Assignment 02

Zihan wei 2018124079

Analysis

To finish this problem, we need to consider the following scenario:
First, when a new customer is coming, if there is no customer in queue and the corresponding service has empty counter, he can directly go to the counter;
Second, if the corresponding service has no empty counter but the other service has empty counter, he can go to the other service;
Third, if all counters are busy or there is already someone in two queues, he need to line up based on his type of service.
In order to simulate this process, we need a two-dimensional array to represent the queue (queue = [[], []]). Queue[0] is for service 1 and queue[1] is for service 2.
Then we need a function for the new customer:

```
def inQueue(element):
    global maxline
    haveposition = False     # a flag to justify if the customer is served
    server = int(element[-1])-1
    if len(queue[server]) == 0:  # if there is no people in queue (new customer's type of service)
        for index, servertime in enumerate(time[server]):
            if servertime <= int(element[0]):  # the new customer can be served
                time[server][index] = int(element[1]) + int(element[0])
                flag[server][index] = False
                num_of_people[server][index] += 1
                haveposition = True
                idle[server][index] += int(element[0]) - servertime
                break
    if haveposition is False: # the new customer is not served
        if len(queue[server-1]) == 0: # justify if he can go to another service
            for index, servertime in enumerate(time[server-1]):
                if servertime <= int(element[0]):  # has empty position
                    time[server-1][index] = int(element[1]) + int(element[0])
                    flag[server-1][index] = False
                    num_of_people[server-1][index] += 1
                    haveposition = True
                    idle[server-1][index] += int(element[0]) - servertime
                    break
    if haveposition is False: # the new customer is still not served
        queue[server].append(element)   # be in queue
        if maxline < len(queue[0]) + len(queue[1]):
            maxline = len(queue[0]) + len(queue[1])
    return
```

This function covers the three points above.
Then we should consider what will happen between two arrival time of customers. Every work has its corresponding service time. If it has spent the corresponding length of time, this work is finished and the counter will be empty so that the counter can serve a new customer. If the queue is mot empty at that time, for example, queue[0] is not empty and one counter of service 1 is empty, the first one of queue[0] can go to use the counter. He will leave the queue. Also, we need to think that service 2 has empty counter

while service 1 has no empty counter. If queue[1] is empty, the customer in queue[0] can use counter of service 1.
Because every second may have finished work, we should use time to simulate this process. Every second we will update the status:

```
for i in range(0, 5000):
    updatequeue(i)      # update the queue at this time
    for line in lines:
        tmp = line.rstrip().split(' ')
        if int(tmp[0]) == i:   # if there is a new customer coming at this time
            inQueue(tmp)
            break
        elif int(tmp[0]) > i:
            break
```

Every second we update the queue and check if there is a new customer arrived. If true, execute the function inQueue().
In function updatequeue(), if one counter's work is finished at that time, we will set the label "true" which means it is empty now. Then it can select a new customer to serve.

```
def updatequeue(i):
    global maxqtime, avaqtime, waitnum
    if len(queue[0]) == 0 and len(queue[1]) == 0:  # no customer is in queue
        return
    for index, item in enumerate(time[0]):  # update the status of each counter at this time
        if i >= item:  # if the work is finished
            flag[0][index] = True
    for index, item in enumerate(time[1]):
        if i >= item:
            flag[1][index] = True
    for index, item in enumerate(flag[0]):
        if item is True:   # if there is a empty counter, select a new customer who is waiting
            if len(queue[0]) != 0:  # select as many people as possible in queue 1
                time[0][index] = i + int(queue[0][0][1])
                flag[0][index] = False
                num_of_people[0][index] += 1
                avaqtime += i - int(queue[0][0][0])
                if maxqtime < i - int(queue[0][0][0]):
                    maxqtime = i - int(queue[0][0][0])
                queue[0].pop(0)                    # First customer out of the queue
                waitnum += 1
```

That is a part of this function.
Each service preferentially serves the customers of the corresponding queue. When service 1 or service 2 cannot serve corresponding customers, we should check if the other service has opportunity to continue serve customer. The conditions are: the other queue is empty and the other service has empty position.

```
if len(queue[0]) == 0 and len(queue[1]) != 0:  # justify if customer can go to service 1
    for index, item in enumerate(flag[0]):
        if item is True:
            time[0][index] = i + int(queue[1][0][1])
            num_of_people[0][index] += 1
            flag[0][index] = False
            avaqtime += i - int(queue[1][0][0])
            if maxqtime < i - int(queue[1][0][0]):
                maxqtime = i - int(queue[1][0][0])
            queue[1].pop(0)
            waitnum += 1
```

That is one part of this function.
Also, we use two-dimensional array to record the finished time, status and customer number of each server.

The result is:

```
Number of people served for each server
7
5
3
6
6
3
finished time of each server
3792
4031
4197
4229
4182
3791
Average time a customer spends in queue
174.0
max time a customer spends in queue
307
max line
4
total idle time for each server
1084
1786
2881
1325
1513
2437
```