**AED – Algoritmos e Estruturas de Dados – 2017/18**



**Trabalho em Grupo (2 a 4 alunos)**

**Fase I e Fase II**

**“SIMPAR – Simulação de Passageiros em Partida Aérea ”**

**Descrição**

Este Trabalho de Grupo, inserido na Avaliação Contínua de AED, tem como objectivo desenvolver um programa na Linguagem de Python e aplicar conceitos adquiridos em Algoritmos e Estruturas e Dados aproveitando as vantagens do paradigma Orientado por Objectos.

O domínio de aplicação é a Simulação de eventos com filas de espera, aplicado ao embarque de passageiros nos balcões de embarque com despacho de bagagem (adiante designado de balcão) para um determinado voo aéreo, cujo programa a elaborar se intitula: “SIMPAR – Simulação de Passageiros em Partida Aérea ”.

O objectivo do SIMPAR é o estudo dos tempos de espera dos passageiros nos balcões, no sentido de se avaliar se se poderá optimizar este serviço de atendimento, seja abrindo novos balcões ou fechando algum.

O SIMPAR utiliza o Tipo Abstracto de Dado – Queue – uma fila de espera associada a cada balcão, onde coloca os passageiros com bagagem (adiante designado de passageiro) a despachar que vão chegando para embarque no voo em causa.

**1.ª Fase**

***NOTA:*** *Utilize no programa SIMPAR os nomes de variáveis, funções, classes e métodos sugeridos neste enunciado.*

**1. As principais variáveis do SIMPAR** são as seguintes:

num\_pass – o número de passageiros com bagagem previsto para este voo

num\_bag – o número máximo de bagagens permitido por passageiro

num\_balcoes – o número de balcões abertos para atendimento e despacho de bagagem

ciclos – os ciclos de tempo em que a simulação decorre.

**2. Classes a considerar**

Considerar os seguintes componentes Classes e suas descrições:

- classe **Passageiro**

Cada objecto passageiro tem:

Os atributos:

**-** bag\_pass – número de bagagens do passageiro

- ciclo\_in – instante em que foi colocado na fila (número do ciclo da simulação)

Os métodos, para além do \_\_init\_\_(…):

- obtem\_bag\_pass – devolve o valor de bag\_pass

- obtem\_ ciclo\_in – devolve o valor de ciclo\_in

- \_\_str\_\_ - devolve uma string com formatação dos atributos, conforme o

exemplo de um passageiro com 4 bagagens no ciclo da simulação 2:

[b:4 t:2]

- classe **Balcao**

Cada objecto balcão tem:

Os atributos:

- n\_balcao – número do balcão

**-** fila – fila de passageiros deste balcão, inicializada vazia fila=Queue()

Considere fazer o import do módulo pythonds para o Anaconda de:

https://pypi.python.org/pypi/pythonds/1.2.1

e no código do SIMPAR, para utilizar a class Queue:

from pythonds.basic import Queue

- inic\_atend – instante em que o balcão está pronto para iniciar o atendimento do passageiro, iniciado com 0 (zero). Se houver passageiro a ser atendido, quando termina, passa a ser a ser o tempo de atendimento. Se estiver vazia a fila, passa ser o instante (ciclo) actual

- passt\_atend – número total de passageiros atendidos por este balcão iniciado com 0 e incrementado de 1 sempre que termina o atendimento de um passageiro

- numt\_bag – número total de bagagens despachadas por este balcão, actualizado sempre que termina o atendimento de mais um passageiro, com o bag\_pass deste

- tempt\_esp – tempo total de espera, em número de ciclos de simulação. É actualizado com o tempo de espera de um passageiro neste balção sempre que este termina de ser atendido

- bag\_utemp - número de bagagens despachadas pelo balcão por unidade de tempo (por ciclo), calculado aleatoriamente entre 1 e num\_bag sempre que o balcão é aberto.

Os métodos, para além do \_\_init\_\_(…):

- muda\_inic\_atend(*…*) – acumula em inic\_atend o “valor” do tempo de atendimento do passageiro

- incr\_passt\_atend(…) – incrementa em 1 o passt\_atend - total de passageiros atendidos por este balcão

- muda\_numt\_bag(…) – acumula em numt\_bag do balcão, o bag\_pass do passageiro quando este termina de ser atendido

- muda\_tempt\_esp(…) – acumula em tempt\_esp o “t” tempo de espera do passageiro

- \_\_str\_\_(…) – devolve uma string com formatação de alguns atributos que mostram o estado do balcão.

Exemplo para o balcão 0 com inic\_atend 1 e a fila de espera dos passageiros vazia, devolve:

Balcão 0 tempo 1 : -

Exemplo para o balcão 2 com inic\_atend 2 e a fila de espera com dois passageiros, devolve:

Balcão 2 tempo 2 : - [b:4 t:1] [b:2 t:2] -

Acrescente ainda os métodos que devolvem os valores dos todos os atributos, por exemplo:

- obtem\_n\_balcao(…) – devolve o valor de n\_balcao

- obtem\_fila(…) – devolve o valor da fila

Etc..

**3. Programa de simulação: descrição algorítmica**

O programa de simulação contém dois ciclos:

**3.1.** Primeiro ciclo **for**: Os ciclos de simulação (unidades de tempo de simulação) são executados pelo ciclo **for** o número de vezes indicado na variável de entrada “**ciclos**”. Em cada ciclo de simulação são atendidos passageiros que estejam nas filas dos balcões e determina-se se é ou não inserido um novo passageiro que chega. Este deve ser inserido na fila do balcão que contem menor número de passageiros em espera.

**3.2.** Segundo ciclo **while**: Depois de terminarem todos os ciclos de simulação, um segundo ciclo **while** cuida do atendimento dos restantes passageiros que entretanto chegaram às filas de espera dos balcões, até que todas as filas de espera fiquem vazias.

**4. Detalhes algorítmicos do programa de simulação SIMRAD**

**4.1. Preparação dos dados de entrada**

Implementadas as classes Passageiro e Balcao, a execução do programa SIMPAR começa através da função simpar\_simula(…), com os seguintes dados de entrada:

num\_pass – o número de passageiros com bagagem previsto para este voo

num\_bag – o número máximo de bagagens permitido por passageiro

num\_balcoes – o número de balcões abertos para atendimento e despacho de bagagem

ciclos – os ciclos de tempo em que a simulação decorre.

simpar\_simula(num\_pass, num\_bag, num\_balcoes, ciclos)

Sugestão inicial de dados de entrada:

num\_pass = 70

num\_bag = 4

num\_balcoes = 4

ciclos = 10

Devem preparar-se a entrada destes dados pelo utilizador via teclado, a fim de poder experimentar várias simulações.

**4.2. A função simpar\_simula(…)**

# cria a lista **balcoes** vazia e adicionar todos os objectos balcão com o atributo **n\_balcao** de 0 até **num\_balcoes**-1

# executa os ciclos de simulação, em número de **ciclos**, através do primeiro ciclo **for** indicado em 3.1

# Afixa o número i do ciclo de simulação - exemplo para o ciclo primeiro:

««« CICLO n.º 1 »»»

# Atende os passageiros nos balcões, através da função (adiante descrita):

atende\_passageiros (i, balcoes)

# Decide se cria ou não um novo passageiro e a sua bagagem

Procurar uma forma de gerar um passageiro, de maneira a que a no primeiro terço de ciclos de simulação, a probabilidade seja de 100%, no segundo terço, de 80% e no terço final de 60%.

Se o passageio for gerado, deve gerar-se a sua bagagem, de forma aleatória entre 1 e num\_bag, e procurar o balcão com fila mais pequena para nesta inserir este passageiro.

# Mostra os balcões da lista **balcoes** através da função:

mostra\_balcoes(balcões)

# com o segundo ciclo – while – (indicado em 3.2) atende os passageiros restantes nas filas, após o último ciclo de simulação indicado em **ciclos**, não havendo lugar à chegada (geração) de novos passageiros.

# Antes do ciclo while, afixa “ Fechou a chegada de novos passageiros” e incrementa **ciclos** de 1 unidade

# ciclo **while** enquanto **existem\_balcoes\_com\_fila(**balcoes**)**

**existem\_balcoes\_com\_fila(**balco

es**)** é uma função que devolve True se existirem filas com passageiros ou devolve False se já não existirem passageiras nas filas.

# Afixa o número i do ciclo de simulação - exemplo para o ciclo actual:

««« CICLO n.º <i> »»»

# Atende os passageiros nos balcões, através da função (adiante descrita):

atende\_passageiros (i, balcoes)

# Incrementa i de 1 e refaz o ciclo while

# Mostra os resultados finais da simulação através da função adiante descrita:

apresenta\_resultados(balcoes)

**4.3. A função atende\_passageiros (tempo, balcoes)**

# Para cada elemento **b** da fila **balcoes**, verifica se a fila de **b** não está vazia

# não estando vazia, verifica se o atendimento ao próximo passageiro **p** terminou

# obtém o número de bagagens do passageiro **p**

# calcula o tempo\_de\_atendimento = tempo - **b.** inic\_atend

# calcula as unidades de tempo para despachar o número de bagagens de **p**:

ut\_bag **=** número de bagagens de **p** / **b.**bag\_utemp

# se ut\_bag < tempo\_de\_atendimento:

# o passageiro **p** sai da fila de **b** e calcula o tempo\_de\_espera:

tempo\_de\_espera = tempo – **p.**ciclo\_in

# afixa informação sobre o passageiro **p** , por exemplo:

--- Atendido passageiro com 2 bagagens no balcão 3 com tempo de espera 2

# actualiza dados de **p:**

- muda **inic\_atend** para tempo+1

- incrementa **passt\_atend** de 1

- muda **numt\_bag** acumulando o número de bagagens do passageiro

- muda o **tempt\_esp** acumulando o tempo\_de\_espera do passageiro

- retira o passageiro da fila

# senão (**ut\_bag** >= tempo\_de\_atendimento):

# os balcões com fila vazia actualizam o inic\_atend com **tempo**

**4.4. A função apresenta\_resultados(balcoes)**

Apresenta os resultados estatísticos finais:

# Para cada **i** balcão na fila **balcões:**

# se **passt\_atend** > 0, afixa, por exemplo:

Balcão 0 despachou 4 bagagens por ciclo:

12 passageiros atendidos com média de bagagens / passageiro = 2.0

Tempo médio de espera = 24.0

# senão (**passt\_atend** = 0), afixa, por exemplo:

Balcão 3 não atendeu passageiros

**5. *Output***

Nesta Fase I, o *output* será:

Afixar os resultados da simulação indicados em cada ciclo de simulação.

Afixar os resultados estatísticos finais indicados em 4.4

**Notas finais:**

- Há que encontrar soluções para:

1. Inicialmente colocar passageiros nas filas de espera
2. Controlar que os passageiros previstos para o voo sejam atendidos e que não sejam ultrapassado o num\_pass – o número de passageiros com bagagem previsto para este voo

- Pode haver uma personalização desta solução, que o grupo possa realizar, deste que cumpra os objectivos solicitados.

**FASE II**

**1.Crie um Ficheiro Texto de nome SimOutput** que guarde as informações que são afixadas como resultantes da simulação indicadas no ponto 5 da Fase I.

**2 Crie a árvore BST** **passTree**

a) Insira em **passTree** o nome de cada passageiro (gerado aleatoriamente pelo método infra) que termina o despacho das suas bagagens.

Para este efeito, utilize um método gerador aleatório de nomes (de passageiros).

Considere o módulo “names“:

Para instalá-lo no Anaconda prompt:

pip install names

#Alguns métodos:

import names

names.get\_full\_name()

names.get\_first\_name()

names.get\_last\_name()

names.get\_first\_name(gender = ‘male’)

names.get\_first\_name(gender = ‘female´)

**3.Obtenha informações** segundo as solicitações **à árvore BST criada**:

a) Dado um nome de passageiro via teclado, verificar se este existe na árvore com este nome;

b) Afixar os nomes dos passageiros contidos na árvore, por ordem alfabética, utilizando o algoritmo adequado de atravessamento da árvore.

**DATAS A CONSIDERAR:**

Entrega de AED\_Trab\_FII:  a 5 de Junho

Avaliação Trabalho por grupos: Diurno: 5 a 14 de Junho

                                              PL : 6 a 14 de Junho

**Instruções sobre a constituição dos Grupos e a entrega do Trabalho**

**Instruções sobre a constituição dos Grupos e a entrega do Trabalho**

**Grupos:**

• O Projecto deve ser efectuado em Grupos de 2 a 4 alunos.

**Procedimentos de entrega**

**Material a entregar:**

**a)** Um relatório escrito do **Trabalho-Fase I e Fase II** com número de **páginas até 30**, com letra de tamanho 12, mais um anexo com o código Python desenvolvido, com letra de tamanho 9, e os seguintes pontos:

1. (\*) Uma capa com a identificação do Trabalho: “Trabalho de Algoritmia e Programação”; título do trabalho; data; Universidade; curso; turno (Diurno/Pós-Laboral); turma; número e nome dos alunos/as;

2. Introdução (problema e objectivos a atingir baseado no enunciado);

3. Descrição e explicação das estruturas de dados utilizadas;

4. Descrição sumária do código implementado, indicando o que faz cada um dos módulos (funções / procedimentos);

5. Manual do Utilizador, com imagens dos menus e dos “outputs” com as respetivas explicações;

6. Conclusões, onde deve realçar os aspectos relacionados com a implementação do

Problema:

* 1. Dificuldades sentidas e forma de as ultrapassar, se aplicável;
  2. Indicação clara do que foi e não foi feito em relação ao enunciado do Trabalho;
  3. Funções e tarefas que cada elemento do grupo desempenhou ao desenvolver este Trabalho.

Anexo: listagem do programa fonte (tamanho 9), comentado, em anexo (não conta para as 20 páginas).

O Material a entregar deve ser apresentado em envelope fechado e endereçado ao docente das aulas práticas, a entregar na Sala dos Livros de Sumários, até à sua hora de fecho do dia limite de entrega.

Neste envelope, deve colar uma cópia da capa ou folha de rosto do Trabalho (\*).

**b)** Por *up-load*, no e-L Moodle de AP, por apenas um elemento do Grupo, até ao dia limite de entrega:

- O Relatório do Trabalho;

- Uma pasta com o Projecto Python.

**IMPORTANTE:**

O relatório e o projecto Python, devem ser designados segundo a seguinte sintaxe, para a Fase II:

**D**iurno A1 e A2

AED18DA1 <RF1|PyF2>\_{<InicialPrimeiroNome><UmNomeDeFamília>}

AED18DB2 <RF1|PyF2>\_{<InicialPrimeiroNome><UmNomeDeFamília>}

**P**ós-laboral A1 e A2:

AED18PLA1\_<RF1|PyF2>\_{<InicialPrimeiroNome><UmNomeDeFamília>}

AED18PLA2\_<RF1|PyF2>\_{<InicialPrimeiroNome><UmNomeDeFamília>}

Exemplo:

Relatório: AED18DA1\_RF2\_NFerreiraDGoncaloJBalagoes

Python: AED18DA1\_PyF2\_NFerreiraDGoncaloJBalagoes

**Sobre a avaliação do Trabalho:**

Para que os trabalhos sejam avaliados têm de cumprir os seguintes requisitos:

R1 - Deve ser possível compilar e executar o programa;

R2 - O programa terá que cumprir os requisitos de cada FASE, pelo menos parcialmente;

R3 - Os grupos são constituídos por 2 a 4 pessoas;

R4 – O Trabalho será desenvolvido em Python (Anaconda3), no ambiente do Spyder.

O Trabalho terá avaliação final em prova de apresentação e discussão após a entrega da Fase II, com a presença de todos os alunos de cada Grupo.

**Note bem:**

O Trabalho devem ser originais.

Cópias de outros programas levarão à anulação do Trabalho.

Todos os alunos de cada Grupo devem conhecer bem o código de Python desenvolvido.