Processo Seletivo Turing USP - 2021

Case Técnico

Pontuação Total: 100 pontos

Olá, candidate! Seja muito bem-vinde ao processo seletivo Turing USP 2021! 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 e vídeos 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 nos dias 28/05 (Sexta-feira) das 17h às 19h e 02/06 (Quarta-feira) das 19h às 21h**, que vão ocorrer **online**, no nosso **Discord**.

O case é dividido em três partes, uma de lógica de programação e algoritmos, outra parte de análise de dados e inteligência artificial e uma terceira composta pelo texto no formato de um Turing Talks. Na primeira parte, **lógica de programação e algoritmos**, serão aceitas resoluções em **Python, C, C++ e Java**. Na segunda parte, **análise de dados e inteligência artificial**, serão aceitos os **formatos jupyter notebook (em Python e R).** A última parte, **redação do Turing Talks**, deve ser

enviada em formato **PDF**. Para a parte de lógica e algoritmos, utilizaremos **correção automática** (ou seja, uma primeira correção — antes da correção humana — será feita por um programa de computador). Portanto, é **proibida a utilização de acentos nas palavras na hora de escrever no seu código**, isso porque pode gerar problemas durante a correção automática.

Também é importante lembrar que quanto mais bem comentado o código melhor, pois facilitará o entendimento e a lógica por trás do mesmo. Neste sentido, códigos mal documentados podem resultar em perda de pontuação.

Formato de Submissão

Quando terminar seu case, envie-o nesse forms. Não aceitaremos múltiplas submissões.

Você deve seguir o seguinte formato:

- As respostas dos exercícios de lógica de programação e algoritmos podem ser entregues em Python (.py), C (.c), C++ (.cpp) e Java (.java). 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).
- 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 (ou demais formas)
| ├ q1b.py (ou demais formas)
| ├ q2a.py (ou demais formas)
| ├ q2b.py (ou demais formas)
| └ desafio.py (ou demais formas)
├ 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, 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:

Wesley Almeida	(python, c, c++, java)	+55 (11) 97027-6690
Fernando Matsumoto	(python, c, c++, java)	+55 (19) 99850-2800
Guilherme Salustiano	(python, c, c++, java)	+55 (41) 99928-7997
Rodrigo Fill	(python, c, c++)	+55 (11) 97277-0014
Noel Eliezer	(python)	+55 (16) 98126-8981
Ariel Guerreiro	(python)	+55 (11) 94331-1181

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:

- Os exemplos 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.

Observações técnicas:

- Nessa parte do case, estamos considerando as seguintes versões de cada linguagem:
  - **Python 3**: versão 3.8 ou anterior
  - **C/C++ 11** (códigos escritos em versões anteriores costumam funcionar na versão 11, mas é bom verificar)
  - **Java 15** ou anterior
- Em java, **não coloque o seu código dentro de nenhum pacote** (não coloque `package xxx;` no início do arquivo).

Q1. Vacas magras no Turing

Formato: .py, .c, .cpp, .java

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` e `azank.pistolado@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
a z a n k . p i s t o l a d o @ u s p . b r
```

Pelo exemplo acima é possível perceber que foram economizadas `6` letras. Os membros cogitaram remover também os caracteres finais que se repetissem, mas entenderam que isso poderia tornar muito difícil a leitura para o professor.

Item A (10 pontos)

Submissão: 1_logica/q1a.py (.c, .cpp, .java)

Você, entusiasmado por ter entrado no grupo, se propôs a fazer um programa que recebe a lista de emails e indica quantas letras foram economizadas. Neste item, você deve determinar quantas letras podem ser economizadas **mantendo a ordem** dos emails.

Obs 1: Considere que todos os emails passados tem 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.

Referências do Item A:

- Vetores - IME-USP
- Ordenação de vetores

Formato de Entrada e Saída

Entrada

Sua entrada é composta de N, seguido de N emails (um por linha).

Limites:

- 1 < N < 100

Saída

Seu programa deve *printar* um número indicando quantas letras podem ser economizadas para o conjunto N de e-mails.

Exemplos

Exemplo 1

Entrada do exemplo 1:

```
3
camila.lobianco@usp.br
camilalala.topp@usp.br
azank.pistolado@usp.br
```

Saída do exemplo 1:

```
6
```

Exemplo 2

Entrada do exemplo 2:

```
5
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 (.c, .cpp, .java)

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 printar os e-mails corretamente formatados a partir da lista impressa ou `ERRO` caso o domínio esteja errado.

Formato de Entrada e Saída

Entrada

Sua entrada é composta de um inteiro N que representa a quantidade de e-mails impressos. Em seguida são fornecidos N e-mails, um em cada linha.

Limites:

- 1 < N < 10000

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:

```
3
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
```

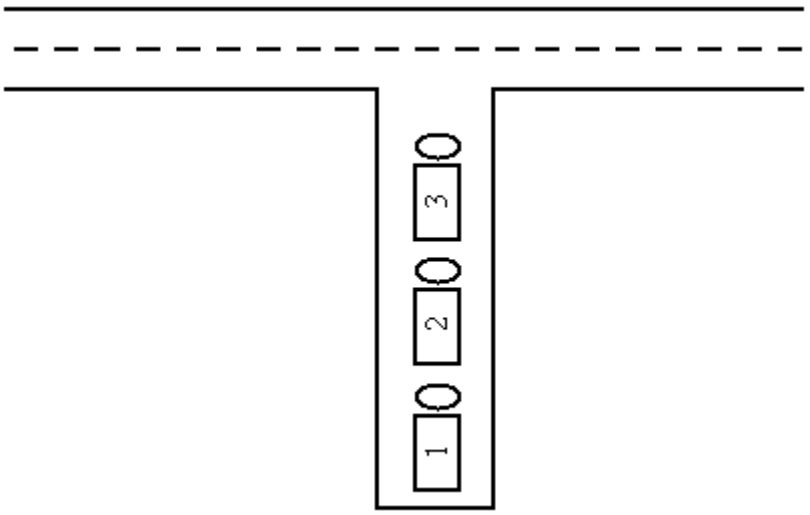
Q2. Estacionamento do Nelnelson

Formato: .py, .c, .cpp, .java

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 um programa 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 N (tamanho do estacionamento e número de instantes que serão monitorados). Em seguida, cada linha representa um instante, ou seja, a primeira linha é o primeiro instante, a segunda, o segundo instante e assim por diante. Para cada linha será dado 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. **Veja o exemplo 1 para maiores explicações**.

Limites:

- $1 \leq K < 1000$
- $1 \leq N < 100000$
- $1 \leq |C| < 1000$

Obs: $|x|$ representa o módulo ou valor absoluto de x, isto é, se $x \geq 0$, então $|x| = x$. Se $x < 0$, então $|x| = -x$. Por exemplo: $|-3| = 3$ e $|3| = 3$. Acima estamos dizendo que o valor absoluto da variável C estará sempre entre os limites especificados.

Saída

Você deve *printar* `sim`, caso seja possível que todos os carros estacionem e saiam sem problemas e `nao`, caso contrário. Caso um carro não consiga entrar no estacionamento por ele já estar lotado, seu programa deve printar `nao`.

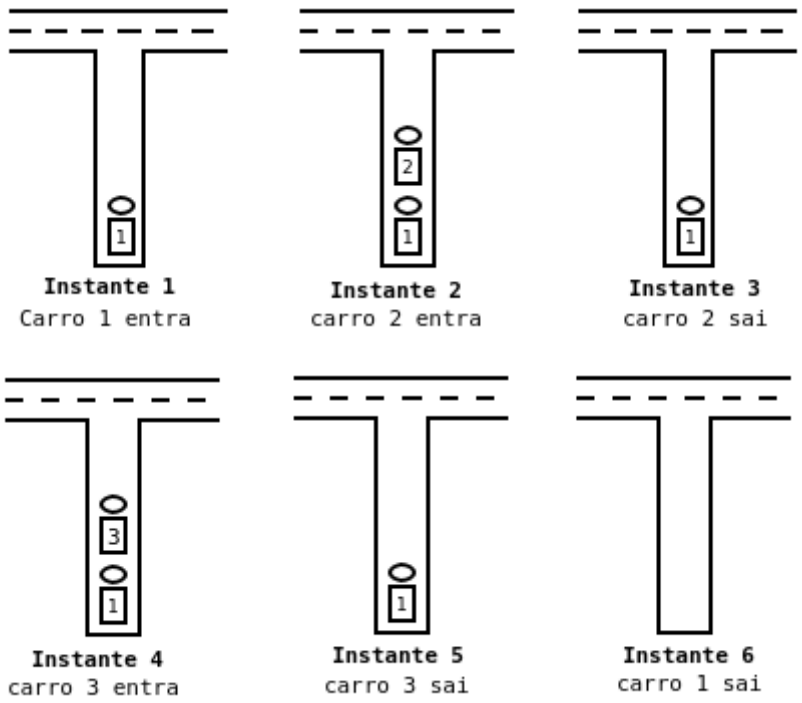
Exemplos

Exemplo 1

Entrada do exemplo 1:

```
3 6
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:

```
3 8
1
2
-2
3
5
-3
-1
-5
```

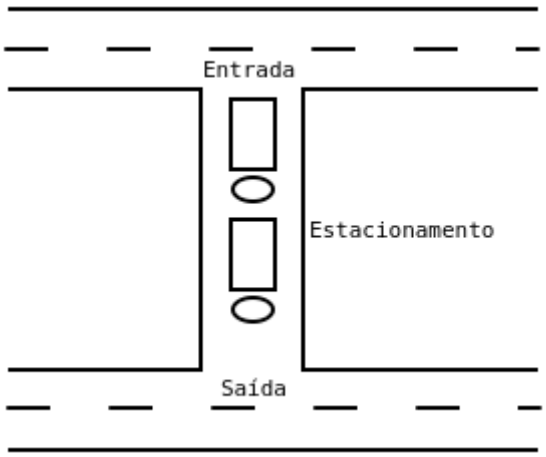
Saída do exemplo 2:

```
nao
```

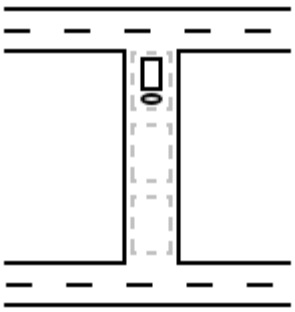
Item B (10 pontos)

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:

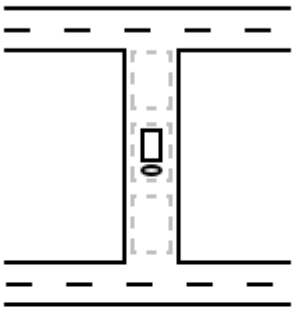


Este estacionamento funciona de uma maneira muito peculiar, pois a cada novo instante o carro avança uma posição dentro dele. Veja o exemplo para um estacionamento de tamanho K = 3 e um determinado carro:



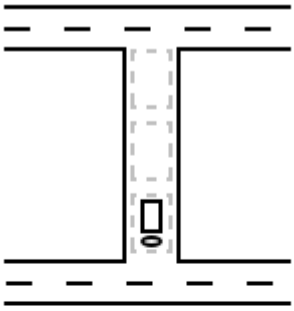
Instante 1:

O carro está no estacionamento



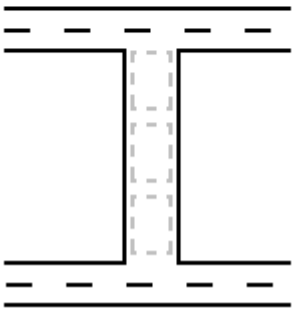
Instante 2:

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?

Formato de Entrada e Saída

Entrada

Sua entrada é composta por dois números K e N (tamanho do estacionamento e números de carros que entram no estacionamento), em seguida, nas próximas N linhas é dado o horário de entrada (E_n) de cada carro (a primeira linha representa o carro 1, a segunda linha o carro 2 e assim por diante). Por fim é dado um instante T, no qual queremos saber como está o estacionamento.

Limites:

- $1 < N, K < 10000$
- $1 \leq E_n < 1000$
- $1 \leq T < 10000$

Saída

Sua saída deve ser composta de K números (tamanho do estacionamento) **separados por 1 espaço entre cada número e em uma única linha**, que indicam como está o estacionamento no instante T. Caso uma posição do estacionamento não possua nenhum carro, você deverá *printar* `0`. Além disso, o primeiro número impresso representa o carro na primeira posição do estacionamento, isto é, logo depois da entrada. Veja o exemplo a seguir para maiores explicações.

Exemplos

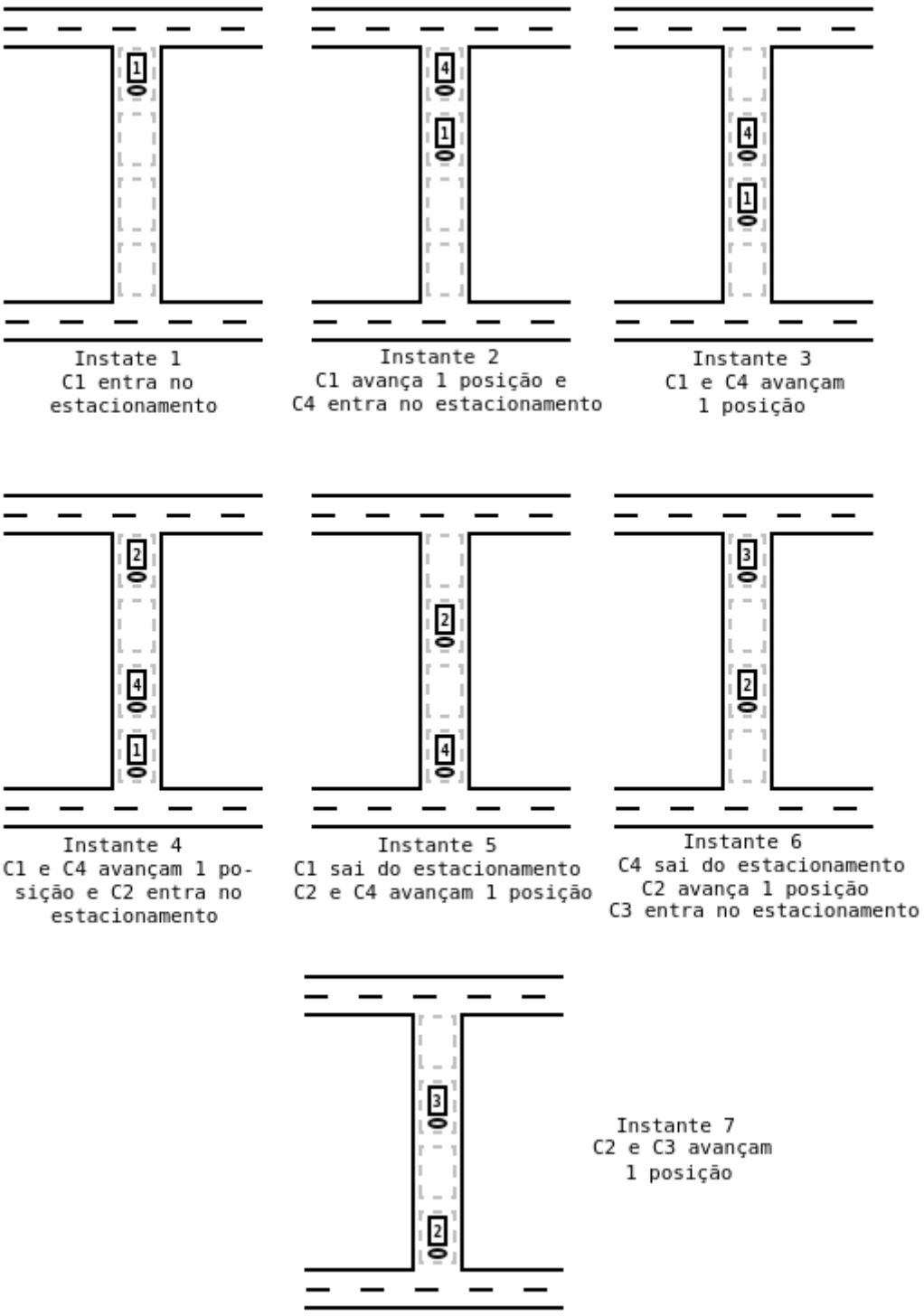
Exemplo 1

Entrada do exemplo 1:

```
4 4
1
4
6
2
7
```

No exemplo acima perceba que K = 4 (tamanho do estacionamento) e N = 4 (número de carros que entram no 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:

```
5 7
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, .c, .cpp, .java

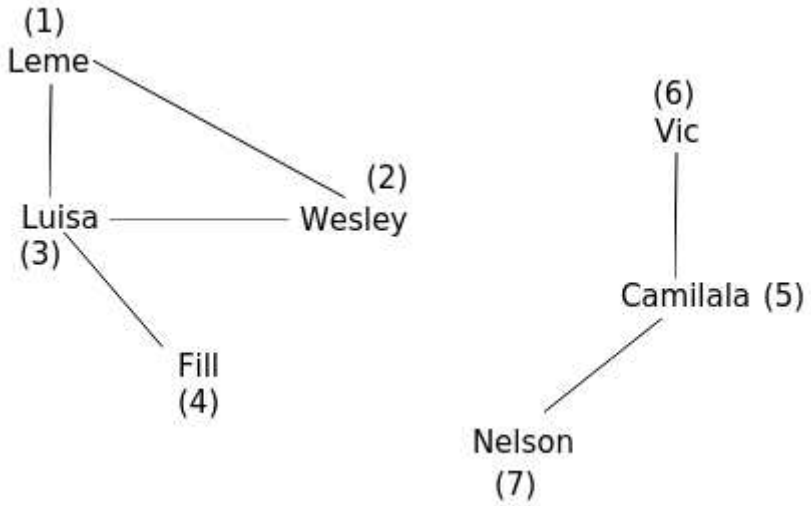
Atenção: esse case tem dois desafios: um de *análise de dados e IA* e outro de *lógica de programação e algoritmos*. Lembre-se de que:

- 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 (.c, .cpp, .java)

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 um programa 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.



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 dois inteiros N e K, que representam o número de membros e de relacionamentos existentes entre os membros do Turing. Cada uma das próximas N linhas contém dois inteiros A e B, correspondendo a um relacionamento entre os membros A e B.

Limites:

- $1 < N, K < 50$
- $1 < A, B < 100$

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 a quantidade de relacionamentos (6) no grupo. Em seguida, cada linha representa um relacionamento. O grafo correspondente é o mesmo que foi mostrado no início dessa questão.

Entrada do exemplo 1:

```
7 6
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 Intermediário

Caso tenha alguma dúvida nessa parte do case, é só perguntar para a gente no nosso Discord (no canal **#duvidas-intermediario-analise**), 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:

Rodrigo Fill	(python)	+55 (11) 97277-0014
Camilla Fonseca	(python, R)	+55 (11) 96163-7978
Luísa Mendes Heise	(python, R)	+55 (11) 99218-7534
Noel Eliezer	(python)	+55 (16) 98126-8981
Vitoria Rodrigues	(python)	+55 (11) 98748-8154

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. Esse exemplo mostra uma análise básica de um dataset de vinhos tintos. O objetivo é identificar como as características do vinho afetam a sua qualidade.
- O 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/R : https://plotly.com/python/
- Tidyverse - R: https://www.tidyverse.org/packages/
- GGplot - R: https://ggplot2.tidyverse.org/index.html

- Lembrando 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.

Para criar o notebook em R, utilize o link: https://colab.research.google.com/notebook#create=true&language=r.

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 desigualdade de gênero é um problema em voga atualmente na sociedade brasileira, bem como ao redor de todo o mundo. Gênero é um fator que influencia o salário que uma pessoa recebe, bem como outros fatores como: nível escolar, idade, ocupação, entre outros. Tendo isso em vista, neste exercício, vamos analisar uma base de dados que juntou dados do site Glassdoor sobre renda, junto com diversas outras características relevantes para analisar o salário de uma pessoa. O objetivo desta

análise é verificar como a questão da desigualdade de gênero ainda influencia a diferença salarial entre homens e mulheres.

Seu objetivo, como cientista de dados, é fazer uma análise das colunas e informar 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://www.kaggle.com/nilimajauhari/glassdoor-analyze-gender-pay-gap

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 relevante para justificar sua conclusão sobre o tema proposto. Orientamos também que divida seu notebook em duas partes: dados contínuos e dados categóricos.

Análise de Dados Contínuos (20 pontos)

Nessa primeira parte da análise de dados, você deve analisar somente as features numéricas (principalmente age, BasePay e BonusPay). 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 de salários, tanto base como bônus.
- Analisar a distribuição de senioridade.
- Existe alguma relação clara entre os salários base e a idade das pessoas? Quais as idades predominantes? Pela sua análise é possível verificar alguma evidência de diferença de salários entre gêneros? 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.

Análise de Dados Categóricos (20 pontos)

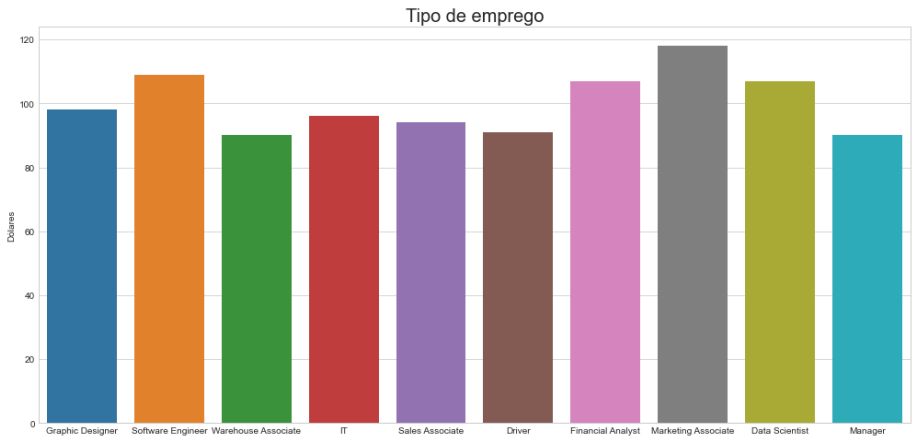
Agora que você já analisou as features numéricas, é hora de analisar as features categóricas, como JobTitle, Education, Dept e, claro, Gender. Diferentemente das features numéricas, os valores dessas features são restritos a um conjunto de categorias. Você novamente deve analisar as features categóricas 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.

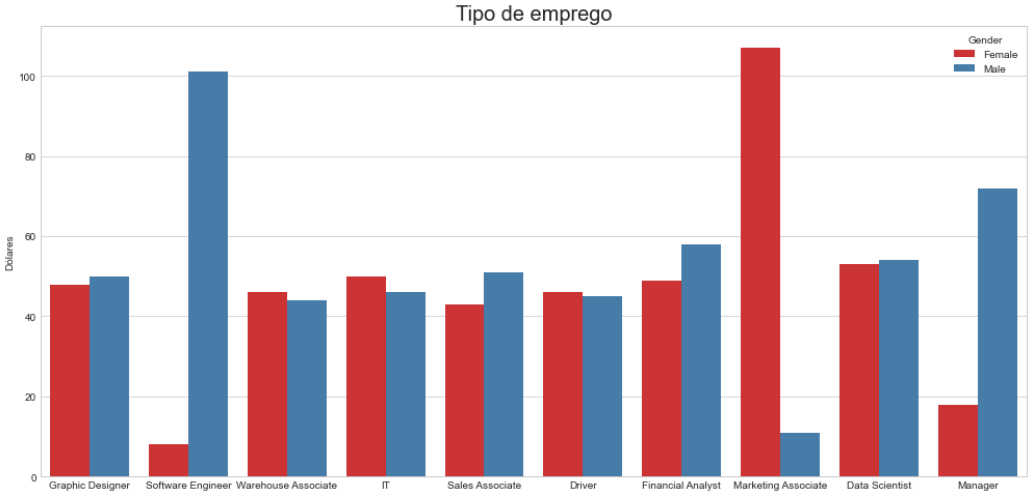
- Analisar a distribuição de gênero
- Analisar a distribuição do nível educacional. Tente fazer essa análise conjuntamente com o gênero.
- Analisar a distribuição de senioridade.
- Existe alguma relação clara entre os salários base e a idade das pessoas? Quais as idades predominantes? Pela sua análise é possível verificar alguma evidência de diferença de salários entre gêneros?

Tanto para análises numéricas quanto categóricas 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, por exemplo:



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



Por isso tente sempre enriquecer sua análise.

Desafio: Regressão Linear (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 linear, que é considerado um modelo simples de regressão, especialmente bom para dados com relações lineares. Nesta última etapa da análise você deve aplicar este algoritmo nos dados anteriores, de forma a tentar prever o salário de uma pessoa dada suas características. Existem diversos recursos disponíveis na internet para isso, recomendamos as seguintes informações:

- O nosso post sobre regressão linear no medium: https://medium.com/turing-talks/turing-talks-11-modelo-de-predi%C3%A7%C3%A3o-regress%C3%A3o-linear-7842709a593b
- Não é necessário implementar a regressão linear do zero, utilize uma biblioteca como scikit-learn no python e Caret em R
- Utilize métricas de avaliação para saber se seu modelo de regressão está razoável ou não.
- Pode 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 com um R-quadrado maior que 0,7. Para saber mais sobre o R-quadrado e métricas de avaliação leia este texto: Como avaliar seu modelo de regressão.

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)

Limite de tamanho: 4000 caracteres, sem contar espaços. 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**.
- 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.