Queue

TEAM B| DATA STRUCTURES & ALGORITHMS

Case Study | Clinic Simulation

- Write a python program that simulates a doctor clinic from 12 PM to 4 PM on any average day , about 10 patients arrive to the clinic at any given hour , the patients ages typically are in between 20 and 60 , each age from 20 to 60 is equally likely . the clinic deals with the patients by a first-come first served manner . patients time with the doctor is calculated by the following formula : patient time with doctor = age / 5 minutes . the output of the simulation are the average waiting time of the patients and the number of remainning patients at 4 PM . if patient time with doctor changes to age / 10 minutes . what are the average waiting time of the patients and the number of remaning patients at 4 PM.

Objectives

- Queues
- Applications
- Implementation
- Clinic Simulation
- Run #Code

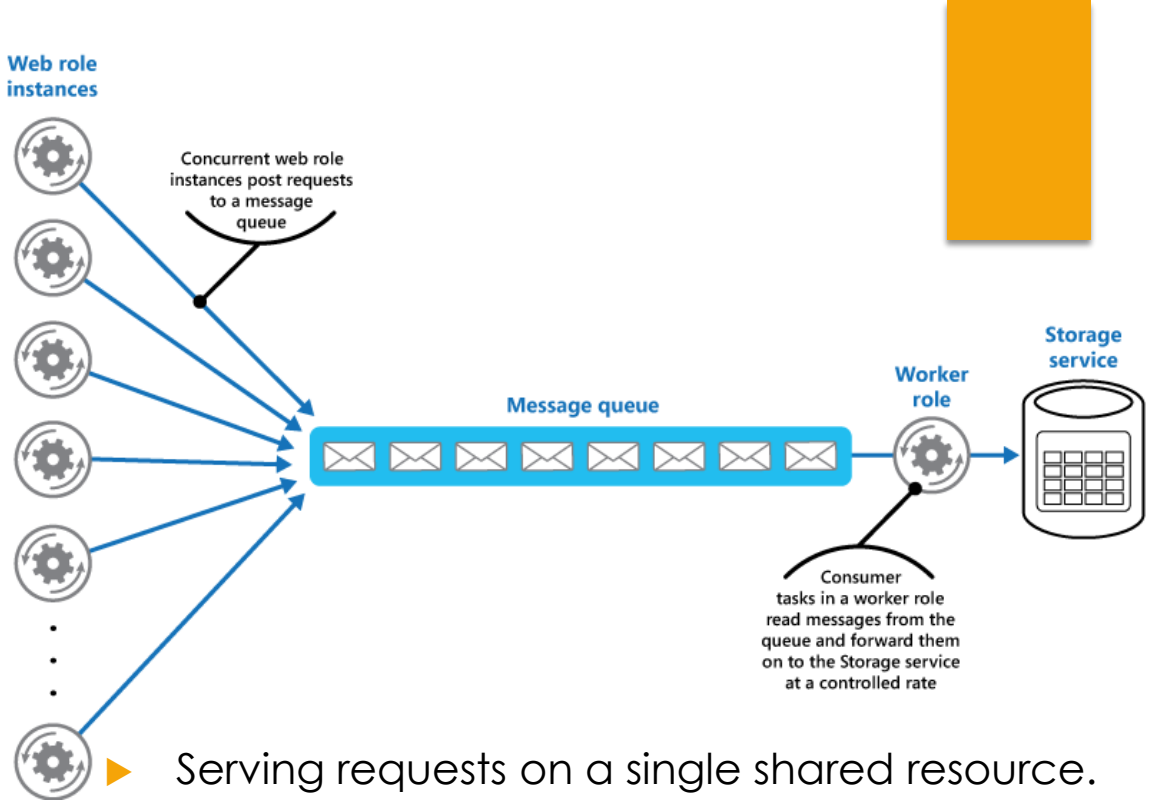
What's a Queue?

- Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is open at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

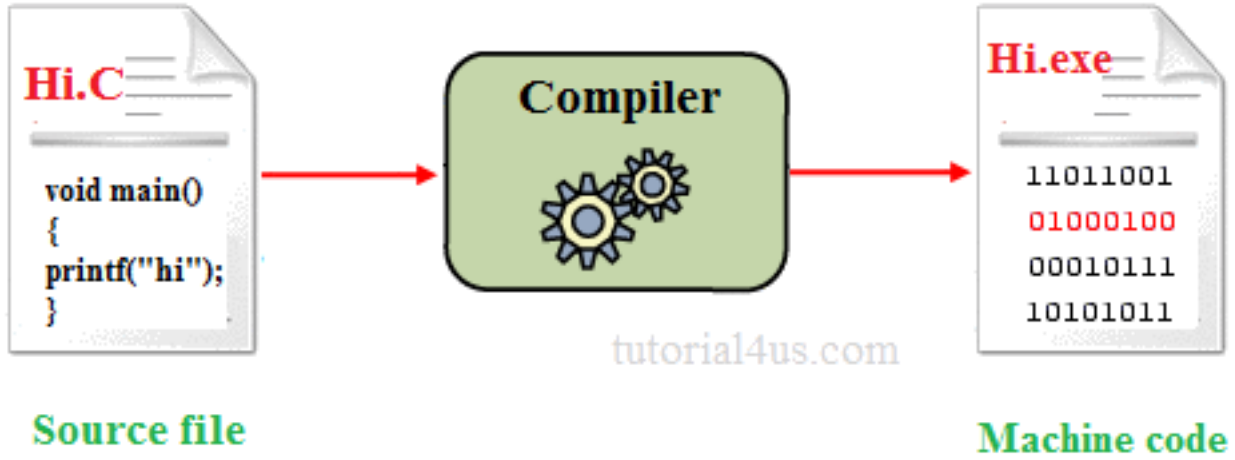
- A real-world example of queue can be a single-lane one-way road, where the vehicle enters first, exits first. More real-world examples can be seen as queues at the ticket windows and bus-stops.

Applications of Queue

- Queue, as the name suggests is used whenever we need to manage any group of objects in an order in which the first one coming in, also gets out first while the others wait for their turn, like in the following scenarios :
- Serving requests on a single shared resource, like a printer, CPU task scheduling etc.
- In real life scenario, Call Center phone systems uses Queues to hold people calling them in an order, until a service representative is free.
- Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive i.e First come first served.



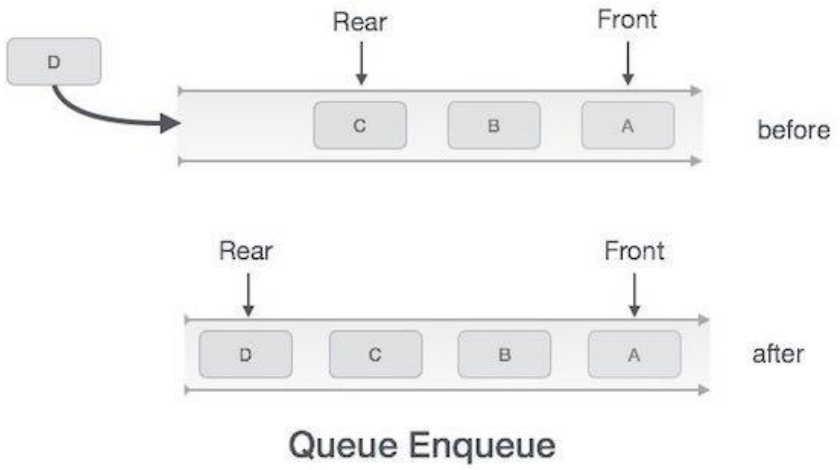
- Serving requests on a single shared resource.



- Compiling the source file into object file line by line

Queue Implementation

- Queue operations may involve initializing or defining the queue, utilizing it, and then completely erasing it from the memory. Here we shall try to understand the basic operations associated with queues −
- **enqueue()** − add (store) an item to the queue.
- **dequeue()** − remove (access) an item from the queue.
- Few more functions are required to make the above-mentioned queue operation efficient. These are −
- **peek()** − Gets the element at the front of the queue without removing it.
- **isfull()** − Checks if the queue is full.
- **isempty()** − Checks if the queue is empty.



Simulation : Clinic Queue

Case Study : Clinic Simulation (1)

- Patients go to the doctor in the clinic.
- The patients are placed in a queue to be examined in a first – come first – served manner.
- On any average day about 10 patients are available in the clinic at any given hour.
- The ages of each patient in between 20 and 60 ages.
- Each patient can take a time equals to his (Age / 5) or (Age / 10).

Case Study : Clinic Simulation (2)

- We could decide by building a simulation that models the clinic.
- As patients keep coming we will add them to a waiting list , a queue of patients waiting to be examined.
- When the doctor finishes a patient , he will look at the queue to see if there are remaining patients.
- The Average amount of time patients will wait to be examined (= the average amount of time they are waiting in the queue).
- To handle and model this situation we need to use some

Probabilities.

Case Study : Clinic Simulation (3)

- If each patient his age is from 20 to 60 is equally likely , the actual time for the doctor takes can be simulated by using a random number between 20 and 60 inclusive.
- This means that there is equal chance of any given time from 20 to 60 appearing.
- If there are 10 patients on average day at any given hour

and the clinic is available from 12 PM to 4 PM (4 hours).

- That means that on average there will be 40 patients during the 4 hours and one patient every 6 minuets.
- 40 / 4 hours = 40 patient / 240 minuets = 1 patient / 6 minuet
- For every 6 minuets , we can simulate the chance that there is a new patient by generating a random number from 1 to 6 inclusive.
- If the number is 6 , we can say a patient has been come.

Main Simulation Steps

- Create a queue of patients , each patient will be given a timestamp for his arrival and the queue is empty to start.
- For each minuet (CurrentMinuet):
  - Does a new patient come ? If so , add him on the queue with the CurrentMinuet as timestamp.
  - If the doctor is not busy and if a patient is waiting,

    - Remove the next patient from the queue and assign him to the doctor.

    - Subtract the timestamp from the CurrentMinuet to compute the waiting time for that patient.

    - Append the waiting time for the patient to a list for later processing.

    - Based on the age of the patient being examined by the doctor, figure out how much time will be required (remaining patients time)
  - The doctor is now busy and the time is subtracted one minuet from the time required
  - If the patient has finished , in other words the time has reached zero , the doctor is no longer busy.
- After the simulation is completed , compute the average waiting from the list of waiting times generated. Create a queue of patients , each patient will be given a timestamp for his arrival and the queue is empty to start.

#CODE

Clinic Queue Simulation – Doctor Class I

```
class Doctor:
    def __init__(self):
        self.currentPatient = None
        self.TimeRemaining = 0

    def tick(self):
        if self. currentPatient != None:
            self.TimeRemaining = self.TimeRemaining - 1
            if self.TimeRemaining <= 0:
                self. currentPatient = None
```

Clinic Queue Simulation – Doctor Class II

```
def busy(self):
    return self.currentPatient != None

def startNext(self, nextPatient):
    self.currentPatient = nextPatient
    self.TimeRemaining = nextPatient.getPatients()/5
```

Clinic Queue Simulation – Patient Class

```
Import random
Class Patient:
    def __init__(self,time):
        self.timeStamp = time
        self.patients = random.randrange(20,61)
    def getPatients(self):
        return self.patients
    def getTimeStamp(self):
        return self.timeStamp
    def waitTime(self, currentTime):
        return currentTime - self.timeStamp
```

Clinic Queue Simulation – Main Simulation I

```
From pythonds.basic.queue import Queue
From Doctor import *
From Patient import *
Import random
Def simulation (numMinutes) :
        myClinic = Doctor()
        myQueue = Queue()
        waitingTimes = []

    for currentMin in range (numMinutes):
```

Clinic Queue Simulation – Main Simulation II

```
        If random.randrage (1,7)==6:
            patient = Patient(currentMin)
            myQueue.enqueue(patient)
        if (not myClinic.busy()) and (not myQueue.isEmpty()):
            nextpatient = myQueue.dequeue()
            waitingtimes.append ( nextpatient.waitTime(currentMin))
            myClinic.startNext(nextpatient)


        myClinic.tick()
averageWait = sum (waitingtimes) / len(waitingtimes)
print ("Average Wait " , averageWait ," Minutes" ,myQueue.size() , "
patients remaining.")
```

Think Like A Geek !

Optimization & Modification

Think Like A Geek !

Optimization & Modification

- Types of clinics
- Doctor time of arrival and leaving
- Break time
- Real Life Probability

```
clinictype = input("please choose the type of your clinic\n"
                   "press 'A' for belly clinic\n"
                   "press 'B' for Ophthalmology\n"
                   "press 'C' for Orthopedic\n"
                   "press 'D' for surgery\n"
                   "press 'E' for dental\n"
                   "press 'F' for Dermatology\n"
                   "press 'G' for Brain and Neurology\n"
                   "press 'H' for Heart\n"
                   "press 'I' for Nose, ear and throat\n"
                   "press 'J' for Psychiatric\n"
                   "press 'K' for Oncology : \n")
```

Clinic Types

```
time_arrival = int(input("Arrival hour of the doctor is (in 24-hour Format): "))
time_leaving = int(input("Leaving hour of the doctor will be (in 24-hour Format): "))
numMins = (time_leaving - time_arrival)*60

youngRateWithDoctor = int(input("Younger patients rate: "))
oldRateWithDoctor = int(input("OLDER patients rate: "))
if(numMins <= 0):
    print("Please enter a valid format")
else:

    Break = int(input("input time you like to take as a break in minutes : "))

    for i in range(10):
        simulation(numMins,youngRateWithDoctor,oldRateWithDoctor,Break)
```

Doctor arrival and leaving time

```
time_arrival = int(input("Arrival hour of the doctor is (in 24-hour Format): "))
time_leaving = int(input("Leaving hour of the doctor will be (in 24-hour Format): "))
numMins = (time_leaving - time_arrival)*60

youngRateWithDoctor = int(input("Younger patients rate: "))
oldRateWithDoctor = int(input("OLDER patients rate: "))
if(numMins <= 0):
    print("Please enter a valid format")
else:

    Break = int(input("input time you like to take as a break in minutes : "))

    for i in range(10):
        simulation(numMins,youngRateWithDoctor,oldRateWithDoctor,Break)
```

Doctor arrival and leaving time, Rate and Break (Inputs)

Break Time

```
halfTime = simulationTime // 2
if currentMin >= halfTime  and currentMin <= halfTime + breakTime:
    continue
```

```
if random.randrange(0, 6) == 4:                # A patient every 6 Minutes

    currentPatient = Patient(currentMin)

    if clinictype in "CDFGHcdfgh":             #to Minimize the probability

        if myTask.getPatientsAge() in range(20, 41):
            if random.randrange(1, 3) == 1:
                myClinicQueue.enqueue(currentPatient)
            continue
        else:
            myClinicQueue.enqueue(currentPatient)

    else:
        myClinicQueue.enqueue(currentPatient)

halfTime = simulationTime // 2
if currentMin >= halfTime  and currentMin <= halfTime + breakTime:
    continue
```

Real time probability

Real Life Probability Explanation

- We all know for sure that the probability for some clinics like " heart , bones and so on is not equally likely with old and young men.
- So we made the probability of young one is more difficult than the old on.
- Hence , if the user chooses a clinic that has a high rate of old men to go to , we will look to their age if they are young we would make it difficult for them to be choose so we made 2 times probabilities for them , and if there is an old man we would just add him on Queue.
- Now we can say that we made a real life probability concept with different clincs.

Let’s run the #code

THANK YOU !

Team Members

Mahmoud Essam - Toka Ashraf - Abd ElRahman Salim
Alaa Fathalla - Ayman Samir - Toka Kamel - Ahmed Wafy - Abeer Zeina
Sara Ghoniem - Youmna AbdelGawad - Mahmoud Youssef