Coding Computer Scheduling Algorithms

How to code FCFS, SJF, NPP, and RR

CSCI 375 Extra Credit Assignment by Tatianna Robinson

How processes are organized

- User is first prompted to enter the total amount of processes
- Is then asked to enter arrival time, burst and priority n times.
- Information is stored in an object of type “`Process`”
- Every entered `Process` object is appended to an array of `processes[]`

```
def getProcesses():
    num = input("How many processes are there? ")

    i = 0
    processes = []
    while i < num:
        name = "p" + str(i+1)
        burst = eval('input("burst time: ")')
        arrival = eval('input("arrival time: ")')
        priority = eval('input("priority: ")')
        processes.append(Process(name, arrival, burst, priority))
        i += 1

    processes = sorted(processes, key=lambda Process: Process.arrival)

    #for i in range(len(processes)):
    #   processes[i].display()

    return processes
```

Process Class

```
class Process:
    def __init__ (self, name, arrival, burst, priority):
        self.name = name
        self.arrival = arrival
        self.burst = burst
        self.priority = priority

    def display(self):
        print (self.name)
        print (self.arrival)
        print (self.burst)
        print (self.priority)
```

FCFS:

- Get minimum arrival time
- Append process name to `gantt` n times with n = burst
- Delete process from array
- Repeat with remaining processes

```
def fcfs(array):
    arr = deepcopy(array)
    gantt = []
    deleted = False

    while len(arr) > 0:
        min = getMinArrival(arr)
        i = 0
        while i < len(arr):
            while len(gantt) < arr[i].arrival:
                gantt.append("00")
            if min == arr[i].arrival:
                #add to gant chart (burst) amount of times
                while arr[i].burst > 0:
                    gantt.append(arr[i].name)
                    arr[i].burst -= 1
                del(arr[i])
                deleted = True
            if not deleted:
                i += 1
            deleted = False
    return (gantt)
```

SJF:

- Get minimum burst time
- Append process name to `gantt` n times with n = burst
- Delete process from array
- Repeat

```
def sjf(array):
    arr = deepcopy(array)
    gantt = []
    deleted = False

    while len(arr) > 0:
        min = getMinBurst(arr)
        i = 0
        while i < len(arr):
            while len(gantt) < arr[i].arrival:
                gantt.append("00")
            if min == arr[i].burst:
                #add to gant chart (burst) amount of times
                while arr[i].burst > 0:
                    gantt.append(arr[i].name)
                    arr[i].burst -= 1
                del(arr[i])
                deleted = True
            if not deleted:
                i += 1
            deleted = False
    #print("SJF:")
    #display(gantt)
    return gantt
```

Non Preemptive Priority:

- Get minimum priority (lower priority value is a higher priority)
- Append process name to `gantt` n times with n = burst
- Delete process from array
- Repeat

```
def npp(array):
    arr = deepcopy(array)
    gantt = []
    deleted = False

    while len(arr) > 0:
        min = getHighestPriority(arr)
        i = 0
        while i < len(arr):
            while len(gantt) < arr[i].arrival:
                gantt.append("00")
            if min == arr[i].priority:
                #add to gantt chart (burst) amount of times
                while arr[i].burst > 0:
                    gantt.append(arr[i].name)
                    arr[i].burst -= 1
                del(arr[i])
                deleted = True
            if not deleted:
                i += 1
            deleted = False
    #print("NPP:")
    #display(gantt)
    return gantt
```

Round Robin

- Get quantum
- Sort by arrival time (RR defaults to FCFS)
- While there are remaining `processes`..
  - Iterator `i` walks from beginning to end of `processes` array
  - Append process name to `gantt` array `q` amount of times
  - Decrease the burst by `q`
  - If process has no remaining bursts, delete it from `processes` array
  - If a process is deleted before quantum is reached, DON'T increment `i`!!!!
  - After you get to process n, return to the front of the processes array by setting `i = 0`
  - Repeat until all processes are deleted

```
def rr(array, q):
    arr = deepcopy(array)
    gantt = []
    deleted = False
    quantum = q

    while len(arr) > 0:
        i = 0
        while i < len(arr):
            deleted = False
            while len(gantt) < arr[i].arrival:
                gantt.append("00")

            if arr[i].burst > 0 and quantum > 0:
                gantt.append(arr[i].name)
                arr[i].burst -= 1
                quantum -= 1

            if arr[i].burst == 0:
                del(arr[i])
                deleted = True

            if quantum == 0:
                quantum = q
                if not deleted:
                    i += 1

    #print("Round Robin:")
    #display(gantt)
    return gantt
```

Sample Output (Extended)

- NO GUI :(
- Gantt chart is an array of strings
- Each index of array has a string containing the process name
- “Push process name into gantt array n amount of times”

```
FCFS:
['p1', 'p1', 'p1', 'p1', 'p1', 'p1', 'p1', 'p1', 'p1', 'p1', 'p2', 'p3', 'p3', 'p4', 'p5', 'p5', 'p5', 'p5', 'p5']
```

Compress(gantt)

- Create a new empty `arr`, which will be the new gantt chart
- Nested loop checks for duplicate entries at `gantt[i]` and `gantt[j]`
- Append the name of process that repeated from `gantt[i]` to `gantt[j]` to `arr` as "`process name: i-j`"

```
def compress(gantt):

    arr = []
    start = 0
    i = 0

    while i < len(gantt):
        start = i # for last append i will be out of range
        j = i+1

        while j < len(gantt) and gantt[i] == gantt[j]:
            j += 1
        if j < len(gantt) and gantt[i] != gantt[j]:
            arr.append( gantt[i] + ":"+ str(i) + "-" + str(j))
        i = j

    #LAST PROCESS is never appended (since process is appended
    #to the gantt chart whenever a difference is found between
    #gantt[i] and gantt[j]

    arr.append(gantt[start] + ":"+ str(start) + "-" + str(j))
    return arr
```

Sample Output (Compressed)

```
FCFS:
['p1', 'p1', 'p1', 'p1', 'p1', 'p1', 'p1', 'p1', 'p1', 'p1', 'p2', 'p3', 'p3', 'p4', 'p5', 'p5', 'p5', 'p5', 'p5']


['p1:0-10', 'p2:10-11', 'p3:11-13', 'p4:13-14', 'p5:14-19']
```

```
Tatiannas-MacBook-Pro:desktop tatianna$ python scheduling.py
How many processes are there? 5
arrival time: 0
burst time: 10
priority: 3
arrival time: 0
burst time: 1
priority: 1
arrival time: 0
burst time: 2
priority: 3
arrival time: 0
burst time: 1
priority: 4
arrival time: 0
burst time: 5
priority: 2

FCFS:
['p1:0-10', 'p2:10-11', 'p3:11-13', 'p4:13-14', 'p5:14-19']

SJF:
['p2:0-1', 'p4:1-2', 'p3:2-4', 'p5:4-9', 'p1:9-19']

NPP:
['p2:0-1', 'p5:1-6', 'p1:6-16', 'p3:16-18', 'p4:18-19']

Round Robin: What is the quantum? 1
['p1:0-1', 'p2:1-2', 'p3:2-3', 'p4:3-4', 'p5:4-5', 'p1:5-6', 'p3:6-7', 'p5:7-8', 'p1:8-9', 'p5:9-10', 'p1:10-11', 'p5:11-12', 'p1:12-13', 'p5:13-14', 'p1:14-19']
```

To be possibly added:

- Exception handling
- GUI
- Turnaround times and wait times for each algorithm
- Support NPP on both low and high priority values
- Round Robin modifications