed(semente)
MINN  = int(input("Qual o número mínimo de pessoas: "))
MAXN  = int(input("Qual o número máximo de pessoas: "))
passo = int(input("Qual o passo: "))
T     = int(input("Qual o número de tentativas em cada experimento: "))
SHOW  = input("Você quer ver as listas que são circulares [s/n]: ")

debug = False
if SHOW == 'S' or SHOW == 's':
debug = True

N = MINN
while N <= MAXN:
pN = experimento(N, T, debug)
print(f"p({N}) = {pN}")
N = N + passo

print("TESTES ENCERRADOS")

########################################################################
def sorteie_amigos( N ):
''' (int) -> list
RECEBE um inteiro N > 0.
RETORNA uma lista de tamanho N, contendo os números de 0 a N em
    ordem aleatória.

ATENÇÃO: a tabulação das linhas foi removida e a ordem de algumas
```

```
ATENÇÃO: a tabulação das linhas foi removida e a ordem de algumas
linhas alterada. Ela deve ser consertada antes possa ser utilizada.
'''
i += 1
amigos = []
while i < N:
return amigos
amigos += [i]
random.shuffle(amigos)
i = 0

######################################################################
def circular(amigo_de):
    ''' (list) -> bool
    RECEBE uma lista amigo_de (de "amigos secretos")
    RETORNA True se a lista for circular e False em caso contrário.
    '''
    # modifique o código abaixo para conter a sua solução.
    print("Vixe!! ainda não fiz a função circular()")
    return True

######################################################################
def experimento(N, T, debug=False):
    ''' (int, int) -> float
    RECEBE um inteiro N > 0, um inteiro T > 0 e um booleano debug.
    RETORNA a probabilidade de uma lista de "amigo secretos" com
        N participantes ser circular.

    Esta probabilidade deve ser calculada a partir de T sorteios
    de listas de tamanho N, e calculando a frequência das listas
    circulares.

    Se a opção debug é True a função deve imprimir todas as
    listas sortedas que forem circulares, como mostrado no
    enunciado.
    '''
    # modifique o código abaixo para conter a sua solução.
    print("Vixe!! ainda não fiz a função experimento()")
    return 0.0

#=====================================================================
# Não modifique as linhas abaixo
# Esse if serve para executar a main() dentro do Spyder
# e não atrapalhar o avaliador
if __name__ == '__main__':
    main()

```

VPL

◄ mais uma provinha teste

Seguir para...

EP07 ►