Processo Seletivo Turing USP - 2022

Case Técnico

Pontuação Total: 100 pontos

Olá, candidate! Seja muito bem-vinde ao processo seletivo Turing USP 2022! Nessa fase do nosso processo seletivo, você terá que resolver um case técnico. São várias perguntas, incluindo um texto em formato de Turing Talks e dois desafios. Envios incompletos serão considerados, mas encorajamos que tente fazer o máximo possível. Os **desafios valem uma pontuação extra** e você pode escolher se quer fazer ou não, mas **deve escolher apenas um dos dois**. Se fizer os dois vamos corrigir apenas um, portanto veja qual faz mais sentido para você, se achar que vale a pena fazer o desafio. Algumas questões são propositalmente desafiadoras, então saiba o que atacar, padawan. O intuito aqui é entender o seu raciocínio e qualidade de código, bem como o seu desenvolvimento no processo.

Lembrete: isso não é uma prova da USP, **sinta-se livre para procurar guias, vídeos e outros recursos na internet para te ajudar**.

Caso tenha alguma dúvida, é só perguntar para a gente no nosso Discord, onde temos canais específicos para cada parte do case. Se preferir, você pode também entrar em contato pelo nosso e-mail turing.usp@gmail.com, pela nossa página facebook.com/turing.usp ou mandar uma mensagem para algum dos membros que estão listados em cada parte do case.

Tentaremos responder o quanto antes! Além disso, **teremos monitorias para ajudá-los,** cujas datas serão divulgadas em nossas redes sociais, então fiquem atentes! Elas ocorrerão de maneira **online em nosso Discord e presencial na USP**.

Estrutura e Orientações

O case é dividido em três partes:

1. **Lógica de programação e algoritmos**
   Essa parte deve ser entregue em python.
   Para essa parte, uma primeira etapa da correção será feita automaticamente

por um programa de computador. Para evitar problemas durante a correção automática:

- É **proibida a utilização de acentos nas palavras na hora de escrever no seu código**;
- É obrigatório **utilizar os nomes das funções iguais aos nomes fornecidos nos enunciados.**

2. **Análise de dados e inteligência artificial**
   Essa parte deve ser entregue em jupyter notebooks escritos em python.
3. **Texto em formato de Turing Talks**
   Essa parte deve ser entregue em formato PDF.

Também é importante lembrar que quanto mais bem escrito o código melhor, pois facilitará o entendimento e a lógica por trás do mesmo. Neste sentido, se esforce para dar bons nomes de variáveis, priorize lógicas simples e se preciso adicione comentários; códigos ilegíveis podem resultar em perda de pontuação.

Indicação de Materiais

Caso você não tenha o costume de programar em python ou precise revisar algum conceito da linguagem, recomendamos alguns de nossos posts no medium e vídeos no youtube como apoio para os exercícios:

- Tutorial de python:
  - Parte 1: Programação | Python — Parte 1 | by Lucas Fentanes Machado | Turing Talks
  - Parte 2: Programação | Python — Parte 2 | by Fernando Matsumoto | Turing Talks

- Cursos gratuitos:
  - Curso de Python Básico do Turing USP: Python Básico - YouTube
  - Curso em vídeo de Python Básico: Curso Python #01
  - Introdução à programação (Python): MAC2166 em Python - YouTube
  - Introdução à Ciência da Computação com Python: Coursera - Parte 1
  - Introdução à Ciência da Computação com Python: Coursera - Parte 2

- Textos e livros:

- O Tutorial Python: python.org (Tutorial mais direto a sintaxe do python, recomendado para quem já tem uma base de lógica de programação)

Formato de Submissão

Quando terminar seu case, envie-o nesse forms. **Só aceitaremos uma única submissão**.

Você deve seguir o seguinte formato:

- As respostas dos exercícios de lógica de programação e algoritmos devem ser entregues em arquivos python (.py). Note que **esperamos um arquivo por questão/item**, com a questão devidamente identificada no nome do arquivo.
- As respostas dos exercícios de análise de dados e IA devem ser entregues em Jupyter Notebook (.ipynb), escritos em python (não será permitido o uso de R).
- O seu texto (Turing Talks) deve ser entregue em pdf.

Os nomes dos arquivos e pastas devem identificar as questões que eles resolvem, **conforme a estrutura abaixo** (só inclua as partes que você resolver, não precisam estar completas para mandá-las). A submissão deve ser feita por meio de um único arquivo zip com o nome "`nusp_nome_sobrenome.zip`".

```
12345678_alan_turing.zip
├ 1_logica
│ ├ q1a.py
│ ├ q1b.py
│ ├ q2a.py
│ ├ q2b.py
│ └ desafio.py
├ 2_analise-e-ia
│ └ analise_e_modelos.ipynb
└ 3_turing-talks
  └ turing-talks.pdf
```

Observações:

- **Recomendamos dar uma passada em todas as partes antes de começar a fazer o case para escolher o que você se sente mais confortável em fazer!** E não fique assustado com o tamanho dos enunciados, a maioria do conteúdo foi colocado justamente para ajudar na realização da tarefa.
- Como nas outras etapas, utilize nusp 0 (zero) se você for membro da USP, mas não tiver nusp.

Parte 1: Lógica de Programação e Algoritmos

40 pontos | Nível Intermediário

Caso tenha alguma dúvida nessa parte do case (seja de interpretação do enunciado ou alguma dúvida técnica), é só perguntar para a gente no nosso Discord (no canal **#duvidas-intermediario-logica**), no nosso e-mail turing.usp@gmail.com, na nossa página facebook.com/turing.usp ou mandar uma mensagem para algum dos membros da equipe abaixo:

Letícia Falconer	+55 (11) 99507-6247
Izabela Marina	+55 (91) 98361-0304
Guilherme Salustiano	+55 (41) 9928-7997

Essa parte do case tem **2 questões** (Q1, Q2) de temas diferentes, cada uma com **itens A e B**, além de uma questão **desafio**. No final de cada item e do desafio são dados exemplos de entrada e saída para que você teste os seus programas.

Algumas dicas gerais:

- Antes de começar a programar, leia os exemplos. Eles podem ajudar na compreensão dos exercícios.
- Depois de escrever os seus programas, é importante testá-los com os exemplos dados, mas lembre-se de que você também pode escrever os seus próprios exemplos.

Observação técnica: nessa parte do case, estamos considerando o **python 3** (qualquer versão 3.x). Essa é a versão utilizada pelo repl.it, então caso esteja utilizando essa plataforma, você não precisa se preocupar com isso.

Exemplo de Resolução

Cada item desta parte do case técnico pede que você defina uma função. Veja abaixo um exemplo de resolução mostrando como isso deve ser feito.

Item A

Escreva uma função que, dados dois números, **determine a soma deles**.

Formato de Entrada e Saída

As função deve ter a seguinte assinatura:

```
soma(a, b)
```

onde:

- a (um int) é o primeiro número.
- b (um int) é o segundo número.

A função soma deve retornar um int.

Exemplo

```
Input:  soma(1, 2)
Output: 3
```

Resolução

```
def soma(a, b):
    return a + b
```

Q1. Vacas magras no Turing

Formato: .py

Com a entrada dos membros ao Turing, a área de estratégia precisa enviar os e-mails de todos os membros para o professor Bufo-Chan cadastrá-los no Datacamp. Bufo-Chan é um professor muito antigo da USP e por isso não utiliza e-mail e a única forma de enviar a lista é por correio.

Infelizmente, como o processo de arrecadação de fundos do grupo ainda não começou, foi decidido utilizar o mínimo possível de tinta para imprimir a lista com emails. Um dos membros sugeriu a seguinte solução: alinhamos os e-mails à direita. A partir do segundo e-mail impresso, **os caracteres iniciais** do próximo e-mail a ser impresso que coincidirem com os do e-mail acima são omitidos, ficando apenas um espaço em branco.

Por exemplo para os emails `camila.lobianco@usp.br`, `camilalala.topp@usp.br`, `camila.doturing@usp.br`, `azank.pistolado@usp.br` e `camila_lobianco@usp.br` a impressão ficará da seguinte forma:

```
          c a m i l a . l o b i a n c o @ u s p . b r
                    l a l a . t o p p @ u s p . b r
                    . d o t u r i n g @ u s p . b r
          a z a n k . p i s t o l a d o @ u s p . b r
          c a m i l a _ l o b i a n c o @ u s p . b r
```

Pelo exemplo acima é possível perceber que foram economizadas 12 letras.

Item A (10 pontos)

Submissão: 1_logica/q1a.py

Você, entusiasmado por ter entrado no grupo, se propôs a fazer uma função *conta_economia(emails)* que recebe a lista de emails e retorna quantas letras foram economizadas. Neste item, você deve determinar quantos caracteres podem ser economizados **mantendo a ordem** dos emails.

Obs 1: Considere que todos os emails passados têm o mesmo tamanho.

Obs 2: As letras do domínio nunca serão economizadas, isto é, ao chegar no `@usp.br` você não deve considerar que tais letras podem ser omitidas. Considere também que os emails só possuem um "@" cada, sendo este aquele que inicia o domínio.

Referências do Item A:

- Vetores - IME-USP
- Ordenação de vetores

Formato de Entrada e Saída

Entrada

Sua entrada é composta de uma lista *emails* com N emails.

Saída

Seu programa deve *retornar* um número indicando quantos caracteres podem ser economizados para o conjunto N de e-mails.

Exemplos

Exemplo 1

Entrada do exemplo 1:

```
conta_economia([“camila.lobianco@usp.br”,
                “camilalala.topp@usp.br”,
                “azank.pistolado@usp.br”])
```

Saída do exemplo 1:

```
6
```

Exemplo 2

Entrada do exemplo 2:

```
conta_economia([“nelson.turing.usp@usp.br”,
                “noelson.turingusp@usp.br”,
                “noelson.estudausp@usp.br”,
                “matsumoto.membrot@usp.br”,
                “mateus.fakemembro@usp.br”])
```

Saída do exemplo 2:

```
12
```

Item B (10 pontos)

Submissão: 1_logica/q1b.py

Enquanto os membros do Turing estavam imprimindo a lista de e-mails, eles perceberam que a impressora foi infectada por um vírus e está imprimindo de forma incorreta. Depois de olhar para várias páginas impressas por um tempo, você percebe que ele está imprimindo cada linha de dentro para fora. Em outras palavras, a metade esquerda de cada linha está sendo impressa a partir do meio da página até a margem esquerda. De maneira similar, a metade direita de cada linha está sendo impressa a partir da margem direita e prosseguindo em direção ao centro da página.

Por exemplo, o e-mail `azank.pistolado@usp.br` está sendo impresso como `otsip.knazarb.psu@odal`. Isso ocorre pelo seguinte processo:

`azank.pistolado@usp.br`	original
`azank.pisto	lado@usp.br`	orig dividido
`otsip.knaza	rb.psu@odal`	div invertido
`otsip.knazarb.psu@odal`	invertido

Além disso, você percebeu que alguns e-mails também tiveram seus domínios embaralhados/errados após a impressão, inviabilizando o envio de tais e-mails ao professor. Um dos exemplos é o e-mail do membro Noel, que foi impresso como `irut.leonrb.pus@gn` e após a desinverte-lo temos `noel.turing@sup.br`. Perceba que nesse caso o domínio `@usp.br` está errado.

Sua tarefa nesta questão é criar a função ***corrige_emails(emails)***, *que* retorna uma lista com os e-mails corretamente formatados a partir da lista impressa ou ERRO `nas posições em` que o domínio esteja errado.

OBS: Caso o número de caracteres seja ímpar, considerar o piso da metade. Assim, a metade de 25 fica 12 ao invés de 12,5.

Formato de Entrada e Saída

Entrada

Sua entrada é composta de um array *emails* com N e-mails.

Saída

Seu programa deve printar 1 e-mail por linha corretamente formatado ou ERRO caso o domínio esteja errado.

Exemplos

Exemplo 1

Entrada do exemplo 1:

```
corrige_emails([“ibol.alimacrb.psu@ocna”,
                “t.alalalimacrb.repsu@ppo”,
                “.orbmem_ovonrb.psu@gnirut”])
```

Saída do exemplo 1:

```
[‘camila.lobianco@usp.br’,’ERRO’, ‘novo_membro.turing@usp.br’]
```

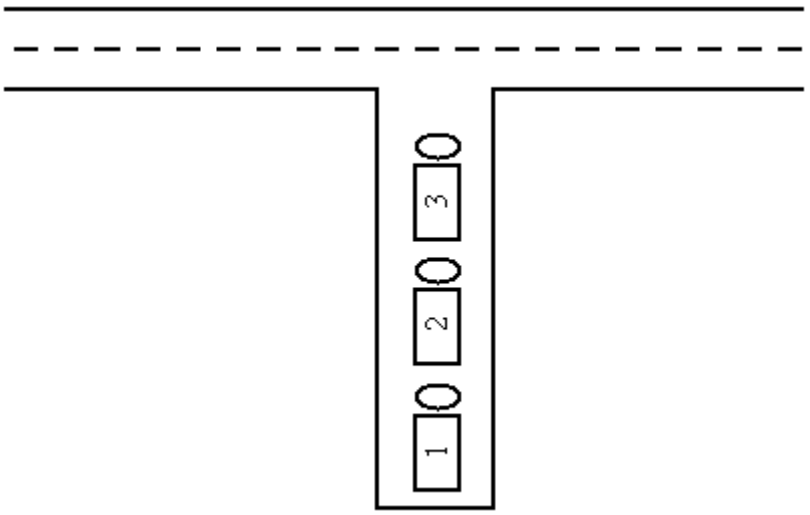
Q2. Estacionamento do Nelnelson

Formato: .py

Depois de muito tempo juntando dinheiro, Nelnelson, um dos membros mais exemplares do Turing, conseguiu juntar dinheiro para comprar seu primeiro carro (parcelado, é claro). Chega de pegar ônibus, agora sua vida será mais fácil. Pelo menos isso é o que ele pensava até lembrar do estacionamento do seu condomínio.

O estacionamento do condomínio é composto de apenas 1 corredor que comporta K carros um atrás do outro. Como este estacionamento só tem um portão, só é possível entrar e sair por ele.

Ao entrar com o primeiro carro, ele ocupa a posição próxima a parede (carro 1). O próximo carro fica à frente deste (carro 2) e assim por diante. A imagem a seguir demonstra um exemplo:



Obviamente, não é possível que um carro passe por cima de outro, portanto só é possível que um carro saia do estacionamento se ele for o último da fila (nesse exemplo o carro 3).

Referências gerais para a questão 2:

- Vetores - IME-USP
- Pilha - IME-USP
- Pilha em Python - Pandas IME USP
- Pilhas em Python com listas

Item A (10 pontos)

Submissão: 1_logica/q2a.py (.c, .cpp, .java)

Você, como um bom programador, quer ajudar Nelnelson a saber se todos os moradores do condomínio conseguiram estacionar seus carros. Dessa forma, você decide escrever uma função ***estacionamento_ok(K, instantes)*** que verifica se é possível que todos os carros consigam estacionar e sair sem problemas.

Dica: Utilize uma pilha para fazer a inserção e remoção dos carros no estacionamento.

Formato de Entrada e Saída

Entrada

Sua entrada é composta de K e de um vetor no qual cada elemento é a representação do que ocorreu nos instantes monitorados. Cada elemento do vetor será um valor C. Se C for positivo, o carro indicado pelo número está entrando no estacionamento. Se C for negativo, o carro indicado pelo número está saindo do estacionamento. O elemento no índice 0 do vetor indica a ação ocorrida no instante 0, o elemento no índice 1, a ação ocorrida no instante 1 e assim por diante. **Veja o exemplo 1 para maiores explicações**.

Saída

Você deve *retornar* `sim`, caso seja possível que todos os carros estacionem e saiam sem problemas e `nao`, caso contrário, i.e. ao primeiro sinal de problema ao entrar ou sair do estacionamento sua função deve retornar `nao`. Caso um carro não consiga entrar no estacionamento por ele já estar lotado, seu programa deve retornar `nao`. Todos os carros devem entrar **e** sair, portanto se o estacionamento não terminar vazio a sua função deve retornar `nao`.

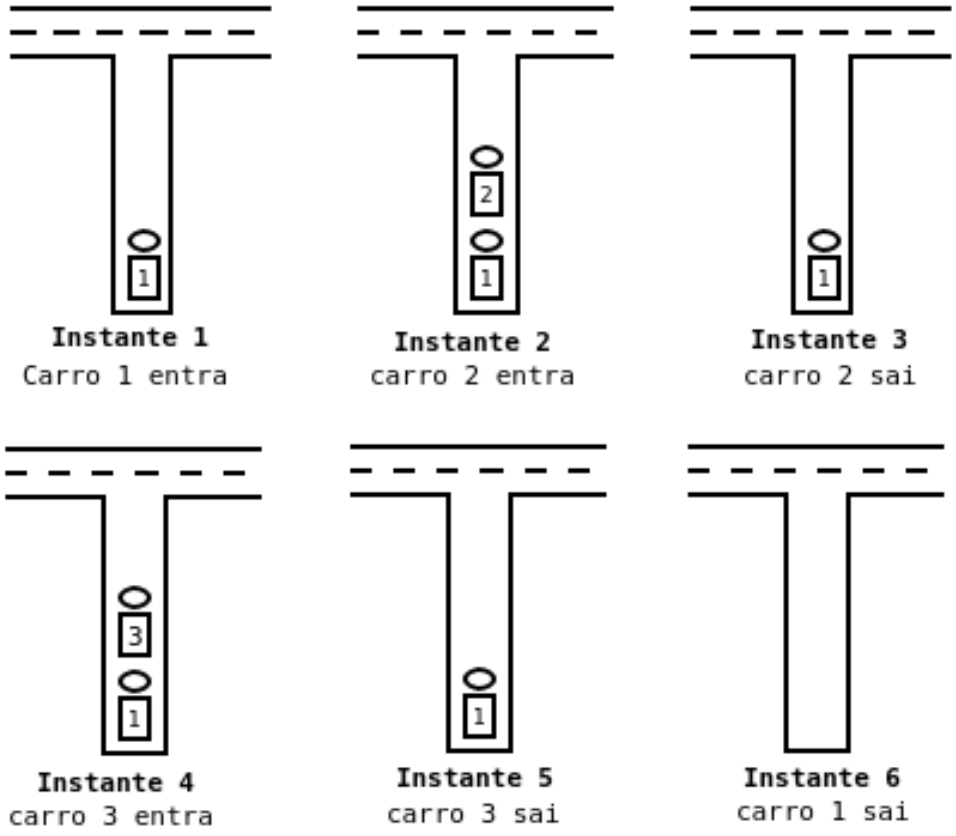
Exemplos

Exemplo 1

Entrada do exemplo 1:

```
estacionamento_ok(3,[1,2,-2,3,-3,-1])
```

No exemplo acima, quando temos `-3`, por exemplo, significa que o carro `3` está saindo do estacionamento. Veja:



Saída do exemplo 1:

```
sim
```

Exemplo 2

Entrada do exemplo 2:

```
estacionamento_ok(3,[1,2,-2,3,5,-3,-1,-5])
```

Saída do exemplo 2:

```
nao
```

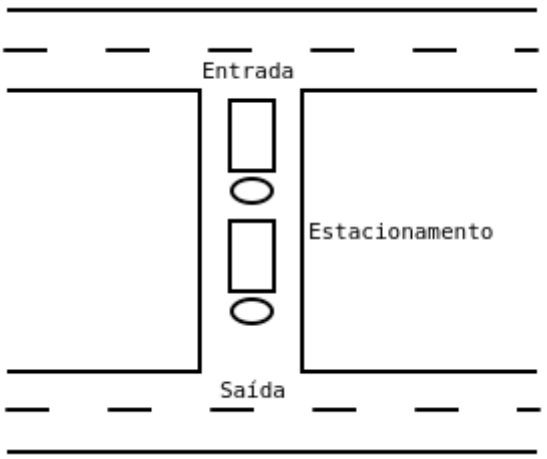
Observações para o item A:

- O usuário pode considerar que um carro que já esteja estacionado não tentará entrar novamente, a não ser que já tenha saído do estacionamento. De forma análoga, um carro que não está estacionado não tentará sair.

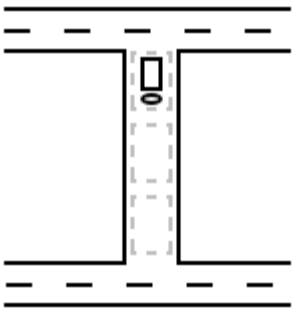
Item B (10 pontos)

Submissão: 1_logica/q2b.py (.c, .cpp, .java)

A rua em que Nelson irá guardar seu carro, que antes era sem saída em uma das extremidades, agora possui uma entrada e uma saída conforme a seguinte imagem mostra:

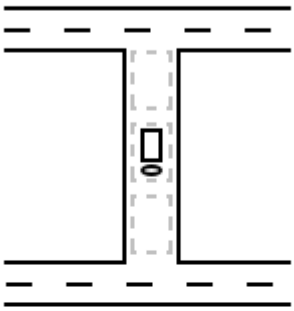


Este estacionamento funciona de uma maneira muito peculiar, pois a cada novo instante o carro avança uma posição dentro dele. Veja o exemplo para um estacionamento de tamanho K = 3 e um determinado carro:



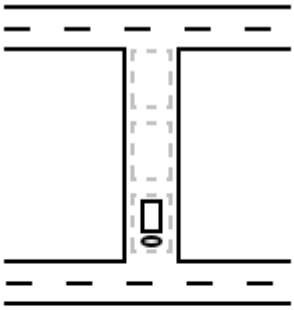
Instante 1:

O carro está no estacionamento



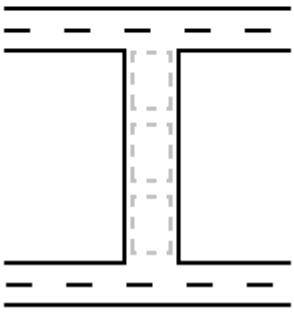
Instante 2:

O carro avança 1 posição no estacionamento



Instante 3:

O carro avança mais 1 posição no estacionamento



Instante 4:

O carro sai do estacionamento

Pelo exemplo, acima percebemos que uma vez que o carro entra no estacionamento, a cada instante ele avança uma posição até sair. Dito isso, você consegue ajudar Nelson a descobrir como está o estacionamento em um determinado instante? Crie a função ***estado_atual(K, horários, T)*** para resolver esse problema.

Formato de Entrada e Saída

Entrada

Sua entrada é composta por **K**, o tamanho do estacionamento; por um vetor ***entradas***, no qual cada elemento representa o horário de entrada E_n do carro associado àquela posição (a primeira posição do vetor representa o carro 1, a segunda posição o carro 2 e assim por diante); e por um instante **T**, no qual queremos saber status do estacionamento.

Saída

Sua saída deve ser um **vetor de K números** (tamanho do estacionamento), que indica como está o estacionamento no instante T. Caso uma posição do estacionamento não possua nenhum carro, você deverá *retornar* `0` naquela posição. Além disso, o primeiro número do vetor representa o carro na primeira posição do estacionamento, isto é, logo depois da entrada. Veja o exemplo a seguir para maiores explicações.

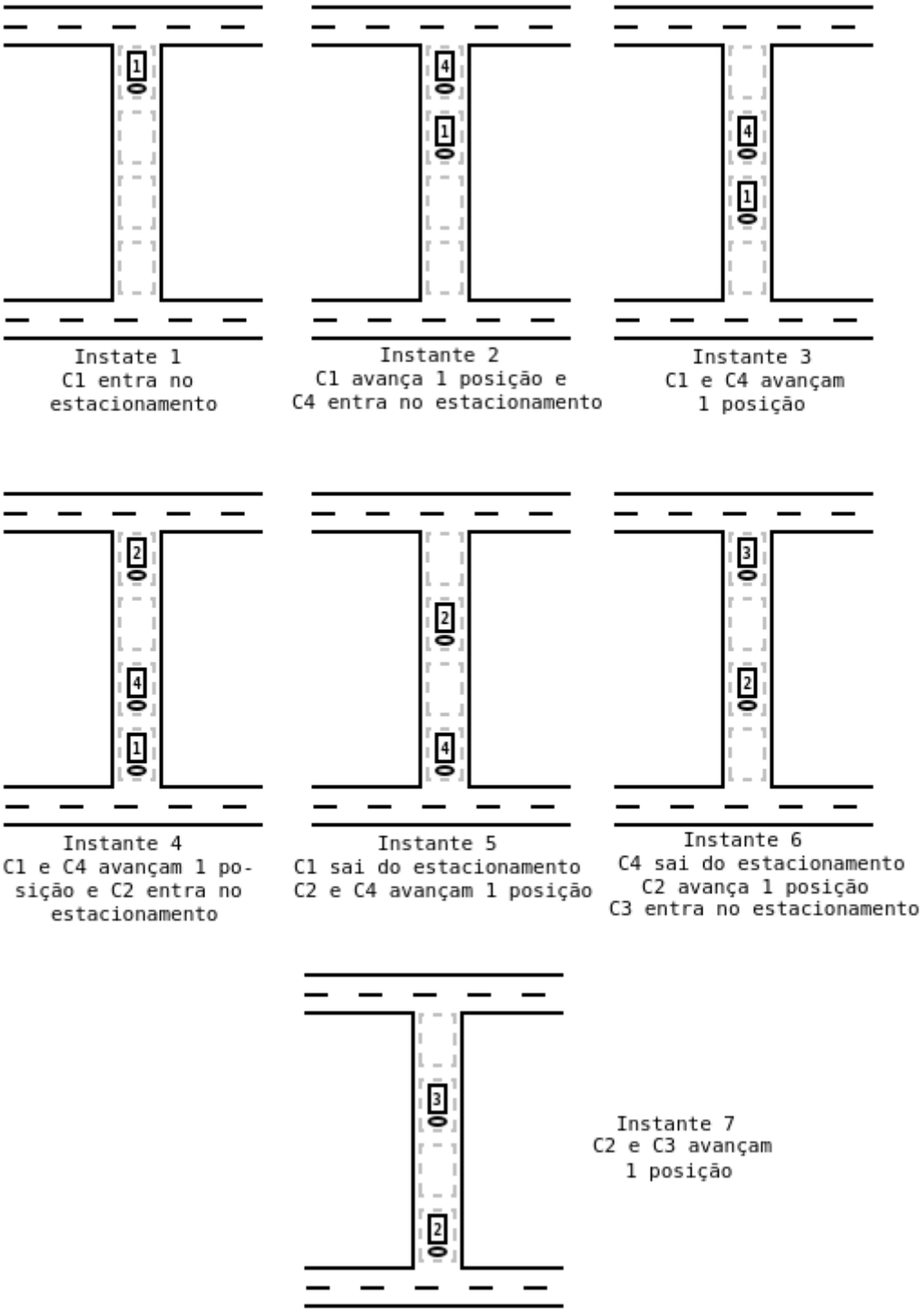
Exemplos

Exemplo 1

Entrada do exemplo 1:

```
estado_atual(4,[1,4,6,2],7)
```

No exemplo acima perceba que K = 4 (tamanho do estacionamento). Além disso, sabemos que: o carro 1 entrou no instante 1, o carro 2 entrou no instante 4, o carro 3 entrou no instante 6 e o carro 4 entrou no instante 2. Por fim, o instante em que queremos saber como está o estacionamento é o 7.



Saída do exemplo 1:

```
[0, 3, 0, 2]
```

Exemplo 2

Entrada do exemplo 2:

```
estado_atual(5,[3,6,12,9,4,15,16],7)
```

Saída do exemplo 2:

```
[0, 2, 0, 5, 1]
```

Observações para o item B:

- O usuário pode considerar que dois carros nunca entrarão no mesmo instante no estacionamento
- Se o instante de interesse é X e um carro **entra** no estacionamento no mesmo instante X, considera-se que o carro **já está** no estacionamento. De maneira análoga, se um carro **sai** do estacionamento no mesmo instante X de interesse, considera-se que o carro **não está** no estacionamento.

Desafio. Amizades do Turing

Formato: .py

Atenção: esse case tem dois desafios: um de *análise de dados e IA* e outro de *lógica de programação e algoritmos*. Lembre-se de que:

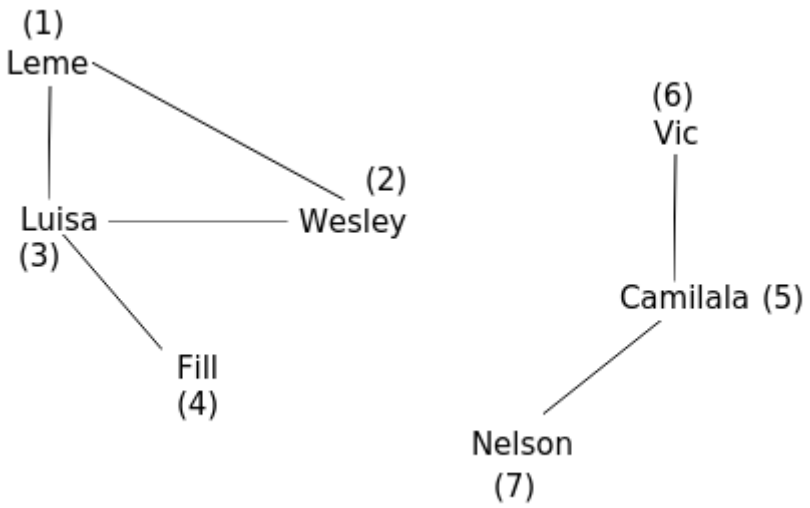
- eles valem uma **pontuação extra**, i.e., eles não podem elevar a sua nota no case técnico acima dos 100 pontos; e
- Se optar por fazê-los, você deve escolher **no máximo um deles**. Em outras palavras, **não entregue ambos desafios**.

Item Único (10 pontos)

Submissão: 1_logica/desafio.py

O Turing é formado por membros de vários institutos da USP, como IME, FFLCH, FEA e POLI. Os membros da área de RH perceberam que micro grupos estão se formando dentro do Turing devido a proximidade de membros do mesmo instituto, o que atrapalha a integração geral do grupo. Você, a fim de ajudar a área de RH do grupo a entender tal fenômeno, se comprometeu a criar uma função ***micro_grupos(membros, relacionamentos)*** que dada a relação de amizade entre os participantes do Turing, devolve quantos micro grupos existem atualmente no grupo. Por simplicidade, os relacionamentos podem ser representados como um **grafo**, como o abaixo.

Obs: Caso um determinado subgrafo tenha apenas 1 membro (pessoa isolada), isto é, um subgrafo com apenas 1 nó, ele **deve** ser considerado como um micro grupo.



No exemplo acima, cada nó é um membro, e cada aresta (linha) representa um relacionamentos, demonstrando que as duas pessoas envolvidas são amigas. Analisando o grafo, percebemos que o número de micro grupos é **2**.

Referências gerais para o desafio:

- Grafos
- Introdução à representação de grafos em Python

Formato de Entrada e Saída

Entrada

Sua entrada é composta de um inteiro *membros* , que representa o número de membros do Turing e de um vetor *relacionamentos*, composto por outros vetores de 2 elementos A e B, correspondendo a um relacionamento entre os membros A e B. .

Saída

Seu programa deve imprimir o número de micro grupos existentes no Turing.

Exemplos

Exemplo 1

Nesse exemplo, que corresponde ao grafo mostrado no enunciado, o primeiro valor da entrada é o número de membros (7) e o segundo é o vetor de relacionamentos. O grafo correspondente é o mesmo que foi mostrado no início dessa questão.

Entrada do exemplo 1:

```
micro_grupos(7, [[1,2],[2,3],[3,1],[3,4],[6,5],[5,7]])
```

Saída do exemplo 1:

```
2
```

Parte 2

Análise de Dados e Inteligência Artificial

40 pontos | Nível I

Caso tenha alguma dúvida nessa parte do case (seja de interpretação do enunciado ou alguma dúvida técnica), é só perguntar para a gente no nosso Discord (no canal **#duvidas-analise1**), no nosso e-mail turing.usp@gmail.com, na nossa página facebook.com/turing.usp ou mandar uma mensagem para algum dos membros da equipe abaixo:

Rian Pereira	+55 (12) 98827-1787
Letícia Falconer	+55 (11) 99507-6247
Mariana Dutra	+55 (21) 99159-9955

Nessa etapa, você deverá realizar uma análise de dados básica. Para isso, você pode consultar, além de outros recursos disponíveis na internet:

- O exemplo analise_wine.ipynb, que mostra uma análise básica de um dataset de vinhos tintos. O objetivo é identificar como as características do vinho afetam a sua qualidade.
- Os nossos posts sobre visualização de dados no medium: https://medium.com/turing-talks/turing-talks-9-visualiza%C3%A7%C3%A3o-de-dados-93df670d479

  e

  https://medium.com/turing-talks/como-visualizar-e-analisar-dados-com-python-f209bfbae68e
- É recomendado que você utilize algumas bibliotecas de visualização de dados para fazer as análises solicitadas, dentre elas recomendamos fortemente:
  - Pandas - Python: https://pandas.pydata.org/docs/
  - Matplotlib - Python: https://matplotlib.org/stable/contents.html
  - Seaborn - Python: https://seaborn.pydata.org/

- Plotly - Python: https://plotly.com/python/

Lembramos que uma análise de dados de qualidade não é feita apenas com gráficos, **é necessário interpretar as informações que o gráfico apresenta**, tirando conclusões e *insights*.

Google Colab

Essa parte do case deve ser feita **utilizando jupyter notebook.** Se você não possui o jupyter instalado no seu computador, se preferir, pode usar o **Google Colab.** O Google Colab é uma ferramenta do Google que permite rodar jupyter notebooks na nuvem.

Para criar um notebook no Colab, acesse o link https://colab.research.google.com/notebook#create=true.

Caso vá fazer o desafio, recomendamos que utilize um *runtime* com GPU. Para isso clique em *Ambiente de Execução > Alterar tipo de ambiente de execução > Acelerador de hardware > GPU.*

O Dataset

A distribuição de renda é um tema em voga atualmente na sociedade brasileira, bem como ao redor de todo o mundo. A renda de uma pessoa depende de diversos fatores, como nível escolar, idade, gênero, ocupação, entre outros. Tendo isso em vista, neste exercício, vamos analisar uma base de dados que apresenta um censo realizado nos EUA que, após verificar diversas características, informa se uma pessoa ganha mais ou menos de 50 mil dólares por ano. O objetivo desta análise é verificar as **relações entre as características de uma pessoa e sua renda final**.

Seu objetivo, como cientista de dados, é fazer uma análise das colunas e apresentar insights que você encontrar nessa jornada. Se baseie em nosso post do Medium e no notebook de exemplo, mas você pode utilizar outras funções e métodos para extrair mais informações úteis.

OBS: Não se preocupe em tirar insights "milagrosos", busque utilizar e procurar métodos interessantes e **comunique** suas descobertas em forma de comentários. Estamos testando mais o seu esforço do que seus resultados aparentes.

Link do dataset:
https://drive.google.com/drive/folders/1YmVrzrhZI-vwBuRTU-75cJNeTxuTGt_9?usp=sharing

Para orientar sua análise, segue uma breve descrição de cada uma das colunas do dataset:

1. **age** - Idade;
2. **workclass** - Classe de trabalho, ou seja, o tipo de vínculo empregatício (ex. Private (privado), State-gov (governo estadual));
3. **fnlwgt** - "final weight", representa o número de unidades na população alvo que correspondem àquela unidade representativa (em outras palavras, é um peso atribuído para cada linha, cujo significado literal é que, para obter os dados completos, deve-se replicar cada linha pelo número dessa feature. E, sim, seriam bilhões de linhas, mas não se assuste com o tamanho, pois são dados acumulados ao longo de décadas.);
4. **education** - Nível de escolaridade representado por categorias;
5. **educational-num** - Nível de escolaridade representado numericamente em ordem crescente de anos dedicados à educação (números inteiros de 1 a 16);
6. **marital-status** - Estado civil;
7. **occupation** - Ocupação, ou seja, o tipo de trabalho exercido por aquela pessoa (ex. Sales (vendas), Farming-fishing (fazenda e/ou pesca));
8. **relationship** - Papel que esse indivíduo ocupa dentro do seu ciclo familiar (ex. Husband (marido), Not-in-family (não está em uma família), Wife (esposa));
9. **race** - Etnia;
10. **gender** - Gênero;
11. **capital-gain** - Ganhos por fontes de investimento externas ao salário;
12. **capital-loss** - Perdas por fontes investimento externas ao salário;
13. **hours-per-week** - Horas de trabalho semanais;
14. **native-country** - País de origem;
15. **income** - Renda anual, pode ser <=50K (menor ou igual a 50 mil dólares ao ano) ou >50K (maior que 50 mil dólares ao ano).

Execução

Submissão: 2_analise-e-ia/analise_e_modelos.ipynb

Para realizar esta parte do case, vamos ajudá-los dando alguma direção geral para a análise, mas lembre-se que você deve, e encorajamos muito, buscar outras análises que possa considerar relevantes para justificar sua conclusão sobre o tema proposto. Orientamos também que divida seu notebook em duas partes: **dados contínuos e dados categóricos.**

Reforçamos que a forma como os gráficos e as análises são expostos influencia a avaliação. Portanto, recomendamos fortemente que tudo seja explicado de forma clara e justificada, além de os gráficos serem bem construídos.

Para a construção de gráficos melhores, incentivamos que vocês, além de lerem os Turing Talks, vejam as documentações da bibliotecas utilizadas (os links estão acima), para que entendam alguns parâmetros que podem ser alterados, como cores, tamanho do gráfico, entre outros.

Análise de Dados Contínuos (15 pontos)

Nessa primeira parte da análise de dados, você deve analisar somente as **features numéricas** (principalmente *age, educational-num* e *hours-per-week*). O seu objetivo é, de forma similar ao que foi feito no exemplo analise_wine.ipynb, analisar cada feature individualmente, assim como as relações entre as features. No entanto, a sua análise deve ir um pouco além do que foi mostrado no exemplo (pense, por exemplo, em utilizar outros tipos de visualizações e identificar outliers). Para isso, recomendamos que você veja o nosso post de visualização de dados (acima).

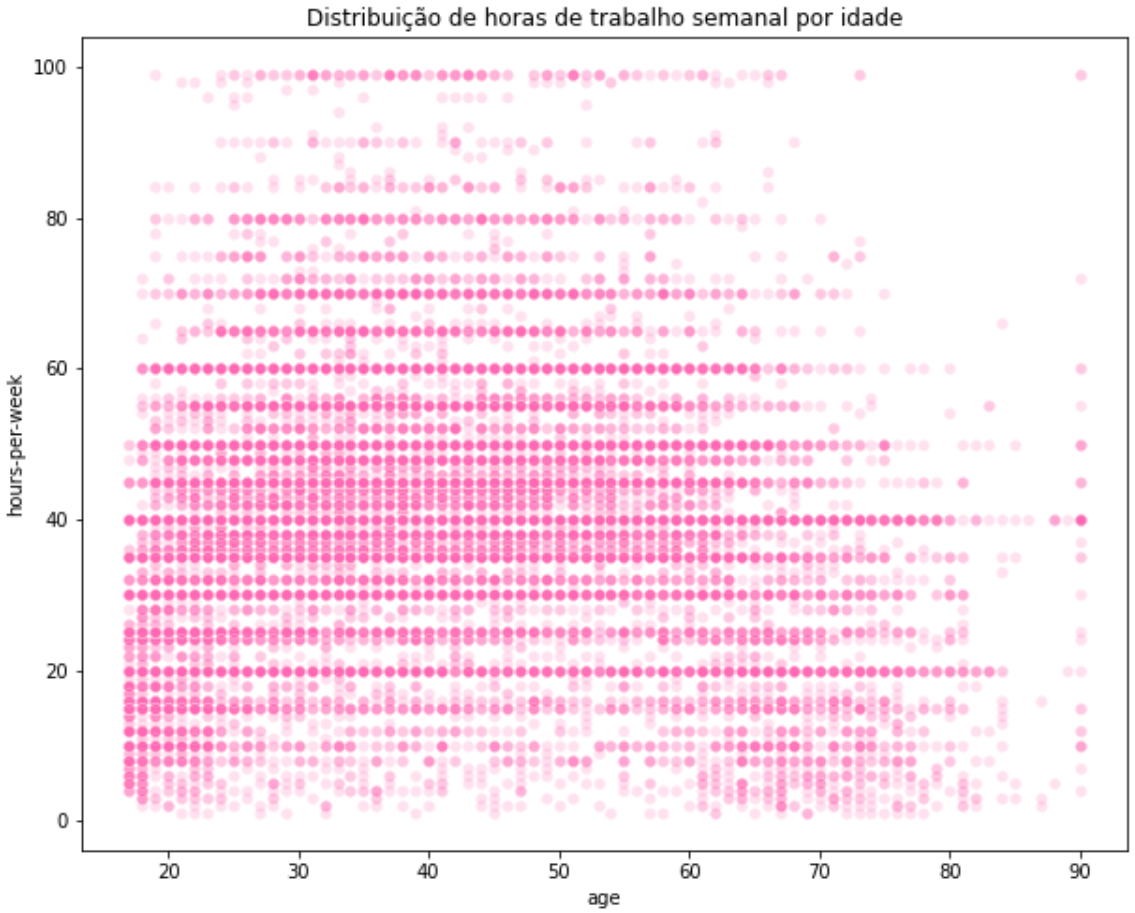
Você pode seguir esta estrutura, mas lembre-se de dar personalidade à sua análise indo além dessas propostas.

- Analisar a distribuição de idades (gráfico de idade vs quantidade em cada categoria);
- Analisar a distribuição da quantidade de anos destinados à educação (gráfico de número de anos despendidos para educação vs quantidade em cada categoria);

- Analisar a distribuição de jornadas de trabalho semanais (quantas horas por semana são destinadas ao trabalho);
- Existe alguma relação clara entre a renda e a idade das pessoas? Quais as idades predominantes? Pela sua análise, é possível verificar alguma evidência de diferença de salários entre quais grupos de pessoas? Justifique.

Obs: Os itens acima servem para guiar sua análise, mas ela não deve ser apenas uma série de respostas a estas perguntas, tente seguir a estrutura do analise_wine.ipynb.

Para fazer suas análises, é importante tentar fazer gráficos que proporcionem informações mais relevantes sobre o objetivo de análise, neste sentido pode ser interessante tentar fazer análises de 2 ou até três características (e, para trabalhar com dados numéricos, talvez seja interessante trabalhar com diferentes tipos de gráficos), por exemplo:



Este gráfico, ao invés de apenas retratar a distribuição de idade ou de horas semanais trabalhadas, expõe a relação da distribuição entre essas features, o que possibilita mais informações.

Por isso tente sempre **enriquecer** sua análise.

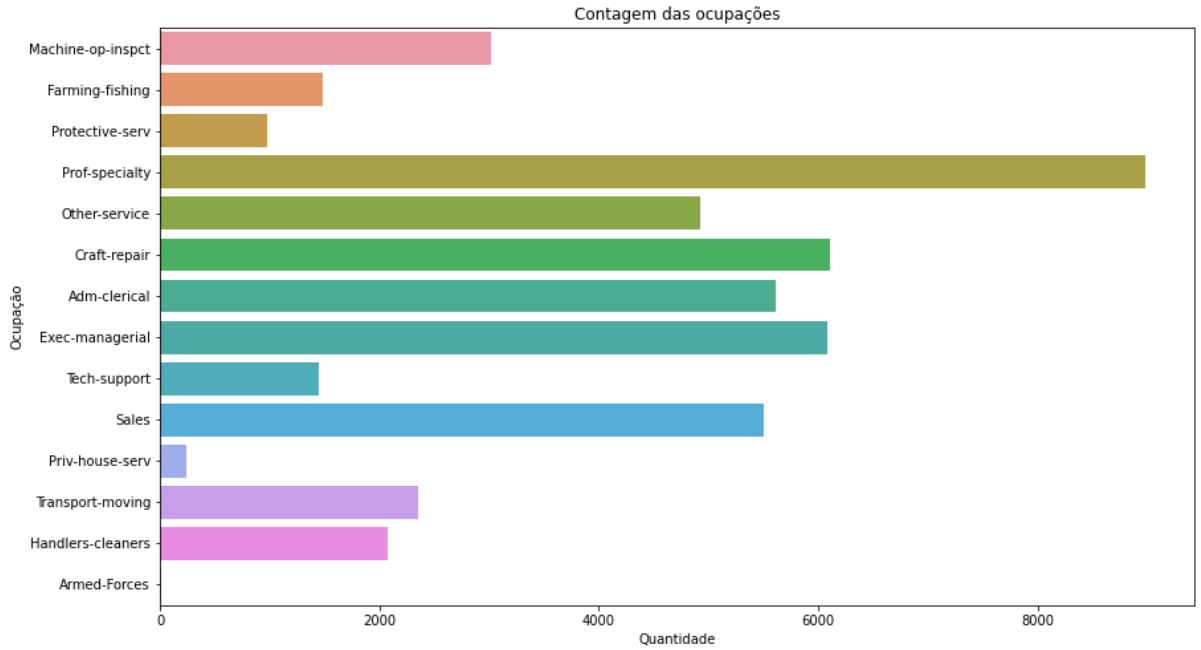
Análise de Dados Categóricos (15 pontos)

Agora que você já analisou as features numéricas, é hora de analisar as **features categóricas**, como *marital-status, education, workclass* e *race*. Diferentemente das features numéricas, os valores dessas features são restritos a um conjunto de categorias. Você novamente deve analisar as features categóricas individualmente, assim como as relações entre elas. Além disso, nessa parte você também deve analisar as relações de features numéricas com categóricas.

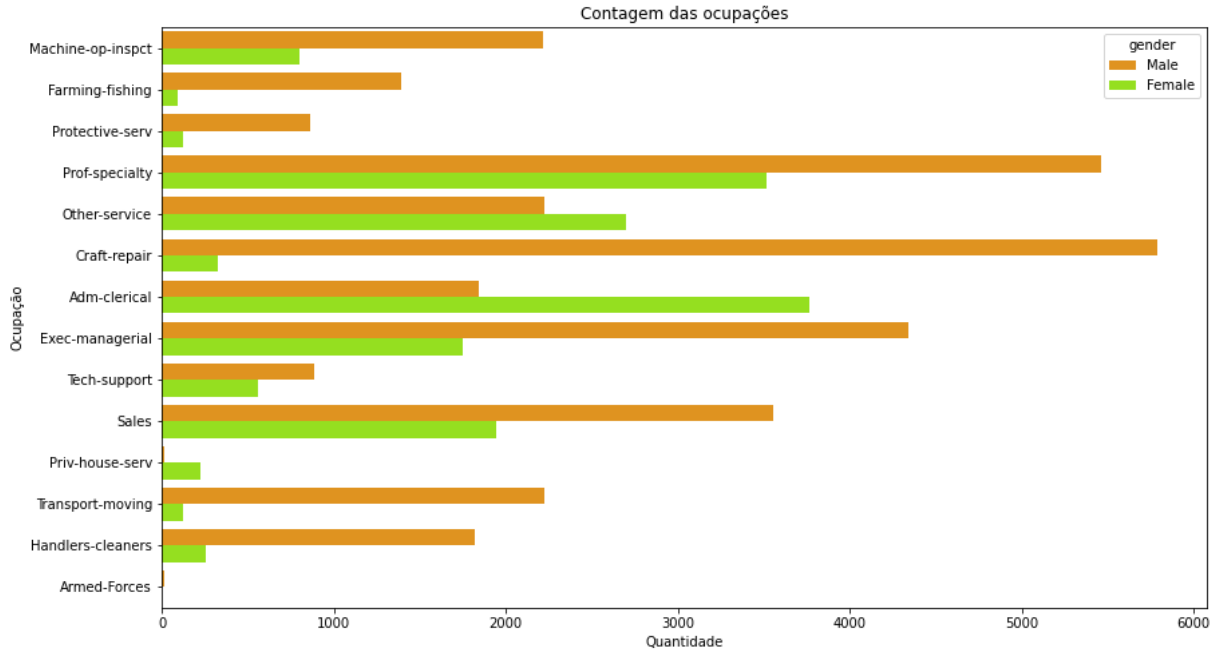
Novamente vamos passar algumas dicas, mas lembre-se de dar personalidade à sua análise indo além dessas propostas.

- Analisar a distribuição do estado civil.
- Analisar a distribuição do nível educacional (com a coluna “education”, que possui uma representação discreta/categórica da educação). Tente fazer essa análise conjuntamente com a renda.
- Analisar a distribuição das classes de trabalho.
- Existe alguma relação clara entre a renda e o estado civil das pessoas? A renda também está relacionada à etnia? Pela sua análise, é possível verificar alguma evidência de diferença de renda entre quais grupos de pessoas? Justifique

Tanto para análises numéricas quanto categóricas é importante tentar fazer gráficos que proporcionem informações mais relevantes sobre o objetivo da análise, neste sentido também pode ser interessante tentar fazer análises de 2 ou até três características, por exemplo:



O gráfico acima tem apenas o tipo de ocupação no eixo y, com a contagem de pessoas naquela categoria em x. Este gráfico traz informações apenas sobre uma característica: tipo de ocupação. Mas podemos correlacionar o gênero nesta análise e temos:



Por isso tente sempre **enriquecer** sua análise.

Conclusões (10 pontos)

Um cientista de dados, depois de promover as suas análises, deve chegar a conclusões acerca do assunto que tanto buscou entender. Você, portanto, deverá, da forma como desejar, apresentar, em uma conclusão, **os principais pontos a que conseguiu chegar**. Para isso, você pode resumir as análises que obteve e tentar relacioná-las.

Por que será que os resultados obtidos foram esses? Tente chegar a uma justificativa, seja por meio de suas análises, sugestões ou pesquisas. Sinta-se à vontade para concluir como achar melhor.

Desafio: Regressão Logística (10 pontos)

Atenção: esse case tem dois desafios: um de *análise de dados e IA* e outro de *lógica de programação e algoritmos*. Lembre-se de que:

- Eles valem uma **pontuação extra**, i.e., eles não podem elevar a sua nota no case técnico acima dos 100 pontos; e
- Se optar por fazê-los, você deve escolher **no máximo um deles**. Em outras palavras, **não entregue ambos os desafios**.

Após um cientista de dados ter uma noção consistente de seus dados, ele pode buscar aplicar um modelo de inteligência artificial sobre a base de dados para buscar predições. O modelo que vamos utilizar neste problema é a **regressão logística**, que é considerado um modelo simples de classificação. Nesta última etapa da análise, você deve aplicar este algoritmo nos dados anteriores, de forma a tentar prever se a renda de uma pessoa é maior ou menor que 50 mil doláres (coluna *income*), dada suas características. Sinta-se livre para consultar os diversos recursos disponíveis na internet sobre o assunto. Para isso, recomendamos as seguintes informações:

- O nosso post sobre regressão logística no medium: https://medium.com/turing-talks/turing-talks-14-modelo-de-predi%C3%A7%C3%A3o-regress%C3%A3o-log%C3%ADstica-7b70a9098e43

- Não é necessário implementar a regressão logística do zero, utilize uma biblioteca como scikit-learn no python
- Utilize métricas de avaliação para saber se seu modelo de regressão logística está razoável ou não.
- Sinta-se livre para tentar diferentes abordagens, fica a seu critério, por exemplo usar uma única *feature*, várias ou todas.
- Uma parte importante será o tratamento dos dados categóricos, que não podem ser passados para um modelo como categorias. Para isso dê uma procurada em one-hot encoding e get dummies.

O guia aqui é alcançar uma regressão logística com uma acurácia maior que 0,7. Para saber mais sobre a acurácia e métricas de avaliação leia este texto: Como Avaliar Seu Modelo de Classificação | by Gustavo Korzune Gurgel | Turing Talks | Medium.

Parte 3: Turing Talks

20 pontos | Nível Intermediário

Questão Única

Submissão: 3_turing-talks/turing-talks.pdf

O Turing USP possui três pilares principais que promovem a sustentação de todas as atividades desenvolvidas interna e externamente pelo grupo, são eles: Estudar, Aplicar e Disseminar conhecimentos de inteligência artificial e ciência de dados.

O pilar de disseminação de conhecimento é de extrema importância para o objetivo do grupo de se tornar uma referência no tema ao redor de São Paulo e do Brasil. Para tal, temos diversos meios que são importantes: fazemos workshops, rodas de conversas e outros eventos, buscamos a publicação de projetos desenvolvidos internamente, como, por exemplo, a publicação de um artigo sobre o projeto do Fernando Pessoa na revista Superinteressante.

Além disso, gravamos aulas e workshops para serem disponibilizados no canal do youtube do grupo. Em especial, uma das atividades mais importantes para o pilar de disseminação é a publicação semanal dos Turing Talks, na plataforma Medium. Estes textos possuem um alcance muito grande por conta do caráter específico de divulgação científica da plataforma. Devido ao seu fácil acesso, linguagem descontraída, abordagem mais direta, periodicidade e por muitas vezes apresentar de forma consistente um tema dificilmente encontrado com a mesma simplicidade na língua portuguesa, o Turing Talks é um dos projetos recorrentes mais importantes do grupo.

Neste contexto da importância, você deve redigir um **Turing Talks** sobre **um dos temas listados abaixo** que aborda conceitos de inteligência artificial. Como referência para esse trabalho você tem a sua disposição um grande acervo em nossa página no Medium: Turing Talks, bem como qualquer outra fonte na internet. O objetivo é escrever um texto que seja informativo e conciso, não queremos textos puramente expositivos como uma página do wikipédia.

O texto não precisa seguir a norma culta, mas imagine que será um texto publicado com seu nome nele, para milhares de pessoas que buscam aquele conhecimento vão ler

e julgar o conteúdo, por isso é esperado que não existam erros ortográficos e gramaticais graves.

Temas:

- Regressão Linear
- Métricas para avaliação de modelos de machine learning
- Feature engineering (one-hot encoding, normalização de dados, etc)

IMPORTANTE: Caso tenha prestado o nosso processo seletivo em 2021, você deve escolher um tema diferente do que escolheu no ano passado.

Limite de tamanho: 4000 caracteres, sem contar espaços, referências e títulos. Recomendamos a fonte Arial ou Times New Roman 12.

Formato de entrega: PDF com o nome especificado na seção *Formato de Submissão*.

Critérios de avaliação:

- Respeitar o limite máximo de caracteres.
- Texto coeso, com começo, meio e fim.
- Referências de pesquisa e construção do texto são **obrigatórias** e podem seguir o formato de sua preferência (uma listagem final do texto, acompanhando-o conforme as referências aparecem...)
- O título e a estrutura estética em um Turing Talks são importantes, atente-se às imagens que você usa para construir o seu texto.
- Originalidade e criatividade do texto.