Operating system homework 1

CPU scheduling algorithm

นาย ดรันภพ เป็งคำตา 580610642

แบบแผนในการทดลอง

ในการทดลองนี้ จะเป็นการทำการทดสอบ algorithm ในการจำลองการจัดลำดับการทำงานของโพรเซส โดยจะทำการทดสอบทั้งหมด 3 algorithms ซึ่งก็คือ

- **First Come First Serve**
- **Short Jobs First**
- **Round Robin**
    - Quantum time = 5
    - Quantum time = 10
    - Quantum time = 20

โดยแต่ละ algorithm จะถูกทดสอบกับชุดข้อมูล 3 ชุด ที่มีการกระจายตัวของข้อมูลที่ต่างกัน และ วัดผลการทำงานด้วยค่า **Average waiting time** ซึ่งจะเป็นการวัดเวลาในการรอประมวลผลเฉลี่ยของทุกโพสเซส โดยไม่คำนวณเวลาที่เข้ามา(Arrival time) ของแต่ละโพรเซส แต่จะคำนวณเฉพาะเวลาในการรอทำงานเท่านั้น ซึ่งหากมีค่าดังกล่าวน้อย จะแสดงว่า algorithm นี้มีการทำงานที่ดีกว่า algorithm ที่ให้ค่ามากกว่าสำหรับชุดข้อมูลชุดเดียวกัน

รูปแบบของชุดข้อมูล

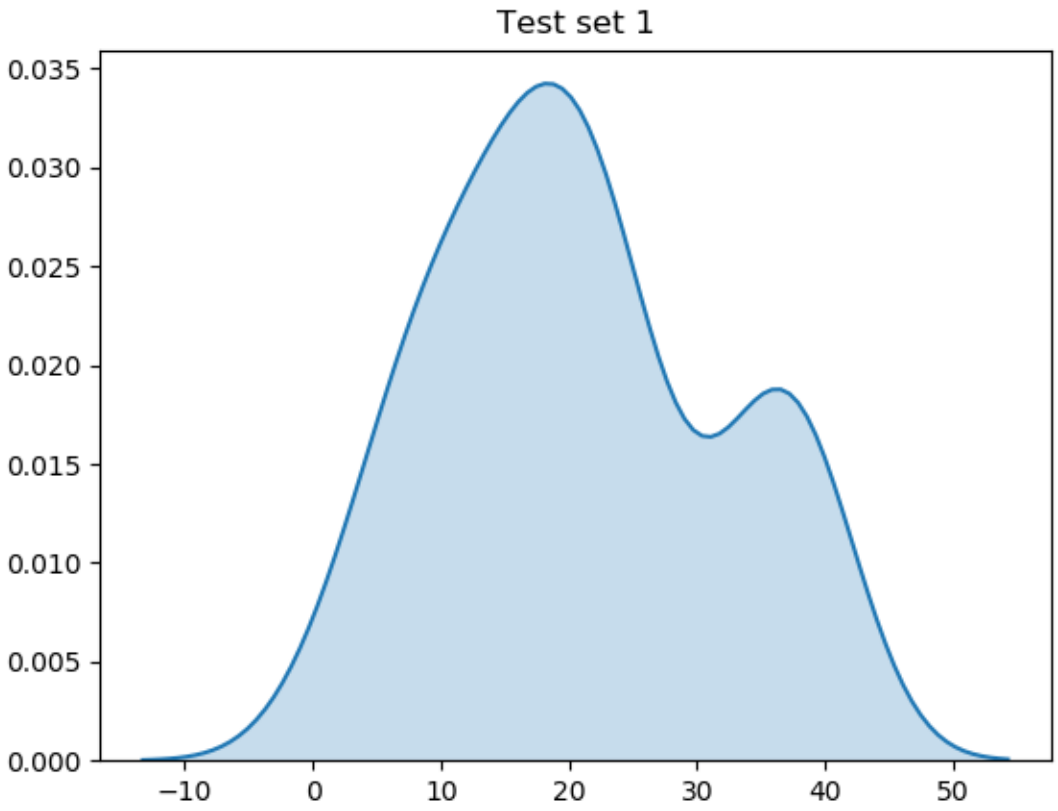
ภายในแต่ละชุดข้อมูลจะประกอบด้วยตัวเลขจำนวนเต็ม ซึ่งมีค่าในช่วง [1 , 40] โดยแสดงถึงเวลาในการทำงานของแต่ละโพรเซส ค่าข้อมูลของแต่ละชุดข้อมูลจะเป็นค่าที่ถูกสุ่มมาในช่วงดังกล่าว แต่จะถูกกำหนดช่วงค่าไว้ 3 ช่วง คือ **สูง**(เวลาในการทำงาน 30-40), **กลาง**(เวลาในการทำงาน 15-25) และ **ต่ำ**(เวลาในการทำงาน 1-10) แต่ละชุดข้อมูลจะถูกกำหนดจำนวนค่าของแต่ละช่วงไว้ เพื่อให้เกิดการกระจายตัวของข้อมูลที่แตกต่างกันไป เพื่อทดสอบว่า “การกระจายตัวของข้อมูลส่งผลต่อการทำงานของแต่ละ algorithm”

ชุดข้อมูลที่ 1

17, 8, 24, 8, 8, 15, 35, 16, 15, 5, 21, 36, 30, 34, 25, 10, 38, 4, 34, 34,
24, 18, 9, 6, 15, 23, 40, 39, 10, 15, 1, 19, 36, 17, 22, 10, 20, 10, 37, 10,
20, 25, 17, 24, 18, 22, 38, 20, 22, 15, 3, 15, 10, 15, 24, 24, 38, 39, 22, 39

โพรเซส สูง จำนวน: 15 โพรเซส กลาง จำนวน: 30 โพรเซส ต่ำ จำนวน: 15

จำนวน 60 โพรเซส



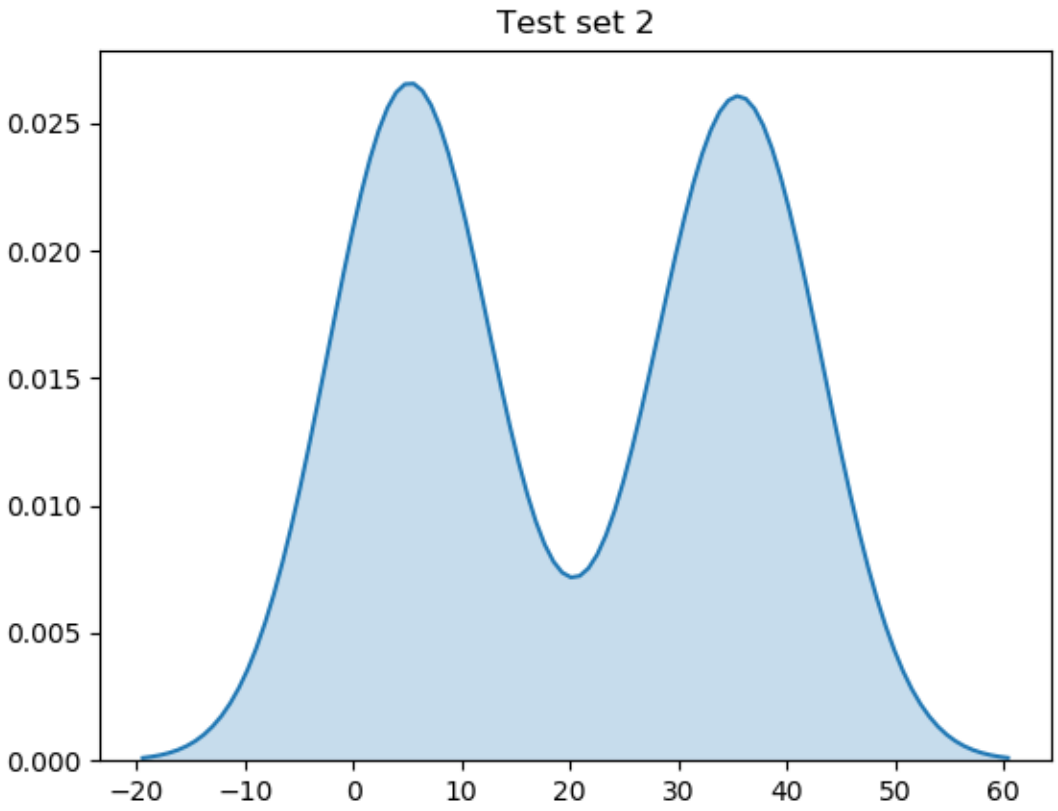
Histogram แสดงการกระจายตัวของชุดข้อมูลที่ 1

ชุดข้อมูลที่ 2

2, 6, 38, 40, 39, 40, 31, 3, 8, 35, 39, 4, 39, 31, 3, 36, 2, 36, 6, 31,
33, 2, 35, 9, 7, 34, 1, 40, 33, 9, 38, 1, 2, 7, 39, 35, 36, 33, 8, 32,
7, 4, 10, 37, 3, 34, 10, 5, 6, 10, 3, 3, 30, 40, 1, 38, 8, 8, 30, 32

โพรเซส สูง จำนวน: 30 โพรเซส กลาง จำนวน: 0 โพรเซส ต่ำ จำนวน: 30

จำนวน 60 โพรเซส



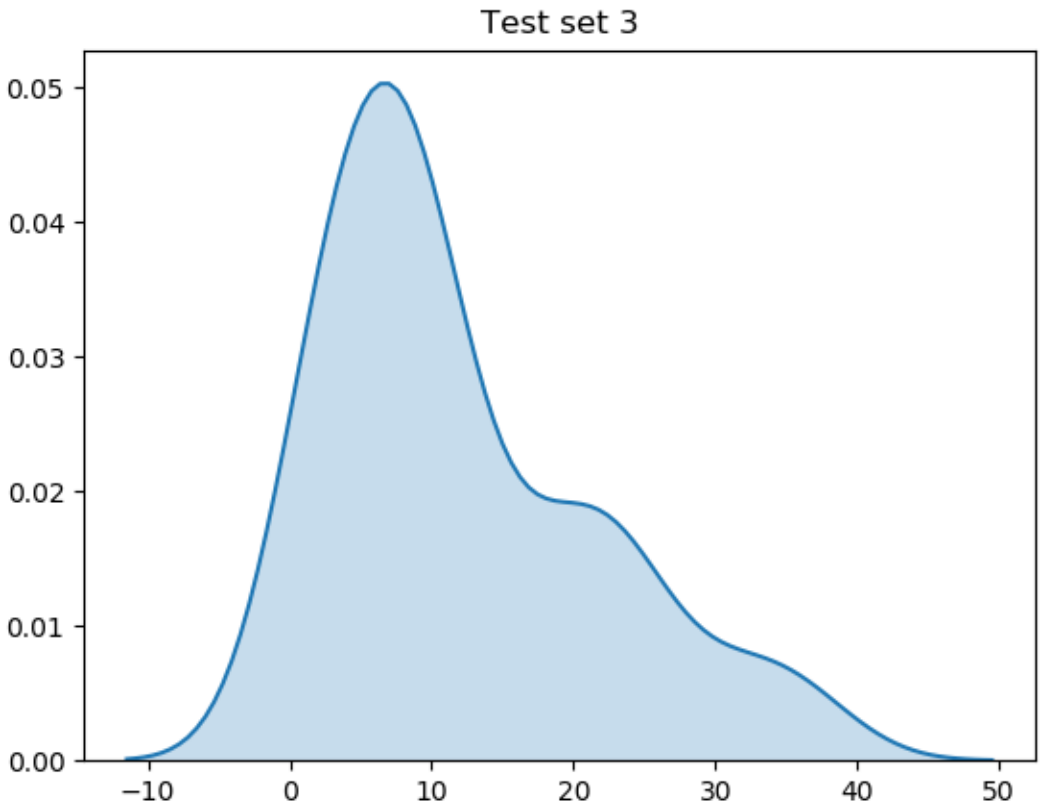
Histogram แสดงการกระจายตัวของชุดข้อมูลที่ 2

ชุดข้อมูลที่ 3

8, 20, 2, 10, 37, 10, 32, 7, 3, 36, 7, 3, 3, 6, 23, 3, 4, 3, 3, 9,
15, 22, 3, 1, 3, 7, 9, 8, 3, 7, 9, 9, 19, 1, 10, 21, 10, 19, 15, 24,
10, 9, 24, 4, 7, 17, 33, 24, 2, 10, 31, 6, 25, 9, 8, 4, 17, 23, 9, 10

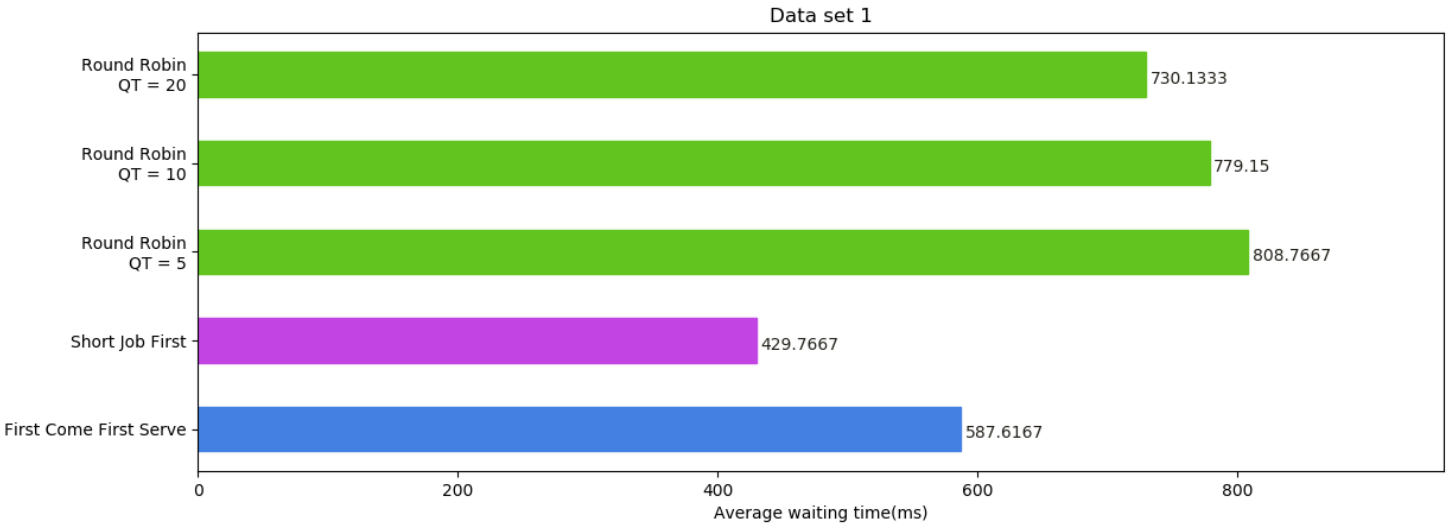
โพรเซส สูง จำนวน: 5 โพรเซส กลาง จำนวน: 15 โพรเซส ต่ำ จำนวน: 40

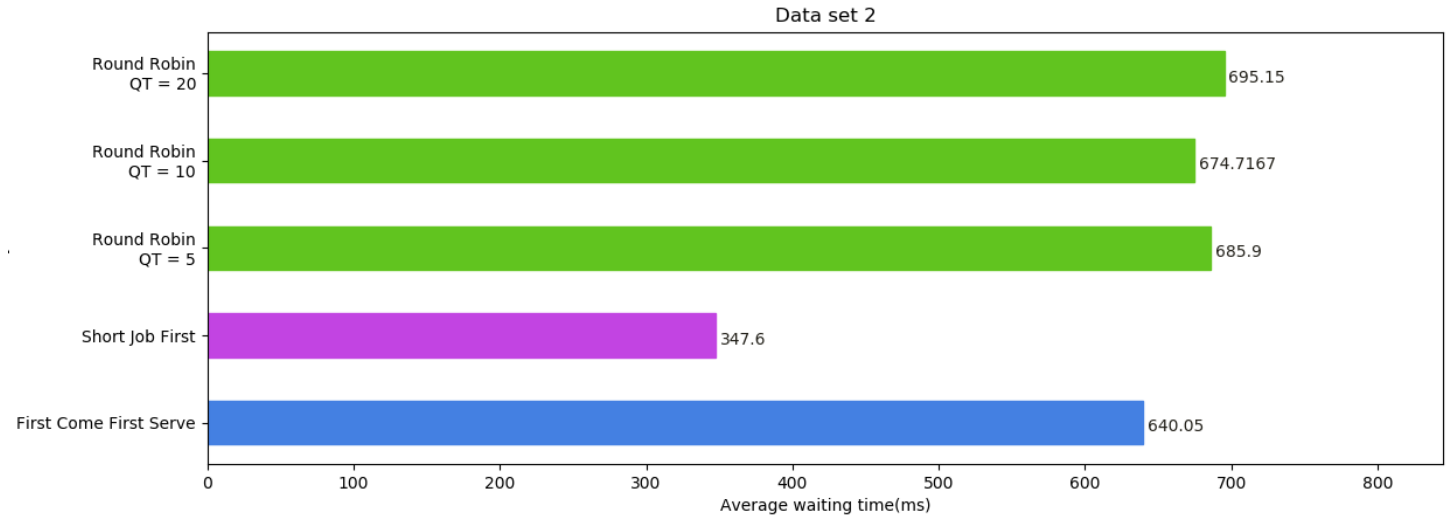
จำนวน 60 โพรเซส

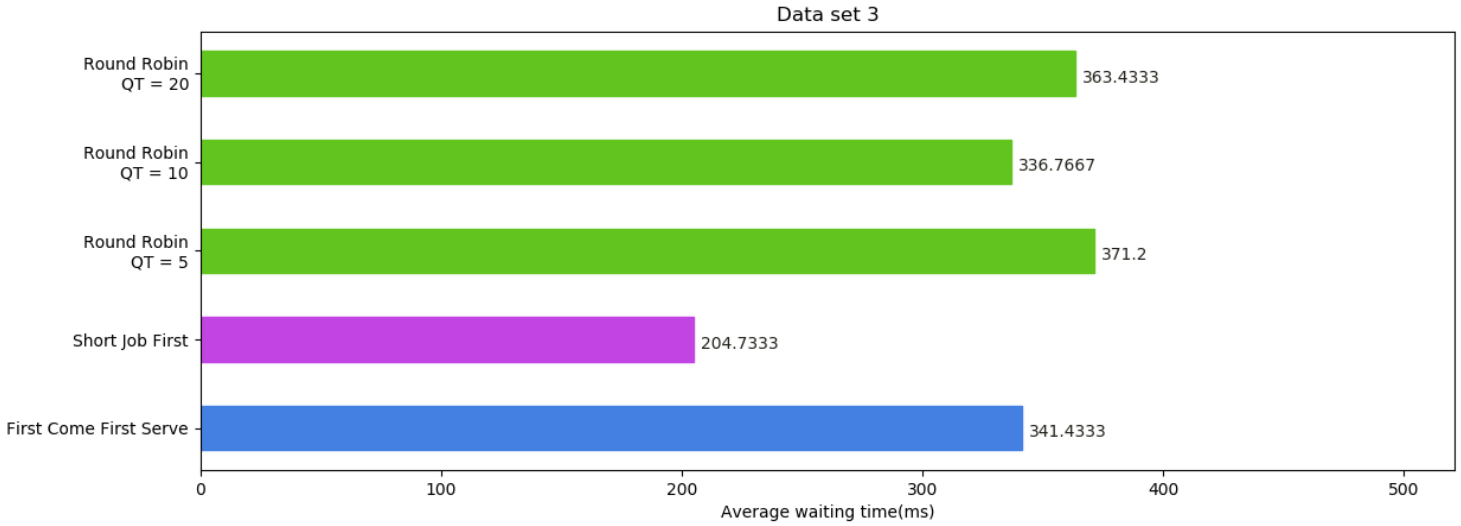


Histogram แสดงการกระจายตัวของชุดข้อมูลที่ 3

ผลจากการทำงานจากชุดข้อมูลแต่ละชุด







สรุปผลการทดลอง

จากการทดลองทั้งสามชุดข้อมูล ทำให้พบว่าการทำงานแบบ Short Jobs First จะมี Average waiting time ที่น้อยกว่า แบบอื่นมาก ในทุกรูปแบบชุดข้อมูล และรองลงมาคือ First Come First Serve ที่มีเวลาในการทำงานน้อยกว่า Round-Robin ในส่วนมาก

สำหรับการทำงานของ Round Robin ซึ่งได้ทดสอบด้วย Quantum time สามค่าในแต่ละชุดข้อมูล ซึ่งได้ผลลัพธ์ที่ค่อนข้างน่าสนใจ คือ ในชุดข้อมูลที่ 1 Round Robin ที่ QT = 20 ทำงานได้เร็วที่สุด และเวลาจะเพิ่มขึ้นเรื่อยๆ สวนทางกับQT และเมื่อนำมาเปรียบเทียบกับการทำงานแบบเดียวกันแต่ละชุดข้อมูล พบว่า เวลาในการทำงานไม่ได้มีความสัมพันธ์ เชิงเส้นตรง กับ QT คือ เวลาในการทำงานที่ดีที่สุดของการทำงาน Round Robin ในชุดข้อมูลที่ 2 และ 3 คือเมื่อ QT = 10 และจะเพิ่มขึ้น หากค่าเพิ่มเป็น 20 หรือ ลดลงเป็น 5 แสดงให้เห็นว่าการกระจายตัวของข้อมูลแต่ละแบบจะมีค่า QT ?เหมาะสามต่างกันไป สำหรับการงานของ Round Robin algorithm

Code การทำงาน

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

testSeed = [[1,10], [15,25], [30,40]]
figc = 0

def testSetGen(ratio):
    result = []
    tmp = []
    for i in range(0,len(ratio)):
        tmp.extend(np.random.random_integers(testSeed[i][0],
testSeed[i][1],ratio[i]))
    np.random.shuffle(tmp)
    for i in range(1,len(tmp)+1):
        result.append([i,tmp[i-1]])
    return result, tmp

def toRangeList(data):
    result = []
    startT = 0
    endT = 0
    for i in range(0,len(data)):
        endT = startT + data[i][1]
        result.append([data[i][0], startT, endT, data[i][1]])
        startT = endT
    return result

def algoFCFS(data):
    return toRangeList(data)

def algoSJF(data):
    return toRangeList(sorted(data, key=lambda x: x[1]))
```

```
def algoRR(data,qtime=10):
    pending = []
    result = []
    for i in data:
        if i[1] <= qtime:
            result.append(i)
        else:
            pending.append([i[0], i[1]-qtime])
            result.append([i[0], qtime])
    while len(pending) > 0:
        x = pending.pop(0)
        if x[1] <= qtime:
            result.append(x)
        else:
            pending.append([x[0], x[1]-qtime])
            result.append([x[0], qtime])
    return toRangeList(result)

def eachDelayTime(data):
    n = 0
    mask = np.zeros((len(data),), dtype=int)
    for i in range(0,len(data)):
        if mask[data[i][0]-1] == 0:
            n += 1
            mask[data[i][0]-1] = 1
    index = []
    for i in range(0,n):
        index.append([])
    for i in range(0,len(data)):
        tmp = [data[i][1], data[i][2]]
        index[data[i][0]-1].append(tmp)
    return index

def delayTime(data):
    x = []
    for i in data:
        tmp = i[0][0]
        for j in range(1,len(i)):
            tmp += i[j][0] - i[j-1][1]
        x.append(tmp)
    return float(sum(x)) / len(data)
```

```
def dataPlot(data):
    global figc

    for i in range(0,len(data)):
        fig = plt.figure(figc)
        ax = fig.add_subplot(111)
        ax.set_title('Test set %d'% (i+1))
        sns.kdeplot(data[i][0],shade=True)
        figc += 1

    for i in range(0,len(data)):
        color_set = ['#4480e2','#c244e2','#61c41f','#61c41f','#61c41f']
        x = [u'First Come First Serve', u'Short Job First', u'Round Robin\nQT =
5', u'Round Robin\nQT = 10', u'Round Robin\nQT = 20']
        y = data[i][1:3]
        y.extend(data[i][3])
        print(y)
        fig, ax = plt.subplots()
        width = 0.50 # the width of the bars
        ind = np.arange(len(y))  # the x locations for the groups
        barlist = ax.barh(ind, y, width, color="blue")
        ax.set_yticks((ind+width/2)-(.25))
        ax.set_yticklabels(x, minor=False)
        ax.set_xlim(0,max(y)+150)
        for j in range(0,len(barlist)):
            barlist[j].set_color(color_set[j])
        plt.title('Data set %d'% (i+1))
        plt.xlabel('Average waiting time(ms)')
        plt.ylabel('y')
        for i, v in enumerate(y):
            ax.text(v + 3, i - 0.1, str(round(v,4)), color='#211f18',
fontweight='light')
    return
```

```
if __name__ == "__main__":
    testSetRatio = []
    testSetRatio.append([15,30,15]) #test set 1
    testSetRatio.append([30, 0,30]) #test set 2
    testSetRatio.append([40,15,5])  #test set 3
    RRqt = []
    RRqt.append(5)
    RRqt.append(10)
    RRqt.append(20)
    result = []
    o = 0
    for i in testSetRatio:
        test, dataset = testSetGen(i)
        print('[Dataset]')
        s = '[' + ', '.join(str(e[1]) for e in test) + ']'
        print(s)

        print('\nFirst Come First Serve Algorithm')
        x = delayTime(eachDelayTime(algoFCFS(test)))
        print('Average waiting time = %.4f'% x)
        print("++++++++++++++++++++++++++")
        print('\nShort Job First Algorithm')
        y = delayTime(eachDelayTime(algoSJF(test)))
        print('Average waiting time = %.4f'% y)
        print("--------------------------")
        print('\nRound Robin Algorithm')
        z = []
        for j in range(0,len(RRqt)):
            z.append(delayTime(eachDelayTime(algoRR(test,RRqt[j]))))
            print('Average waiting time (QT = %2d) = %.4f'%  (RRqt[j], z[j]))
            print("^^^^^^^^^^^^^^^^^^^^^^^^^^")
        result.append([dataset,x,y,z])
        o += 1

    dataPlot(result)
    plt.show()
```