CENTRO DE CIÊNCIA E TECNOLOGIA
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS
UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

Aplicação de Filas
Parte 2

Disciplina: Estrutura de Dados I

Prof. Fermín Alfredo Tang Montané

Curso: Ciência da Computação

Aplicações de Filas (Queues)

- Estudaremos uma aplicação de filas:
  - Simulador de Balcão de Passagens Aéreas (Código em Python).

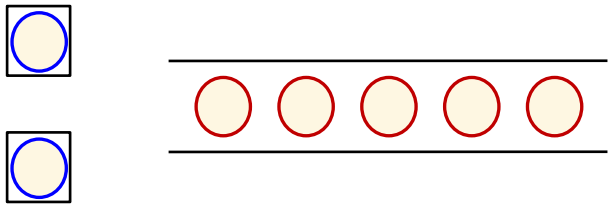
Aplicações de Filas (Queues)
Simulações Computacionais

- Computadores são usados para modelar e simular sistemas do mundo real.
- As simulações são aplicações de computador projetadas para representar e reagir a eventos que acontecem no sistema. O objetivo das simulações pode ser estudar o comportamento do sistema na ocorrência de um evento conhecido ou experimentar a ocorrência de eventos inusuais.
- Dentre as aplicações mais conhecidas temos modelos de previsão de clima e simuladores de vôo. Na área empresarial, simulações computacionais procuram determinar o número de empregados necessários para fornecer um serviço adequado aos clientes.
- Estudaremos o problema de uma linha aérea que deseja saber quantos agentes de vendas de tickets aéreos são necessários no balcão, em um determinado período de tempo, para atender aos clientes com rapidez suficiente.
- Embora a empresa possa estudar os hábitos dos clientes, resulta mais económico e rápido construir um modelo de simulação para o sistema real. Este tipo de sistema pertence a categoria de Sistema de Filas.

Aplicações de Filas (Queues)

Balção de Passagens Aéreas – Sistema de Filas

- Simular um Balcão de Passagens Aéreas significa modelar um sistema de filas (queuing system), onde os clientes permanecem em uma fila esperando pelo serviço fornecido pelos atendentes.
- Um estrutura de fila é usada para modelar este sistema, de forma a estudar certos comportamentos e obter informações de interesse. Dentre as informações mais comuns procura-se saber: o tempo médio de espera na fila e o comprimento médio na fila.
- Sistemas de filas podem considerar fila única ou múltiplas filas.
- Consideraremos um sistema de fila única com vários atendentes.



Atendentes Fila de Clientes

Aplicações de Filas (Queues)
Modelo de Filas - Parâmetros

- Um modelo de um sistema de filas pode ser construído usando simulação de eventos discreta. Neste caso, a simulação é uma sequência de eventos que causam uma mudança no sistema.
- No modelo de Balcão de Passagens Aéreas, a simulação é realizada sobre um período de tempo predefinido. A passagem de tempo é representada por um loop que realiza incrementos sobre uma variável de tempo discreta. Os eventos somente podem acontecer entre intervalos de tempo discreto.
- Utiliza-se minutos como unidade de tempo de forma que não podem acontecer eventos em períodos inferiores a essa unidade de tempo.
- A simulação possui parâmetros definidos pelo usuário, como:
  - Tempo de simulação, dada em número de unidades de tempo.
  - Número de servidores que fornecem atendimento aos clientes.
  - Tempo de serviço esperado para concluir um atendimento.
  - Distribuição do tempo entre chegadas, usada para representar quando os clientes chegam.

Aplicações de Filas (Queues)

Modelo de Filas – Regras para gerenciar eventos

- Um conjunto de regras são definidas para manipular eventos a cada mudança do relógio.
- Para determinar o tempo médio de espera dos clientes antes de serem atendidos considera-se as seguintes três regras:
  - Regra 1.- Se um cliente chega, ele é adicionado a fila. Apenas um cliente pode chegar em cada interação do relógio.
  - Regra 2.- Se há clientes esperando, para cada servidor livre, o próximo cliente na fila inicia o seu atendimento.
  - Regra 3.- Para cada servidor ocupado, se o atendimento foi concluído, o cliente deixa o sistema e o servidor fica ocioso.

Aplicações de Filas (Queues)
Modelo de Filas – Aleatoriedade

- Para o modelo funcionar corretamente, alguns eventos devem ocorrer de forma aleatória. Neste caso a chegada de um cliente deve ser aleatória.
- Uma forma de modelar a chegada consiste em considerar como dado de entrada o tempo médio entre chegadas $(tmec)$. Calcula-se a probabilidade de uma chegada como $(1/tmec)$.
- Para simular uma chegada com esta probabilidade, gera-se um número aleatório r entre 0 e 1. Se o número aleatório é menor o igual que $(1/tmec)$ temos uma chegada. Caso contrário, não temos uma chegada.

Aplicações de Filas (Queues)
Balção de Passagens Aéreas – Implementação

- Implementa-se um sistema de eventos discretos baseado em filas para analisar o tempo médio de espera que os passageiros tem aguardar pelo serviço no Balcão de Passagens Aéreas. Considera-se que existem vários agentes atendendo aos clientes.
- A implementação compreende as classes: Passenger, TicketAgent e TicketCounterSimulation.
- A aplicação recebe parâmetros da seguinte forma:

```
Number of minutes to simulate: 25
Number of ticket agents: 2
Average service time per passenger: 3
Average time between passenger arrival: 2
```

- Produzindo resultados como:

```
Number of passengers served =  12
Number of passengers remaining in line = 1
The average wait time was 1.17 minutes.
```

Aplicações de Filas (Queues)
Balção de Passagens Aéreas – Eventos

- Uma forma de entender melhor a simulação de eventos é fazer listagem dos mesmos em função do tempo.

```
Time   2: Passenger 1 arrived.
Time   2: Agent 1 started serving passenger 1.
Time   3: Passenger 2 arrived.
Time   3: Agent 2 started serving passenger 2.
Time   5: Passenger 3 arrived.
Time   5: Agent 1 stopped serving passenger 1.
Time   6: Agent 1 started serving passenger 3.
Time   6: Agent 2 stopped serving passenger 2.
Time   8: Passenger 4 arrived.
Time   8: Agent 2 started serving passenger 4.
Time   9: Agent 1 stopped serving passenger 3.
Time  10: Passenger 5 arrived.
Time  10: Agent 1 started serving passenger 5.
Time  11: Passenger 6 arrived.
Time  11: Agent 2 stopped serving passenger 4.
Time  12: Agent 2 started serving passenger 6.
Time  13: Passenger 7 arrived.
Time  13: Agent 1 stopped serving passenger 5.
```

```
Time  14: Passenger 8 arrived.
Time  14: Agent 1 started serving passenger 7.
Time  15: Passenger 9 arrived.
Time  15: Agent 2 stopped serving passenger 6.
Time  16: Agent 2 started serving passenger 8.
Time  17: Agent 1 stopped serving passenger 7.
Time  18: Passenger 10 arrived.
Time  18: Agent 1 started serving passenger 9.
Time  19: Passenger 11 arrived.
Time  19: Agent 2 stopped serving passenger 8.
Time  20: Agent 2 started serving passenger 10.
Time  21: Agent 1 stopped serving passenger 9.
Time  22: Agent 1 started serving passenger 11.
Time  23: Passenger 12 arrived.
Time  23: Agent 2 stopped serving passenger 10.
Time  24: Agent 2 started serving passenger 12.
Time  25: Passenger 13 arrived.
Time  25: Agent 1 stopped serving passenger 11.
```

Aplicações de Filas (Queues)
Balção de Passagens Aéreas – Classe Passenger

- A classe **Passenger** (Passageiro) é usada para armazenar e gerenciar as informações referentes a: i) identificação de um passageiro; e ii) tempo de chegada do mesmo. A identificação do passageiro será usada no registro de informações de um evento.
- O tempo de chegada do passageiro será usado para determinar o tempo de espera na fila antes de ser atendido por um agente.

Listing 8.6 The `Passenger` class defined in the `simpeople.py` module.

```
1  # Used to store and manage information related to an airline passenger.
2  class Passenger :
3     # Creates a passenger object.
4    def __init__( self, idNum, arrivalTime ):
5      self._idNum = idNum
6      self._arrivalTime = arrivalTime
7
8     # Gets the passenger's id number.
9    def idNum( self ) :
10     return self._idNum
11
12    # Gets the passenger's arrival time.
13   def timeArrived( self ) :
14     return self._arrivalTime
```

Constructor

Número de identificação

Tempo de chegada

Aplicações de Filas (Queues)

Balção de Passagens Aéreas – Classe TicketAgent

- A classe **TicketAgent** (Agente de Bilhetes) é usada para armazenar e gerenciar as informações referentes ao prestadores de serviço. A informação inclui: i) número de identificação do agente; ii) referencia ao passageiro que está sendo atendido; iii) tempo de finalização do atendimento.
- O tempo de finalização é calculado como a soma do tempo atual mais o tempo médio de atendimento.

Listing 8.7 The `TicketAgent` class defined in the `simpeople.py` module.

```
1  # Used to store and manage information related to an airline ticket agent.
2  class TicketAgent :
3     # Creates a ticket agent object.
4     def __init__( self, idNum ):
5        self._idNum = idNum
6        self._passenger = None
7        self._stopTime = -1
8
9     # Gets the ticket agent's id number.
10    def idNum( self ):
11       return self._idNum
12
13    # Determines if the ticket agent is free to assist a passenger.
14    def isFree( self ):
15       return self._passenger is None
```

Constructor

Número de identificação

Verifica se o agente está livre

Aplicações de Filas (Queues)
Balção de Passagens Aéreas – Classe TicketAgent

- O método **IsFinished()** verifica se o passageiro que esta sendo atendido já concluiu o seu atendimento.
- O método **startService()** serve para que o agente inicie um novo atendimento, registrando: i) uma referencia ao passageiro atendido; e ii) o tempo de finalização do serviço; ambos passados como parâmetros.
- O método **stopService()** serve para finalizar o atendimento do agente, retornando a referencia ao passageiro atendido e fazendo que o campo _passenger tenha novamente uma referência nula.

```
16
17    # Determines if the ticket agent has finished helping the passenger.
18    def isFinished( self, curTime ):
19      return self._passenger is not None and self._stopTime == curTime
20
21    # Indicates the ticket agent has begun assisting a passenger.
22    def startService( self, passenger, stopTime ):
23      self._passenger = passenger
24      self._stopTime = stopTime
25
26    # Indicates the ticket agent has finished helping the passenger.
27    def stopService( self ):
28      thePassenger = self._passenger
29      self._passenger = None
30      return thePassenger
```

Verifica o fim do atendimento

Inicio de Atendimento

Fim de Atendimento

Aplicações de Filas (Queues)
Simulador - Balção de Passagens Aéreas

- A figura ilustra os objetos **Passenger** (Passageiro) e **TicketAgent** (Agente de Bilhetes)

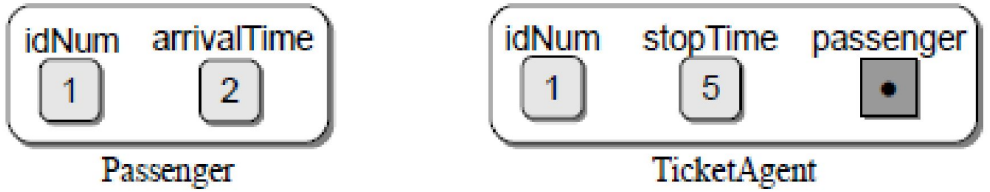


Figure 8.12: Sample `Passenger` and `TicketAgent` objects.

Aplicações de Filas (Queues)

Balção de Passagens Aéreas – Classe TicketCounter Simulation

- A classe **TicketCounterSimulation** permite gerenciar a simulação de venda de tickets aéreos. Esta classe possui utiliza vários módulos:
  - i) array.- que implementa o uso da classe Array;
  - ii) llistqueue.- que implementa o uso da TAD Fila (Queue);
  - iii) people.- que implementa as classes TicketAgent e Passenger.

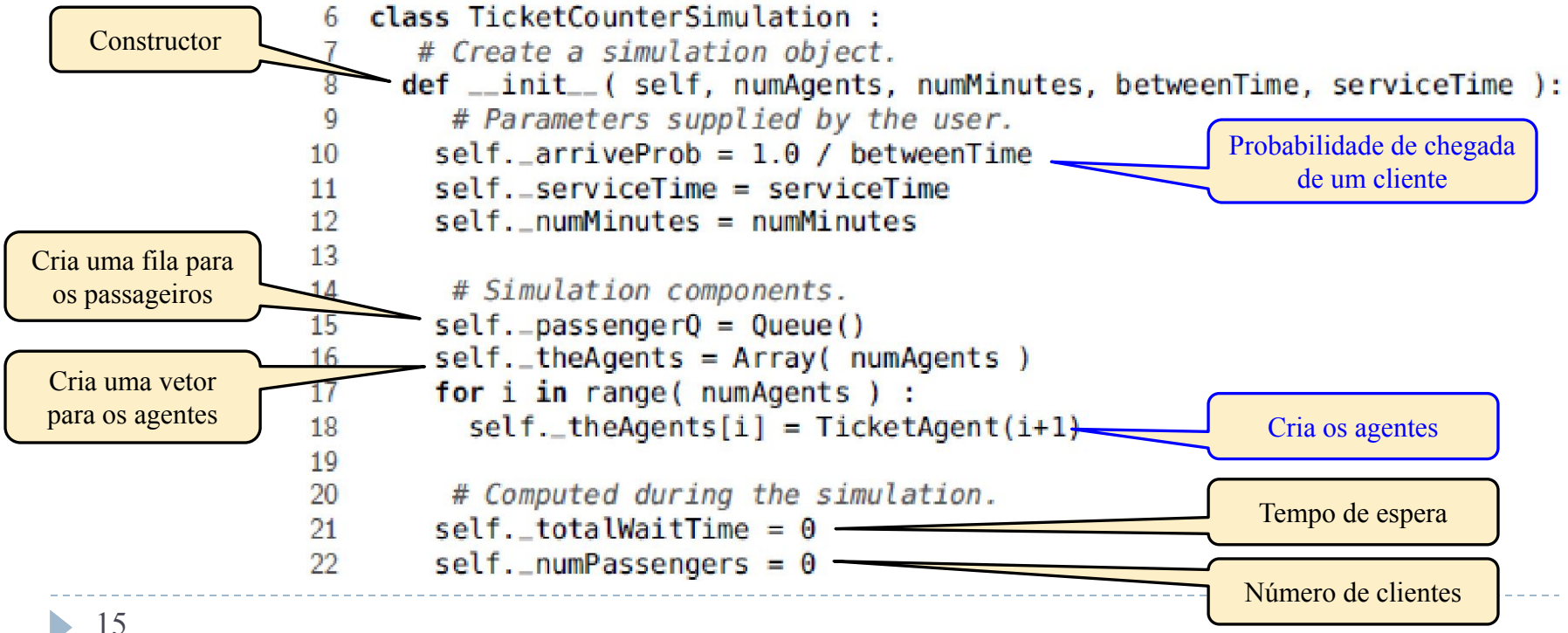
Listing 8.8 The `simulation.py` module.

```
1  # Implementation of the main simulation class.
2  from array import Array
3  from llistqueue import Queue
4  from people import TicketAgent, Passenger
5
```

Aplicações de Filas (Queues)

Balção de Passagens Aéreas – Classe TicketCounter Simulation

- A classe **TicketCounterSimulation** possui vários parâmetros:
  - i) número de agentes (numAgents); ii) minutos de simulação(numMinutes); iii) tempo entre chegadas (betweenTime) e iv) tempo de serviço (serviceTime).
- O construtor inicializa os campos usados na simulação; cria uma fila para armazenar os passageiros; e um vetor para gerenciar os agentes; e inicializa os campos que armazenam os resultados da simulação.




```
6  class TicketCounterSimulation :
7    # Create a simulation object.
8    def __init__( self, numAgents, numMinutes, betweenTime, serviceTime ):
9      # Parameters supplied by the user.
10     self._arriveProb = 1.0 / betweenTime
11     self._serviceTime = serviceTime
12     self._numMinutes = numMinutes
13
14     # Simulation components.
15     self._passengerQ = Queue()
16     self._theAgents = Array( numAgents )
17     for i in range( numAgents ) :
18       self._theAgents[i] = TicketAgent(i+1)
19
20     # Computed during the simulation.
21     self._totalWaitTime = 0
22     self._numPassengers = 0
```

Aplicações de Filas (Queues)

Balção de Passagens Aéreas – Classe TicketCounter Simulation

- A figura ilustra o objeto **TicketCounterSimulation** (Simulador de Contador de Bilhetes).

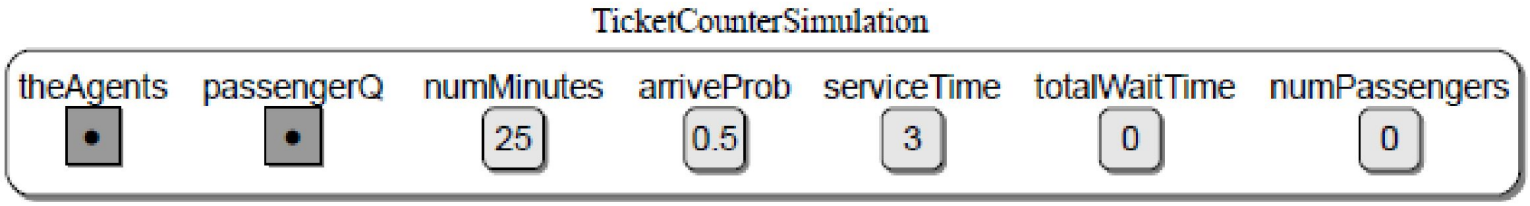


Figure 8.13: A sample `TicketCounterSimulation` object.

Aplicações de Filas (Queues)

Balção de Passagens Aéreas – Classe TicketCounter Simulation

- A simulação é executada chamando ao método **run()** que simula a passagem de tempo (em minutos) mediante um loop controlado por um contador discreto (curTime). A simulação termina quando o contador do loop atinge o valor de _numMinutes.
- Em cada iteração do loop, são gerenciadas três regras de simulação: i) o tratamento das chegadas; ii) o tratamento do inicio de serviço; ii) o tratamento do fim de serviço.
  - O método **_handleArrival()** determina se um passageiro chegou no momento atual e gerencia a sua chegada.
  - O método **_handleBeginService()** determina se há algum agente livre e permite que o próximo passageiro na fila comece a seu atendimento.
  - O método **_handleEndService()** determina se algum atendimento terminou e sinaliza a saída do cliente.

```
23
24      # Run the simulation using the parameters supplied earlier.
25    def run( self ):
26      for curTime in range(self._numMinutes + 1) :
27        self._handleArrival( curTime )
28        self._handleBeginService( curTime )
29        self._handleEndService( curTime )
30
```

Aplicações de Filas (Queues)
O método printResults()

- O método **printResults()** imprime os resultados da simulação de eventos discretos para o sistema de venda de Tickets Aéreos. Para isso realiza o seguintes cálculos:
  - Calcula o número total de clientes atendidos (numServed), como a diferença entre o número total de passageiros e o comprimento da fila (len(self._passengerQ)).
  - Calcula o número médio de espera (avgWait: average wait), como o tempo total de espera dividido pelo número total de clientes.
  - Imprime os três valores: numServed, len(self._passengerQ), avgWait.

Total de clientes atendidos

Tempo médio de espera

Comprimento da Fila

```
31      # Print the simulation results.
32     def printResults( self ):
33         numServed = self._numPassengers - len(self._passengerQ)
34         avgWait = float( self._totalWaitTime ) / numServed
35         print( "" )
36         print( "Number of passengers served = ", numServed )
37         print( "Number of passengers remaining in line = %d" %
38                len(self._passengerQ) )
39         print( "The average wait time was %4.2f minutes." % avgWait )
```

Aplicações de Filas (Queues)
Simulador - Balção de Passagens Aéreas

- Os métodos **_handleArrive(), _handleBeginService(), _handleEndService()** precisam ser implementados.

```
40
41    # The remaining methods that have yet to be implemented.
42    # def _handleArrive( curTime ):        # Handles simulation rule #1.
43    # def _handleBeginService( curTime ):  # Handles simulation rule #2.
44    # def _handleEndService( curTime ):    # Handles simulation rule #3.
```

Aplicações de Filas (Queues)
Simulador - Balção de Passagens Aéreas

- A tabela mostra os resultados de várias simulações para o sistema de venda de Tickets Aéreos.

- Foram feitos três grupos de simulações, onde se varia o tempo da simulação em minutos: 100, 500, 1000, 5000, 10000.
- Além disso, varia-se o numero de agentes , o tempo médio de serviço.
- Observa-se que o último grupo revela a melhor escolha.

Num Minutes	Num Agents	Average Service	Time Between	Average Wait	Passengers Served	Passengers Remaining
100	2	3	2	2.49	49	2
500	2	3	2	3.91	240	0
1000	2	3	2	10.93	490	14
5000	2	3	2	15.75	2459	6
10000	2	3	2	21.17	4930	18
100	2	4	2	10.60	40	11
500	2	4	2	49.99	200	40
1000	2	4	2	95.72	400	104
5000	2	4	2	475.91	2000	465
10000	2	4	2	949.61	4000	948
100	3	4	2	0.51	51	0
500	3	4	2	0.50	240	0
1000	3	4	2	1.06	501	3
5000	3	4	2	1.14	2465	0
10000	3	4	2	1.21	4948	0

Table 8.1: Sample results of the ticket counter simulation experiment.

Referências

- Rance Necaise. Data Structures and Algorithms Using Python. Capítulo 8. Queues. 2011.