CS110 Assignment 2 - Fall 2021, Minerva University

A day in the life of a Minervan—Part I

This is an LBA assignment. Current city is Seoul, South Korea. Notice: All of the activities mentioned in this assignment followed local COVID-19 Prevention Guidelines such as social distancing, and wearing masks.

NAME = "Steven H. Yang"

COLLABORATORS = "Jessey Uche-Nwichi"

Q1

A)

Prepare a table containing all the activities that you plan to do in the city of your rotation, with a short, compelling justification of why they are interesting.

UPDATE THE TABLE WITH NEW TABLE

ID	Description	Minutes	Dependencies	Justification	Priority	TConstraint
0	Start working at WeWork	10	[8,9,10,11]	WeWork is my favorite place to work	50	X
1	Meeting Civic Partner	60	[12,13,14,15]	My team has a weekly meeting with civic partner	67	840
2	CS110 Assignment First Draft	120	[16,17,18,19,20]	This is literally what I am doing now	100	X
3	Have a meal	60	[21,22,23,24,25]	Very essential to survive at Minerva	78	X
4	Put clean clothes in closet	30	[26,27,28,29,30,31,32]	I want to stay clean	55	X
5	Go to a Korean Cafe near the res	10	[33,34,35]	There are so many cafes near the res to work	50	X
6	Watch one episode of K Drama on Netflix	60	[36,37,38]	Korea is popluar for Netflix series	30	X
7	Have a street food of Korea	15	[39,40,41]	I found so many of them look so good	46	1140
8	Reload my transportation card balance	5	[]	Everywhere I go, I need a transportation card	75	X
9	Find which WeWork branch that I want to go on that day	5	[]	There are multiple WeWork branches, I have to choose one	65	X
10	Book a desk on WeWork app	5	[9]	I cannot enter WeWork without booking a desk	65	X
11	Rid bus to WeWork	20	[8,9,10]	There are no WeWorks around the res, I have to take a bus	60	X
12	Review the meeting agenda	10	[]	This is critical as I am the team lead	80	805
13	Find appropriate outfit for the meeting	10	[]	Working in Korea is formal	30	815
14	Set up necesscities like laptop, and iPad	10	[]	I cannot miss these for the meeting!	90	825
15	Remind Group members that we have meeting with meeting agenda	5	[12]	As a team leader, remind members about the meeting	30	835
16	Plug in my laptop to a charger	5	[]	Jupyter notebook eats my battery, so this is important	46	X
17	Open Forum	5	[16]	This is where I can get all the class related materials	70	X
18	Read Assignment Instruction	10	[16,17]	Assignment cannot be done without reading the instruction	90	X
19	Open Jupyter notebook	5	[16]	This is where CS110ers live	70	X
20	Solve Q1 to Q3	60	[16,17,18,19]	It was my goal for today	96	X
21	Confirm which groceries I have	5	[]	I didn't want to buy more groceries, so see what I have now	10	X
22	Prepare utensils	5	[]	I don't think it's hygenic to eat/cook with my hands	5	X

ID	Description	Minutes	Dependencies	Justification	Priority	TConstraint
23	Find recipe	5	[21]	So what am I going to have today	5	X
24	Cook	15	[21,22,23]	No raw food!	11	X
25	Store in containers	5	[21,22,23,24]	This will save my time later	46	X
26	Pack the laundry in the laundry bag	5	[]	This helps me to move my laundry easier	3	X
27	Go to the nearest self-washing machine store	5	[26]	This is actually cheaper than the ones in the res	15	X
28	Put the laundry into the machine	5	[26,27]	This is the first step you get closer to a clean life	13	X
29	Run the machine	20	[26,27,28]	Machine never run unless humans start it	18	X
30	Take out laundry	5	[26,27,28,29]	I don't want to leave my clothes there for forever!	40	X
31	Bring laundry back to the res	5	[26,27,28,29,30]	Let's go back home to put them on the drying rack	80	X
32	Dry them on the rack	60	[26,27,28,29,30,31]	It also saves money from not using a dryer	85	X
33	Google the top instagrammable cafe near the res	5	[]	Work views from these cafes are amazing	30	X
34	Leave the res hall	5	[33]	Time to go out...	5	X
35	Walk to the cafe	5	[33,34]	There are so many cafes in walkable distance	3	X
36	Login to Netflix	5	[]	Netflix cannot be watched unless I login on my profile	3	X
37	Find Trending in Korea	5	[36]	Netflix tracks my IP address and give recommendations in Korea	4	X
38	Start Episode 1 for the Top 1 Trending in Korea	5	[36,37]	Let's see what's a trend in Korea	10	X
39	Find a random street vendor near Sookmyung University	5	[]	There are so many street vendors in this area because of university students	20	1115
40	Order Tteokbokki	5	[39]	My favorite Korean street food!	37	1120
41	Enjoy	15	[40]	This is exactly why we have food	40	1125

B)

How will you store information about these activities and sub-tasks?

From ID 0 to 7, they are the big tasks that I wanted to do in a day and each has sub-tasks. This means that each big task cannot be completed unless all of the sub-tasks are done. These sub-tasks will be represented with dependencies. Not only this, each task has different priority value, and duration of the task. There are even some tasks that have time constratint which is the task should be done at a certain time no matter what.

Understand that each task has all attributes mentioned in the table and some of them are necessary to create a task scheduler, I use class Task to store all of the tasks above with appropriate attributes. Commonly, each task will be stored into the class with the same order of attributes. Attributes are following: task_id, description,duration, dependencies, priority, tconstraint, status="N". Even status is not mentioned in the table, I assume none of the task has been done yet; therefore I will have status = 'N' at the end of the each class.

C)

Describe how your scheduler will work, with an emphasis on why a priority queue is a well-suited data structure to handle the prioritization of tasks, and how you have defined and computed the priority value of each task and/or sub-task.

As we discussed in the class, heap sort is a great algorithm to apply to prioritizing tasks. It is because that each task will have a numerical value that represents the priority of each task. I use MaxHeap in this assignment; therefore, a task with a greater number of priority means should be done before other tasks with lower priority values. Using MaxHeapify is helpful here to construct a heap data structure from the raw input data.

Before running the task scheduler, I assume that I have not done any of the tasks; therefore, I append status 'N,' which stands for not started. The scheduler will run the tasks in the priority queue first. However, at the very beginning of the algorithm, the priority queue is empty. This is where heappush comes. Heappush will put the tasks without dependencies into the priority queue as the ones with dependencies cannot be run unless all of their dependent tasks are finished. Heappush is used to insert the element mentioned in its arguments into the heap. While heappush inserts an element, it adjusts the order; therefore, the heap structure is still maintained.

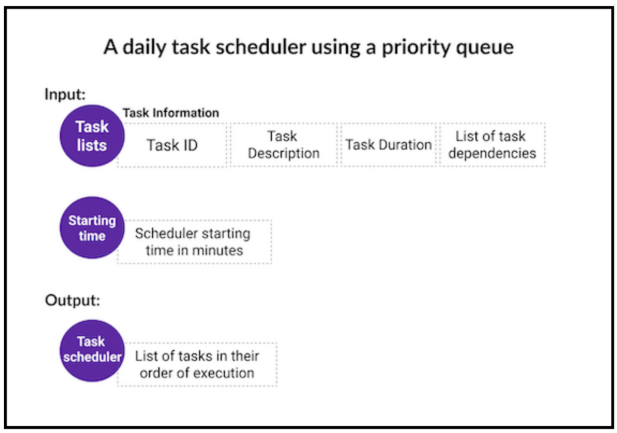
When the priority queue is prepared and in max-heap, the task scheduler will first run the tasks with the highest priority. If the task that the scheduler just ran is one of the dependencies of other tasks, it will remove the task just ran's the ID from other tasks' dependencies. Also, the current time will be updated by the duration of the task. Once the task is done, it will be changed to the status 'C,' which is completed.

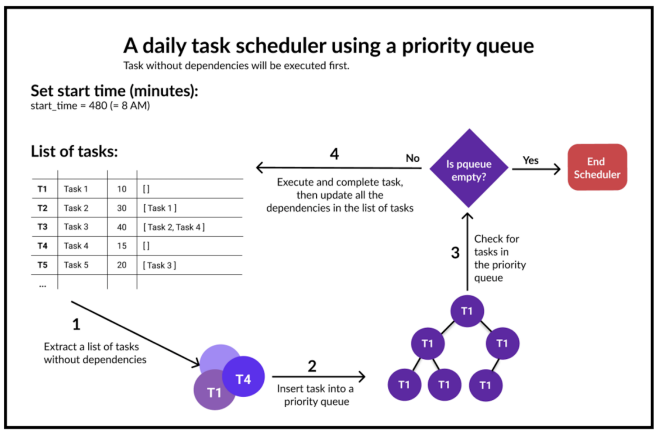
However, there are some tasks that should be done at a certain time that one cannot push back, such as meeting with a civic partner. These tasks have a variable called TConstraint which states that this task should be running at this time no matter what. I implement this reality in the code as follows:

1. Before running each task, the scheduler calculates the remaining time until the fixed time of TConstraint Task.
2. If a task that the scheduler was about to run's duration, which is the one that is about to ready from the priority queue's duration, is smaller than TConstraint - Current_Time, it runs.
3. Otherwise, I will take a break during that time.

I defined the priority values of each task based on how much I feel that's a loss for me if I don't complete it. For example, missing a CS110 class will result in doing make-up work that I most don't like to do has 100 priority value.

Session 5.2 Task Scheduler Diagram was useful to built this project.





Q2

A)

Program an activity scheduler in Python, which receives the list of tasks above as input and returns a schedule for you to follow.

In [73]:

```
class MaxHeapq:
    """
    A class that implements properties and methods
                that support a max priority queue data structure

                Attributes
          ----------
          heap : arr
              A Python list where key values in the max heap are stored
          heap_size: int
              An integer counter of the number of keys present in the max heap
          """

    def __init__(self):
        """
        Parameters
        ----------
        None
        """
        self.heap       = []
        self.heap_size  = 0

    def left(self, i):
        """
        Takes the index of the parent node
        and returns the index of the left child node

        Parameters
        ----------
        i: int
          Index of parent node

        Returns
        ----------
        int
          Index of the left child node

        """
        return 2 * i + 1

    def right(self, i):
        """
        Takes the index of the parent node
        and returns the index of the right child node

        Parameters
        ----------
        i: int
            Index of parent node

        Returns
        ----------
        int
            Index of the right child node

        """

        return 2 * i + 2
```

```
    def parent(self, i):
        """
        Takes the index of the child node
        and returns the index of the parent node

        Parameters
        ----------
        i: int
            Index of child node

        Returns
        ----------
        int
            Index of the parent node

        """

        return (i - 1)//2

    def maxk(self):
        """
        Returns the highest key in the priority queue.

        Parameters
        ----------
        None

        Returns
        ----------
        int
            the highest key in the priority queue

        """
        return self.heap[0]


    def heappush(self, key):
        """
        Insert a key into a priority queue

        Parameters
        ----------
        key: int
            The key value to be inserted

        Returns
        ----------
        None
        """
        self.heap.append(-float("inf"))
        self.increase_key(self.heap_size,key)
        self.heap_size+=1

    def increase_key(self, i, key):
        """
        Modifies the value of a key in a max priority queue
        with a higher value

        Parameters
        ----------
        i: int
```

```
            The index of the key to be modified
        key: int
            The new key value

        Returns
        ----------
        None
        """
        if key.priority < self.heap[i]:
            raise ValueError('new key is smaller than the current key')
        self.heap[i] = key
        while i > 0 and self.heap[self.parent(i)].priority < self.heap[i].priori
ty:
            j = self.parent(i)
            holder = self.heap[j]
            self.heap[j] = self.heap[i]
            self.heap[i] = holder
            i = j


    def heapify(self, i):
        """
        Creates a max heap from the index given

        Parameters
        ----------
        i: int
            The index of of the root node of the subtree to be heapify

        Returns
        ----------
        None
        """
        l = self.left(i)
        r = self.right(i)
        heap = self.heap
        if l <= (self.heap_size-1) and heap[l].priority>heap[i].priority:
            largest = l
        else:
            largest = i
        if r <= (self.heap_size-1) and heap[r].priority > heap[largest].priority
:
            largest = r
        if largest != i:
            heap[i], heap[largest] = heap[largest], heap[i]
            self.heapify(largest)

    def heappop(self):
        """
        returns the larest key in the max priority queue
        and remove it from the max priority queue

        Parameters
        ----------
        None

        Returns
        ----------
        int
            the max value in the heap that is extracted
        """
```

```
        if self.heap_size < 1:
            raise ValueError('Heap underflow: There are no keys in the priority
queue ')
        maxk = self.heap[0]
        self.heap[0] = self.heap[-1]
        self.heap.pop()
        self.heap_size-=1
        self.heapify(0)
        return maxk
```

In [74]:

```
class Task:
    """
    - id: Task Id
    - description: Short description of the task
    - duration: Duration in minutes
    - priority: Priority level of a task (ranging from 0 to 100)
    - status: Current status of the task:

    """
    #Initializes an instance of Task
    def __init__(self,task_id,description,duration,dependencies,priority,tconstr
aint,status="N"):
        self.id = task_id
        self.description = description
        self.duration = duration
        self.dependencies = dependencies
        self.priority = priority
        self.status = status
        self.tconstraint = tconstraint
    def __repr__(self):
        return f"{self.description} - id: {self.id}\n \tDuration:{self.duration}
\n\tDepends on: {self.dependencies}\n\tStatus: {self.status} \n\tPriority: {self
.priority} \n\tconstraint: {self.tconstraint}"

    def __lt__(self, other):
        return self.priority < other.priority

class TaskScheduler:
    """
    A Simple Daily Task Scheduler Using Priority Queues
    """
    NOT_STARTED ='N'
    IN_PRIORITY_QUEUE = 'I'
    COMPLETED = 'C'

    def __init__(self, tasks):
        self.tasks = tasks
        self.priority_queue = MaxHeapq()
        self

    def print_self(self):
        print('Input List of Tasks')
        for t in self.tasks:
            print(t)

    def remove_dependency(self, task_id):
        """
        Input: list of tasks and task_id of the task just completed
        Output: lists of tasks with t_id removed
        """
        for t in self.tasks:
            if t.id != task_id and task_id in t.dependencies:
                t.dependencies.remove(task_id)

    def get_tasks_ready(self):
        """
        Implements step 1 of the scheduler
        Input: list of tasks
        Output: list of tasks that are ready to execute (i.e. tasks with no pend
```

```
ending task dependencies)
        """
        for task in self.tasks:
            if task.status == self.NOT_STARTED and not task.dependencies and not
task.tconstraint: # If task has no dependencies and is not yet in queue
                #Since we are not including constrainted tasks into the priority
queue.
                task.status = self.IN_PRIORITY_QUEUE # Change status of the task
                ######## Push task into the priority queue (1line of code requir
ed)
                ####### your code here
                self.priority_queue.heappush(task)

    def check_unscheduled_tasks(self):
        """
        Input: list of tasks
        Output: boolean (checks the status of all tasks and returns True if at l
east one task has status = 'N'
        """
        for task in tasks:
            if task.status == self.NOT_STARTED:
                return True
        return False

    def format_time(self, time):
        """
        Input: time in minutes
        Output: Converts the time in hours and minutes
        """
        return f"{time//60}h{time%60:02d}"

    def run_task_scheduler(self, starting_time = 480):
        """
        This function actually runs the task scheduler with the given starting_t
ime of the day.
        Input: starting time which is 8 AM
        Output: Task Schedule
        """
        current_time = starting_time
        with_constraint = [] #open a list that can store tasks with time constra
int.
        for task in self.tasks:
            if task.tconstraint is not False:
                with_constraint.append(task.tconstraint)
                #append each task with time constraints as it does not need Heap
to implement the schedule.
        with_constraint = sorted(with_constraint)
        constraint_index = 0
        while self.check_unscheduled_tasks() or self.priority_queue.heap_size !=
0:
            #STEPs 1 and 2: Extract tasks ready to execute (those without depend
encies) and push them into the priority queue
            self.get_tasks_ready()
            task = self.priority_queue.maxk() #Use Maxk here because it is still
unsure that current task can be popped or not.
            if len(self.priority_queue.heap) > 0 :  #STEP 3: Check for tasks in
 the priority queue.
                # STEP 4: get the tasks on top of the priority queue (1 line of
 code required)
                ### your code here
                if constraint_index < len(with_constraint) and task.duration > (
```

```
with_constraint[constraint_index] - current_time):
                    #if the current task's duration from the priority queue is g
reater than constrained time - current time, there's no way to do this task.
                    for index in range(len(self.tasks)):
                        if self.tasks[index].tconstraint == with_constraint[cons
traint_index]:
                            self.tasks[index].priority = 101 #give the maximum p
riority
                            task = self.tasks[index]
                            ## Push task into the priority queue
                            self.priority_queue.heappush(task)
                            self.priority_queue.heapify(index)
                            task = self.priority_queue.heappop()
                            if task.tconstraint - current_time > 0:
                                print("Take a break for", task.tconstraint - cur
rent_time,'minutes 😌 ')
                            current_time = task.tconstraint
                            constraint_index += 1

                            break
                else:
                    task = self.priority_queue.heappop()
                #task = self.priority_queue.heappop()
                print(f"⏰ Simple Scheduler at time {self.format_time(current_tim
e)} started executing task {task.id} that takes {task.duration} mins")
                current_time += task.duration
                print(f"✅ Completed Task {task.id} - '{task.description}' at ti
me {self.format_time(current_time)}\n")
                ####### if the task is completed, it cannot be a dependency on o
ther tasks, so remove it from the dependency list (1 line of code required)
                self.remove_dependency(task.id)
                task.status = self.COMPLETED
        total_time = current_time - starting_time
        print(f"🏁 Completed all planned tasks in {total_time//60}h{total_time%6
0:02d}min")
```

In [75]:

```
#This code reads .csv of the table and cretes lines of code that I can use to ma
nually input the tasks
import pandas as pd
df = pd.read_csv("Q1A Table - Sheet1.csv")
for x in range(0,40):
    if df.TConstraint[x] == "X":
        df.TConstraint[x] = False
    print("Task(",df.ID[x],", \'",df.Description[x],"\',",df.Minutes[x],",",df.D
ependencies[x],",",df.Priority[x],",",df.TConstraint[x],"),")
    #have last line seperately to not to include the comma end.
print("Task(",df.ID[40],", \'",df.Description[40],"\',",df.Minutes[40],",",df.De
pendencies[40],",",df.Priority[40],",",df.TConstraint[40],")")
```

```
Task( 0 , ' Start working at WeWork ', 10 , [8,9,10,11] , 50 , False
),
Task( 1 , ' Meeting Civic Partner ', 60 , [12,13,14,15] , 67 , 840
),
Task( 2 , ' CS110 Assignment First Draft ', 120 , [16,17,18,19,20] ,
100 , False ),
Task( 3 , ' Have a meal ', 60 , [21,22,23,24,25] , 78 , False ),
Task( 4 , ' Put clean clothes in closet ', 30 , [26,27,28,29,30,31,3
2] , 55 , False ),
Task( 5 , ' Go to a Korean Cafe near the res ', 10 , [33,34,35] , 50
, False ),
Task( 6 , ' Watch one episode of K Drama on Netflix ', 60 , [36,37,3
8] , 30 , False ),
Task( 7 , ' Have a street food of Korea ', 15 , [39,40] , 46 , 1125
),
Task( 8 , ' Reload my transportation card balance ', 5 , [] , 75 , F
alse ),
Task( 9 , ' Find which WeWork branch that I want to go on that day
', 5 , [] , 65 , False ),
Task( 10 , ' Book a desk on WeWork app ', 5 , [9] , 65 , False ),
Task( 11 , ' Rid bus to WeWork ', 20 , [8,9,10] , 60 , False ),
Task( 12 , ' Review the meeting agenda ', 10 , [] , 80 , 805 ),
Task( 13 , ' Find appropriate outfit for the meeting ', 10 , [] , 30
, 815 ),
Task( 14 , ' Set up necesscities like laptop, and iPad ', 10 , [] ,
90 , 825 ),
Task( 15 , ' Remind Group members that we have meeting with meeting
agenda ', 5 , [12] , 30 , 835 ),
Task( 16 , ' Plug in my laptop to a charger ', 5 , [] , 46 , False
),
Task( 17 , ' Open Forum ', 5 , [16] , 70 , False ),
Task( 18 , ' Read Assignment Instruction ', 10 , [16,17] , 90 , Fals
e ),
Task( 19 , ' Open Jupyter notebook ', 5 , [16] , 70 , False ),
Task( 20 , ' Solve Q1 to Q3 ', 60 , [16,17,18,19] , 96 , False ),
Task( 21 , ' Confirm which groceries I have ', 5 , [] , 10 , False
),
Task( 22 , ' Prepare utensils ', 5 , [] , 5 , False ),
Task( 23 , ' Find recipe ', 5 , [21] , 5 , False ),
Task( 24 , ' Cook ', 15 , [21,22,23] , 11 , False ),
Task( 25 , ' Store in containers ', 5 , [21,22,23,24] , 46 , False
),
Task( 26 , ' Pack the laundry in the laundry bag ', 5 , [] , 3 , Fal
se ),
Task( 27 , ' Go to the nearest self-washing machine store ', 5 , [2
6] , 15 , False ),
Task( 28 , ' Put the laundry into the machine ', 5 , [26,27] , 13 ,
False ),
Task( 29 , ' Run the machine ', 20 , [26,27,28] , 18 , False ),
Task( 30 , ' Take out laundry ', 5 , [26,27,28,29] , 40 , False ),
Task( 31 , ' Bring laundry back to the res ', 5 , [26,27,28,29,30] ,
80 , False ),
Task( 32 , ' Dry them on the rack ', 60 , [26,27,28,29,30,31] , 85 ,
False ),
Task( 33 , ' Google the top instagrammable cafe near the res ', 5 ,
[] , 30 , False ),
Task( 34 , ' Leave the res hall ', 5 , [33] , 5 , False ),
Task( 35 , ' Walk to the cafe ', 5 , [33,34] , 3 , False ),
Task( 36 , ' Login to Netflix ', 5 , [] , 3 , False ),
Task( 37 , ' Find Trending in Korea ', 5 , [36] , 4 , False ),
Task( 38 , ' Start Episode 1 for the Top 1 Trending in Korea ', 5 ,
```

```
[36,37] , 10 , False ),
Task( 39 , ' Find a random street vendor near Sookmyung University
', 5 , [] , 20 , 1115 ),
Task( 40 , ' Order Tteokbokki ', 5 , [39] , 37 , 1120 )

<ipython-input-75-fa94ebd27e53>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py
  df.TConstraint[x] = False
```

B)

In addition to the actual scheduler, provide at least one simple example to demonstrate how your scheduler prioritizes tasks based on their priority value.

In [76]:

```
tasks = [
Task( 0 , ' Start working at WeWork ', 10 , [8,9,10,11] , 50 , False ),
Task( 1 , ' Meeting Civic Partner ', 60 , [12,13,14,15] , 67 , 840 ),
Task( 2 , ' CS110 Assignment First Draft ', 120 , [16,17,18,19,20] , 100 , False
),
Task( 3 , ' Have a meal ', 60 , [21,22,23,24,25] , 78 , False ),
Task( 4 , ' Put clean clothes in closet ', 30 , [26,27,28,29,30,31,32] , 55 , Fa
lse ),
Task( 5 , ' Go to a Korean Cafe near the res ', 10 , [33,34,35] , 50 , False ),
Task( 6 , ' Watch one episode of K Drama on Netflix ', 60 , [36,37,38] , 30 , Fa
lse ),
Task( 7 , ' Have a street food of Korea ', 15 , [39,40] , 46 , 1125 ),
Task( 8 , ' Reload my transportation card balance ', 5 , [] , 75 , False ),
Task( 9 , ' Find which WeWork branch that I want to go on that day ', 5 , [] , 6
5 , False ),
Task( 10 , ' Book a desk on WeWork app ', 5 , [9] , 65 , False ),
Task( 11 , ' Rid bus to WeWork ', 20 , [8,9,10] , 60 , False ),
Task( 12 , ' Review the meeting agenda ', 10 , [] , 80 , 805 ),
Task( 13 , ' Find appropriate outfit for the meeting ', 10 , [] , 30 , 815 ),
Task( 14 , ' Set up necesscities like laptop, and iPad ', 10 , [] , 90 , 825 ),
Task( 15 , ' Remind Group members that we have meeting with meeting agenda ', 5
, [12] , 30 , 835 ),
Task( 16 , ' Plug in my laptop to a charger ', 5 , [] , 46 , False ),
Task( 17 , ' Open Forum ', 5 , [16] , 70 , False ),
Task( 18 , ' Read Assignment Instruction ', 10 , [16,17] , 90 , False ),
Task( 19 , ' Open Jupyter notebook ', 5 , [16] , 70 , False ),
Task( 20 , ' Solve Q1 to Q3 ', 60 , [16,17,18,19] , 96 , False ),
Task( 21 , ' Confirm which groceries I have ', 5 , [] , 10 , False ),
Task( 22 , ' Prepare utensils ', 5 , [] , 5 , False ),
Task( 23 , ' Find recipe ', 5 , [21] , 5 , False ),
Task( 24 , ' Cook ', 15 , [21,22,23] , 11 , False ),
Task( 25 , ' Store in containers ', 5 , [21,22,23,24] , 46 , False ),
Task( 26 , ' Pack the laundry in the laundry bag ', 5 , [] , 3 , False ),
Task( 27 , ' Go to the nearest self-washing machine store ', 5 , [26] , 15 , Fal
se ),
Task( 28 , ' Put the laundry into the machine ', 5 , [26,27] , 13 , False ),
Task( 29 , ' Run the machine ', 20 , [26,27,28] , 18 , False ),
Task( 30 , ' Take out laundry ', 5 , [26,27,28,29] , 40 , False ),
Task( 31 , ' Bring laundry back to the res ', 5 , [26,27,28,29,30] , 80 , False
),
Task( 32 , ' Dry them on the rack ', 60 , [26,27,28,29,30,31] , 85 , False ),
Task( 33 , ' Google the top instagrammable cafe near the res ', 5 , [] , 30 , Fa
lse ),
Task( 34 , ' Leave the res hall ', 5 , [33] , 5 , False ),
Task( 35 , ' Walk to the cafe ', 5 , [33,34] , 3 , False ),
Task( 36 , ' Login to Netflix ', 5 , [] , 3 , False ),
Task( 37 , ' Find Trending in Korea ', 5 , [36] , 4 , False ),
Task( 38 , ' Start Episode 1 for the Top 1 Trending in Korea ', 5 , [36,37] , 10
, False ),
Task( 39 , ' Find a random street vendor near Sookmyung University ', 5 , [] , 2
0 , 1115 ),
Task( 40 , ' Order Tteokbokki ', 5 , [39] , 37 , 1120 )]
task_scheduler = TaskScheduler(tasks)
task_scheduler.run_task_scheduler()
```

⏰Simple Scheduler at time 8h00 started executing task 8 that takes 5 mins
✅ Completed Task 8 - ' Reload my transportation card balance ' at time 8h05

⏰Simple Scheduler at time 8h05 started executing task 9 that takes 5 mins
✅ Completed Task 9 - ' Find which WeWork branch that I want to go on that day ' at time 8h10

⏰Simple Scheduler at time 8h10 started executing task 10 that takes 5 mins
✅ Completed Task 10 - ' Book a desk on WeWork app ' at time 8h15

⏰Simple Scheduler at time 8h15 started executing task 11 that takes 20 mins
✅ Completed Task 11 - ' Rid bus to WeWork ' at time 8h35

⏰Simple Scheduler at time 8h35 started executing task 0 that takes 10 mins
✅ Completed Task 0 - ' Start working at WeWork ' at time 8h45

⏰Simple Scheduler at time 8h45 started executing task 16 that takes 5 mins
✅ Completed Task 16 - ' Plug in my laptop to a charger ' at time 8h50

⏰Simple Scheduler at time 8h50 started executing task 17 that takes 5 mins
✅ Completed Task 17 - ' Open Forum ' at time 8h55

⏰Simple Scheduler at time 8h55 started executing task 18 that takes 10 mins
✅ Completed Task 18 - ' Read Assignment Instruction ' at time 9h05

⏰Simple Scheduler at time 9h05 started executing task 19 that takes 5 mins
✅ Completed Task 19 - ' Open Jupyter notebook ' at time 9h10

⏰Simple Scheduler at time 9h10 started executing task 20 that takes 60 mins
✅ Completed Task 20 - ' Solve Q1 to Q3 ' at time 10h10

⏰Simple Scheduler at time 10h10 started executing task 2 that takes 120 mins
✅ Completed Task 2 - ' CS110 Assignment First Draft ' at time 12h10

⏰Simple Scheduler at time 12h10 started executing task 33 that takes 5 mins
✅ Completed Task 33 - ' Google the top instagrammable cafe near the res ' at time 12h15

⏰Simple Scheduler at time 12h15 started executing task 21 that takes 5 mins
✅ Completed Task 21 - ' Confirm which groceries I have ' at time 12h20

⏰Simple Scheduler at time 12h20 started executing task 34 that takes 5 mins
✅ Completed Task 34 - ' Leave the res hall ' at time 12h25

⏰

⏰Simple Scheduler at time 12h25 started executing task 23 that take
s 5 mins
✅ Completed Task 23 - ' Find recipe ' at time 12h30

⏰Simple Scheduler at time 12h30 started executing task 22 that take
s 5 mins
✅ Completed Task 22 - ' Prepare utensils ' at time 12h35

⏰Simple Scheduler at time 12h35 started executing task 24 that take
s 15 mins
✅ Completed Task 24 - ' Cook ' at time 12h50

⏰Simple Scheduler at time 12h50 started executing task 25 that take
s 5 mins
✅ Completed Task 25 - ' Store in containers ' at time 12h55

Take a break for 30 minutes 😌
⏰Simple Scheduler at time 13h25 started executing task 12 that take
s 10 mins
✅ Completed Task 12 - ' Review the meeting agenda ' at time 13h35

⏰Simple Scheduler at time 13h35 started executing task 13 that take
s 10 mins
✅ Completed Task 13 - ' Find appropriate outfit for the meeting ' a
t time 13h45

⏰Simple Scheduler at time 13h45 started executing task 14 that take
s 10 mins
✅ Completed Task 14 - ' Set up necesscities like laptop, and iPad '
at time 13h55

⏰Simple Scheduler at time 13h55 started executing task 15 that take
s 5 mins
✅ Completed Task 15 - ' Remind Group members that we have meeting w
ith meeting agenda ' at time 14h00

⏰Simple Scheduler at time 14h00 started executing task 1 that takes
60 mins
✅ Completed Task 1 - ' Meeting Civic Partner ' at time 15h00

⏰Simple Scheduler at time 15h00 started executing task 3 that takes
60 mins
✅ Completed Task 3 - ' Have a meal ' at time 16h00

⏰Simple Scheduler at time 16h00 started executing task 35 that take
s 5 mins
✅ Completed Task 35 - ' Walk to the cafe ' at time 16h05

⏰Simple Scheduler at time 16h05 started executing task 5 that takes
10 mins
✅ Completed Task 5 - ' Go to a Korean Cafe near the res ' at time 1
6h15

⏰Simple Scheduler at time 16h15 started executing task 26 that take
s 5 mins
✅ Completed Task 26 - ' Pack the laundry in the laundry bag ' at ti
me 16h20

⏰Simple Scheduler at time 16h20 started executing task 27 that take
s 5 mins
✅ Completed Task 27 - ' Go to the nearest self-washing machine stor

e ' at time 16h25

⏰Simple Scheduler at time 16h25 started executing task 28 that take
s 5 mins
✅ Completed Task 28 - ' Put the laundry into the machine ' at time
16h30

⏰Simple Scheduler at time 16h30 started executing task 29 that take
s 20 mins
✅ Completed Task 29 - ' Run the machine ' at time 16h50

⏰Simple Scheduler at time 16h50 started executing task 30 that take
s 5 mins
✅ Completed Task 30 - ' Take out laundry ' at time 16h55

⏰Simple Scheduler at time 16h55 started executing task 31 that take
s 5 mins
✅ Completed Task 31 - ' Bring laundry back to the res ' at time 17h
00

⏰Simple Scheduler at time 17h00 started executing task 32 that take
s 60 mins
✅ Completed Task 32 - ' Dry them on the rack ' at time 18h00

⏰Simple Scheduler at time 18h00 started executing task 4 that takes
30 mins
✅ Completed Task 4 - ' Put clean clothes in closet ' at time 18h30

⏰Simple Scheduler at time 18h30 started executing task 36 that take
s 5 mins
✅ Completed Task 36 - ' Login to Netflix ' at time 18h35

⏰Simple Scheduler at time 18h35 started executing task 39 that take
s 5 mins
✅ Completed Task 39 - ' Find a random street vendor near Sookmyung
University ' at time 18h40

⏰Simple Scheduler at time 18h40 started executing task 40 that take
s 5 mins
✅ Completed Task 40 - ' Order Tteokbokki ' at time 18h45

⏰Simple Scheduler at time 18h45 started executing task 7 that takes
15 mins
✅ Completed Task 7 - ' Have a street food of Korea ' at time 19h00

⏰Simple Scheduler at time 19h00 started executing task 37 that take
s 5 mins
✅ Completed Task 37 - ' Find Trending in Korea ' at time 19h05

⏰Simple Scheduler at time 19h05 started executing task 38 that take
s 5 mins
✅ Completed Task 38 - ' Start Episode 1 for the Top 1 Trending in K
orea ' at time 19h10

⏰Simple Scheduler at time 19h10 started executing task 6 that takes
60 mins
✅ Completed Task 6 - ' Watch one episode of K Drama on Netflix ' at
time 20h10

🏁 Completed all planned tasks in 12h10min

Q3

A)

Describe as clearly as you can any changes you will need to make to the first version of the scheduler to include multi-tasking activities.

First, the class Task should include an attribute that can distinguish whether the task is multitaskable and which tasks can be done simultaneously. To implement this, make multitask attributes, and for those that are not multitaskable, I note []. For the ones that can be multitaskable, I note [a,b,c...] where a,b,c are the task id of the other tasks that can be done simultaneously.

As the current code above pops the task right away, even I make a comparison with the multitaskable task, it will perform right away, and it does not make send because one task cannot be done together was actually done with another. To avoid this, I will use maxk to have the current task but not run it and compare it with the multitaskable task. If there are two multitaskable tasks, if one of them runs, it will increase the current time and be increased again by another task. Therefore, I will only update the current time by the task's duration that took longer to finish. After these two are completed, they will be removed from the queue.

B)

Describe how constraints in the scheduling process are handled by a priority queue.

In my project, all of the sub-tasks of the tasks that have time constraints are done continuously. It is because, in reality, if those sub-tasks are done separately, it'd not make sense.

In order to implement these constraints in the code, I will divide the timeline before/after all the time-constrained tasks. To visualize this, the timeline looks like this: [A]B[C]D[E], where B and C are main/sub-tasks that have time constraints. I will use MaxHeap Priority for [A],[C],[E] but collect the results at once so that we know the tasks done at [A] were removed from the queue, so it does not repeat in [C], and [E].

This code will be iteratively repeated until there are no tasks left in the queue.

Q4

In [77]:

```
#### Q4

class Task:
    """
    - id: Task Id
    - description: Short description of the task
    - duration: Duration in minutes
    - priority: Priority level of a task (ranging from 0 to 100)
    - status: Current status of the task:

    """
    #Initializes an instance of Task
    def __init__(self,task_id,description,duration,dependencies,multitask,priori
ty,tconstraint, status="N"):
        self.id = task_id
        self.description = description
        self.duration = duration
        self.dependencies = dependencies
        self.priority = priority
        self.status = status
        self.tconstraint = tconstraint
        self.multitask = multitask
    def __repr__(self):
        return f"{self.description} - id: {self.id}\n \tDuration:{self.duration}
\n\tDepends on: {self.dependencies}\n\tStatus: {self.status} \n\tPriority: {self
.priority} \n\tconstraint: {self.tconstraint}\n\multitask: {self.multitask}"

    def __lt__(self, other):
        return self.priority < other.priority

class TaskScheduler:
    """
    A Simple Daily Task Scheduler Using Priority Queues
    """
    NOT_STARTED ='N'
    IN_PRIORITY_QUEUE = 'I'
    COMPLETED = 'C'

    def __init__(self, tasks):
        self.tasks = tasks
        self.priority_queue = MaxHeapq()
        self

    def print_self(self):
        print('Input List of Tasks')
        for t in self.tasks:
            print(t)

    def remove_dependency(self, task_id):
        """
        Input: list of tasks and task_id of the task just completed
        Output: lists of tasks with t_id removed
        """
        for t in self.tasks:
            if t.id != task_id and task_id in t.dependencies:
                t.dependencies.remove(task_id)

    def get_tasks_ready(self):
        """
```

```
        Implements step 1 of the scheduler
        Input: list of tasks
        Output: list of tasks that are ready to execute (i.e. tasks with no pend
ending task dependencies)
        """
        for task in self.tasks:
            if task.status == self.NOT_STARTED and not task.dependencies and not
task.tconstraint: # If task has no dependencies and is not yet in queue
                #Since we are not including constrainted tasks into the priority
queue.
                task.status = self.IN_PRIORITY_QUEUE # Change status of the task
                ######## Push task into the priority queue (1line of code requir
ed)
                ####### your code here
                self.priority_queue.heappush(task)

    def check_unscheduled_tasks(self):
        """
        Input: list of tasks
        Output: boolean (checks the status of all tasks and returns True if at l
east one task has status = 'N'
        """
        for task in tasks:
            if task.status == self.NOT_STARTED:
                return True
        return False

    def format_time(self, time):
        """
        Input: time in minutes
        Output: Converts the time in hours and minutes
        """
        return f"{time//60}h{time%60:02d}"

    def multitasking(self, task, current_time):
        #implements multi tasking
        """
        Current_time cannot be updated more than once when we multi task. This f
unction chooses how should we increase the current_time value.
        Input: Each task, current time
        Output: Updated current_time
        """
        if task.multitask != []:
            multi_ready = task
            for i in range(len(self.tasks)):
                tsk = self.tasks[i]
                if task.multitask[0] == tsk.id: #
                    multi_ready = tsk
            if task.duration > multi_ready.duration: #TEST CASE 1
                #task.duration -= task.multi_ready.duration
                multi_ready.priority = 101 #Raise priority to the max
                #build_max_heap
                if multi_ready.status == self.IN_PRIORITY_QUEUE:
                    heap_size = len(self.tasks) #If this is in the queue alread
y, build max heap
                    for k in range(heap_size//2,-1,-1):
                        self.priority_queue.heapify(k)
                elif multi_ready.status == self.NOT_STARTED:
                    multi_ready.status = self.IN_PRIORITY_QUEUE #If this is not
 in priority queue,
                    #put this in the priority queue
```

```
                    self.priority_queue.heappush(multi_ready)

                task_run_first=self.priority_queue.heappop() #pop the first elem
ent
                print(f"⏰ Simple Scheduler at time {self.format_time(current_tim
e)} started executing task {task_run_first.id} that takes {task_run_first.durati
on} mins")
                print('👨‍💻 Multi tasking')
                current_time += task_run_first.duration #increase the current ti
me to return ticking
                print(f"✅ Completed Task {task_run_first.id} - '{task_run_first
.description}' at time {self.format_time(current_time)}\n")


                self.remove_dependency(task_run_first.id)
                task_run_first.status = self.COMPLETED


                #now run the longer task with just the remaining time
                current_time -= task_run_first.duration #return the time back fo
r the next one
                print(f"⏰ Simple Scheduler at time {self.format_time(current_tim
e)} started executing task {task.id} that takes {task.duration} mins")
                print('👨‍💻 Multi tasking')
                current_time += task.duration
                print(f"✅ Completed Task {task.id} - '{task.description}' at ti
me {self.format_time(current_time)}\n")
                # if the task is completed, it cannot be a dependency on other t
asks, so remove it from the dependency list (1 line of code required)

                self.remove_dependency(task.id)
                task.status = self.COMPLETED

            elif task.duration < multi_ready.duration:
                multi_ready.priority = 101
                #task.multi_ready.duration -= task.duration
                if multi_ready.status == self.IN_PRIORITY_QUEUE:
                    heap_size = len(self.tasks)
                    for k in range(heap_size//2,-1,-1):
                        self.priority_queue.heapify(k)
                elif multi_ready.status == self.NOT_STARTED:
                    multi_ready.status = self.IN_PRIORITY_QUEUE
                    self.priority_queue.heappush(multi_ready)

                task_run_second=self.priority_queue.heappop()

                print(f"⏰ Simple Scheduler at time {self.format_time(current_tim
e)} started executing task {task_run_second.id} that takes {task_run_second.dura
tion} mins")
                print('👨‍💻 Multi tasking')
                current_time += task_run_second.duration
                print(f"✅ Completed Task {task_run_second.id} - '{task_run_seco
nd.description}' at time {self.format_time(current_time)}\n")

                self.remove_dependency(task.id)
                task.status = self.COMPLETED

                #now run the longer task with just the remaining time
                current_time -= task.duration
                print(f"⏰ Simple Scheduler at time {self.format_time(current_tim
e)} started executing task {task.id} that takes {task.duration} mins")
```

```
                print('👷 Multi tasking')
                current_time += multi_ready.duration
                print(f"✅ Completed Task {task.id} - '{task.description}' at ti
me {self.format_time(current_time)}\n")

                self.remove_dependency(task_run_second.id)
                task_run_second.status = self.COMPLETED

            else:
                multi_ready.priority = 101
#                 #task.multi_ready.duration -= task.duration
                if multi_ready.status == self.IN_PRIORITY_QUEUE:
                    heap_size = len(self.tasks)
                    for k in range(heap_size//2,-1,-1):
                        self.priority_queue.heapify(k)
                elif multi_ready.status == self.NOT_STARTED:
                    multi_ready.status = self.IN_PRIORITY_QUEUE
                    self.priority_queue.heappush(multi_ready)

                task_run_second=self.priority_queue.heappop()

                print(f"⏰ Simple Scheduler at time {self.format_time(current_tim
e)} started executing task {task_run_second.id} that takes {task_run_second.dura
tion} mins")
                print('👷 Multi tasking')
                current_time += task.duration
                print(f"✅ Completed Task {task_run_second.id} - '{task_run_seco
nd.description}' at time {self.format_time(current_time)}\n")

                self.remove_dependency(task_run_second.id)
                task_run_second.status = self.COMPLETED

                #now run the longer task with just the remaining time

                current_time -= task.duration
                print(f"⏰ Simple Scheduler at time {self.format_time(current_tim
e)} started executing task {task.id} that takes {task.duration} mins")
                print('👷 Multi tasking')
                current_time += multi_ready.duration
                print(f"✅ Completed Task {task.id} - '{task.description}' at ti
me {self.format_time(current_time)}\n")
                self.remove_dependency(task.id)
                task.status = self.COMPLETED

        else: #if task is not multi-taskable
            print(f"⏰ Simple Scheduler at time {self.format_time(current_time)}
 started executing task {task.id} that takes {task.duration} mins")
            current_time += task.duration
            print(f"✅ Completed Task {task.id} - '{task.description}' at time {
self.format_time(current_time)}\n")
            # if the task is completed, it cannot be a dependency on other task
s, so remove it from the dependency list (1 line of code required)

            self.remove_dependency(task.id)
            task.status = self.COMPLETED

        return current_time

    def run_task_scheduler(self, starting_time = 480):
        """
        This function actually runs the task scheduler with the given starting_t
```

```
ime of the day.
        Input: starting time which is 8 AM
        Output: Task Schedule
        """
        current_time = starting_time
        with_constraint = []
        for task in self.tasks:
            if task.tconstraint is not False:
                with_constraint.append(task.tconstraint)
        with_constraint = sorted(with_constraint)
        constraint_index = 0
        while self.check_unscheduled_tasks() or self.priority_queue.heap_size !=
0:
            #STEPs 1 and 2: Extract tasks ready to execute (those without depend
encies) and push them into the priority queue
            self.get_tasks_ready()
            task = self.priority_queue.maxk()

            if len(self.priority_queue.heap) > 0 :  #STEP 3: Check for tasks in
 the priority queue.
                # STEP 4: get the tasks on top of the priority queue (1 line of
 code required)
                ### your code here
                if constraint_index < len(with_constraint) and task.duration > (
with_constraint[constraint_index] - current_time):
                    for index in range(len(self.tasks)):
                        if self.tasks[index].tconstraint == with_constraint[cons
traint_index]:
                            self.tasks[index].priority = 101 #give the maximum p
riority
                            task = self.tasks[index]
                            self.priority_queue.heappush(task)
                            self.priority_queue.heapify(index)
                            task = self.priority_queue.heappop()
                            if task.tconstraint - current_time > 0:
                                print("Take a break for", task.tconstraint - cur
rent_time,'minutes 😌 ')
                            current_time = task.tconstraint
                            constraint_index += 1

                            break
                else:
                    task = self.priority_queue.heappop()
#                 #task = self.priority_queue.heappop()
#                 print(f"⏰Simple Scheduler at time {self.format_time(current_t
ime)} started executing task {task.id} that takes {task.duration} mins")
#                 # go thru each multitask
#                 # add each multitask that is not complete and has no dependenc
ies to heap
#                 # heapify
#                 # pop top one

#                 # change status
#                 # filter original heap and pop the multitask that was complete
d
#                 current_time += task.duration
#                 print(f"✅ Completed Task {task.id} - '{task.description}' at
 time {self.format_time(current_time)}\n")
#                 ####### if the task is completed, it cannot be a dependency on
other tasks, so remove it from the dependency list (1 line of code required)
#                 ####### your code here
```

```
#                 self.remove_dependency(task.id)
#                 task.status = self.COMPLETED
                current_time = self.multitasking(task, current_time)
        total_time = current_time - starting_time
        print(f"🏁 Completed all planned tasks in {total_time//60}h{total_time%60:02d}min")
```

In [78]:

```
#This code reads .csv of the table and cretes lines of code that I can use to ma
nually input the tasks
import pandas as pd
df = pd.read_csv("Q1A Table - Sheet2.csv")
for x in range(0,40):
    if df.TConstraint[x] == "X":
        df.TConstraint[x] = False
    print("Task(",df.ID[x],", \'",df.Description[x],"\',",df.Minutes[x],",",df.D
ependencies[x],",",df.Multitask[x],",",df.Priority[x],",",df.TConstraint[x],"),"
)
print("Task(",df.ID[40],", \'",df.Description[40],"\',",df.Minutes[40],",",df.De
pendencies[40],",",df.Multitask[x],",",df.Priority[40],",",df.TConstraint[40],
")")
```

```
<ipython-input-78-be2c11d6fac5>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py
  df.TConstraint[x] = False
```

```
Task( 0 , ' Start working at WeWork ', 10 , [8,9,10,11] , [] , 50 ,
False ),
Task( 1 , ' Meeting Civic Partner ', 60 , [12,13,14,15] , [] , 67 ,
840 ),
Task( 2 , ' CS110 Assignment First Draft ', 120 , [16,17,18,19,20] ,
[] , 100 , False ),
Task( 3 , ' Have a meal ', 60 , [21,22,23,24,25] , [] , 78 , False
),
Task( 4 , ' Put clean clothes in closet ', 30 , [26,27,28,29,30,31,3
2] , [] , 55 , False ),
Task( 5 , ' Go to a Korean Cafe near the res ', 10 , [33,34,35] , []
, 50 , False ),
Task( 6 , ' Watch one episode of K Drama on Netflix ', 60 , [36,37,3
8] , [] , 30 , False ),
Task( 7 , ' Have a street food of Korea ', 15 , [39,40] , [] , 46 ,
1125 ),
Task( 8 , ' Reload my transportation card balance ', 5 , [] , [9] ,
75 , False ),
Task( 9 , ' Find which WeWork branch that I want to go on that day
', 5 , [] , [8] , 65 , False ),
Task( 10 , ' Book a desk on WeWork app ', 5 , [9] , [] , 65 , False
),
Task( 11 , ' Rid bus to WeWork ', 20 , [8,9,10] , [] , 60 , False ),
Task( 12 , ' Review the meeting agenda ', 10 , [] , [] , 80 , 805 ),
Task( 13 , ' Find appropriate outfit for the meeting ', 10 , [] , []
, 30 , 815 ),
Task( 14 , ' Set up necesscities like laptop, and iPad ', 10 , [] ,
[] , 90 , 825 ),
Task( 15 , ' Remind Group members that we have meeting with meeting
agenda ', 5 , [12] , [] , 30 , 835 ),
Task( 16 , ' Plug in my laptop to a charger ', 5 , [] , [] , 46 , Fa
lse ),
Task( 17 , ' Open Forum ', 5 , [16] , [] , 70 , False ),
Task( 18 , ' Read Assignment Instruction ', 10 , [16,17] , [] , 90 ,
False ),
Task( 19 , ' Open Jupyter notebook ', 5 , [16] , [] , 70 , False ),
Task( 20 , ' Solve Q1 to Q3 ', 60 , [16,17,18,19] , [] , 96 , False
),
Task( 21 , ' Confirm which groceries I have ', 5 , [] , [22] , 10 ,
False ),
Task( 22 , ' Prepare utensils ', 5 , [] , [21] , 5 , False ),
Task( 23 , ' Find recipe ', 5 , [21] , [] , 5 , False ),
Task( 24 , ' Cook ', 15 , [21,22,23] , [] , 11 , False ),
Task( 25 , ' Store in containers ', 5 , [21,22,23,24] , [] , 46 , Fa
lse ),
Task( 26 , ' Pack the laundry in the laundry bag ', 5 , [] , [] , 3
, False ),
Task( 27 , ' Go to the nearest self-washing machine store ', 5 , [2
6] , [] , 15 , False ),
Task( 28 , ' Put the laundry into the machine ', 5 , [26,27] , [] ,
13 , False ),
Task( 29 , ' Run the machine ', 20 , [26,27,28] , [] , 18 , False ),
Task( 30 , ' Take out laundry ', 5 , [26,27,28,29] , [] , 40 , False
),
Task( 31 , ' Bring laundry back to the res ', 5 , [26,27,28,29,30] ,
[] , 80 , False ),
Task( 32 , ' Dry them on the rack ', 60 , [26,27,28,29,30,31] , [] ,
85 , False ),
Task( 33 , ' Google the top instagrammable cafe near the res ', 5 ,
[] , [] , 30 , False ),
Task( 34 , ' Leave the res hall ', 5 , [33] , [] , 5 , False ),
```

```
Task( 35 , ' Walk to the cafe ', 5 , [33,34] , [] , 3 , False ),
Task( 36 , ' Login to Netflix ', 5 , [] , [] , 3 , False ),
Task( 37 , ' Find Trending in Korea ', 5 , [36] , [] , 4 , False ),
Task( 38 , ' Start Episode 1 for the Top 1 Trending in Korea ', 5 ,
[36,37] , [] , 10 , False ),
Task( 39 , ' Find a random street vendor near Sookmyung University
', 5 , [] , [] , 20 , 1115 ),
Task( 40 , ' Order Tteokbokki ', 5 , [39] , [] , 37 , 1120 )
```

In [79]:

```
tasks = [
Task( 0 , ' Start working at WeWork ', 10 , [8,9,10,11] , [] , 50 , False ),
Task( 1 , ' Meeting Civic Partner ', 60 , [12,13,14,15] , [] , 67 , 840 ),
Task( 2 , ' CS110 Assignment First Draft ', 120 , [16,17,18,19,20] , [] , 100 ,
False ),
Task( 3 , ' Have a meal ', 60 , [21,22,23,24,25] , [] , 78 , False ),
Task( 4 , ' Put clean clothes in closet ', 30 , [26,27,28,29,30,31,32] , [] , 55
, False ),
Task( 5 , ' Go to a Korean Cafe near the res ', 10 , [33,34,35] , [] , 50 , Fals
e ),
Task( 6 , ' Watch one episode of K Drama on Netflix ', 60 , [36,37,38] , [] , 30
, False ),
Task( 7 , ' Have a street food of Korea ', 15 , [39,40] , [] , 46 , 1125 ),
Task( 8 , ' Reload my transportation card balance ', 5 , [] , [9] , 75 , False
),
Task( 9 , ' Find which WeWork branch that I want to go on that day ', 5 , [] , [
8] , 65 , False ),
Task( 10 , ' Book a desk on WeWork app ', 5 , [9] , [] , 65 , False ),
Task( 11 , ' Rid bus to WeWork ', 20 , [8,9,10] , [] , 60 , False ),
Task( 12 , ' Review the meeting agenda ', 10 , [] , [] , 80 , 805 ),
Task( 13 , ' Find appropriate outfit for the meeting ', 10 , [] , [] , 30 , 815
),
Task( 14 , ' Set up necesscities like laptop, and iPad ', 10 , [] , [] , 90 , 82
5 ),
Task( 15 , ' Remind Group members that we have meeting with meeting agenda ', 5
, [12] , [] , 30 , 835 ),
Task( 16 , ' Plug in my laptop to a charger ', 5 , [] , [] , 46 , False ),
Task( 17 , ' Open Forum ', 5 , [16] , [] , 70 , False ),
Task( 18 , ' Read Assignment Instruction ', 10 , [16,17] , [] , 90 , False ),
Task( 19 , ' Open Jupyter notebook ', 5 , [16] , [] , 70 , False ),
Task( 20 , ' Solve Q1 to Q3 ', 60 , [16,17,18,19] , [] , 96 , False ),
Task( 21 , ' Confirm which groceries I have ', 5 , [] , [22] , 10 , False ),
Task( 22 , ' Prepare utensils ', 5 , [] , [21] , 5 , False ),
Task( 23 , ' Find recipe ', 5 , [21] , [] , 5 , False ),
Task( 24 , ' Cook ', 15 , [21,22,23] , [] , 11 , False ),
Task( 25 , ' Store in containers ', 5 , [21,22,23,24] , [] , 46 , False ),
Task( 26 , ' Pack the laundry in the laundry bag ', 5 , [] , [] , 3 , False ),
Task( 27 , ' Go to the nearest self-washing machine store ', 5 , [26] , [] , 15
, False ),
Task( 28 , ' Put the laundry into the machine ', 5 , [26,27] , [] , 13 , False
),
Task( 29 , ' Run the machine ', 20 , [26,27,28] , [] , 18 , False ),
Task( 30 , ' Take out laundry ', 5 , [26,27,28,29] , [] , 40 , False ),
Task( 31 , ' Bring laundry back to the res ', 5 , [26,27,28,29,30] , [] , 80 , F
alse ),
Task( 32 , ' Dry them on the rack ', 60 , [26,27,28,29,30,31] , [] , 85 , False
),
Task( 33 , ' Google the top instagrammable cafe near the res ', 5 , [] , [] , 30
, False ),
Task( 34 , ' Leave the res hall ', 5 , [33] , [] , 5 , False ),
Task( 35 , ' Walk to the cafe ', 5 , [33,34] , [] , 3 , False ),
Task( 36 , ' Login to Netflix ', 5 , [] , [] , 3 , False ),
Task( 37 , ' Find Trending in Korea ', 5 , [36] , [] , 4 , False ),
Task( 38 , ' Start Episode 1 for the Top 1 Trending in Korea ', 5 , [36,37] , []
, 10 , False ),
Task( 39 , ' Find a random street vendor near Sookmyung University ', 5 , [] ,
[] , 20 , 1115 ),
Task( 40 , ' Order Tteokbokki ', 5 , [39] , [] , 37 , 1120 )]
task_scheduler = TaskScheduler(tasks)
```

```
task_scheduler.run_task_scheduler()

# print the scheduler's input (1 line of code required)
# your code here
```

⏰Simple Scheduler at time 8h00 started executing task 9 that takes 5 mins
👨‍💻Multi tasking
✅ Completed Task 9 - ' Find which WeWork branch that I want to go on that day ' at time 8h05

⏰Simple Scheduler at time 8h00 started executing task 8 that takes 5 mins
👨‍💻Multi tasking
✅ Completed Task 8 - ' Reload my transportation card balance ' at time 8h05

⏰Simple Scheduler at time 8h05 started executing task 10 that takes 5 mins
✅ Completed Task 10 - ' Book a desk on WeWork app ' at time 8h10

⏰Simple Scheduler at time 8h10 started executing task 11 that takes 20 mins
✅ Completed Task 11 - ' Rid bus to WeWork ' at time 8h30

⏰Simple Scheduler at time 8h30 started executing task 0 that takes 10 mins
✅ Completed Task 0 - ' Start working at WeWork ' at time 8h40

⏰Simple Scheduler at time 8h40 started executing task 16 that takes 5 mins
✅ Completed Task 16 - ' Plug in my laptop to a charger ' at time 8h45

⏰Simple Scheduler at time 8h45 started executing task 17 that takes 5 mins
✅ Completed Task 17 - ' Open Forum ' at time 8h50

⏰Simple Scheduler at time 8h50 started executing task 18 that takes 10 mins
✅ Completed Task 18 - ' Read Assignment Instruction ' at time 9h00

⏰Simple Scheduler at time 9h00 started executing task 19 that takes 5 mins
✅ Completed Task 19 - ' Open Jupyter notebook ' at time 9h05

⏰Simple Scheduler at time 9h05 started executing task 20 that takes 60 mins
✅ Completed Task 20 - ' Solve Q1 to Q3 ' at time 10h05

⏰Simple Scheduler at time 10h05 started executing task 2 that takes 120 mins
✅ Completed Task 2 - ' CS110 Assignment First Draft ' at time 12h05

⏰Simple Scheduler at time 12h05 started executing task 33 that takes 5 mins
✅ Completed Task 33 - ' Google the top instagrammable cafe near the res ' at time 12h10

⏰Simple Scheduler at time 12h10 started executing task 22 that takes 5 mins
👨‍💻Multi tasking
✅ Completed Task 22 - ' Prepare utensils ' at time 12h15

⏰Simple Scheduler at time 12h10 started executing task 21 that takes 5 mins

👷 Multi tasking
✅ Completed Task 21 - ' Confirm which groceries I have ' at time 12h15

⏰ Simple Scheduler at time 12h15 started executing task 34 that takes 5 mins
✅ Completed Task 34 - ' Leave the res hall ' at time 12h20

⏰ Simple Scheduler at time 12h20 started executing task 23 that takes 5 mins
✅ Completed Task 23 - ' Find recipe ' at time 12h25

⏰ Simple Scheduler at time 12h25 started executing task 24 that takes 15 mins
✅ Completed Task 24 - ' Cook ' at time 12h40

⏰ Simple Scheduler at time 12h40 started executing task 25 that takes 5 mins
✅ Completed Task 25 - ' Store in containers ' at time 12h45

Take a break for 40 minutes 😌
⏰ Simple Scheduler at time 13h25 started executing task 12 that takes 10 mins
✅ Completed Task 12 - ' Review the meeting agenda ' at time 13h35

⏰ Simple Scheduler at time 13h35 started executing task 13 that takes 10 mins
✅ Completed Task 13 - ' Find appropriate outfit for the meeting ' at time 13h45

⏰ Simple Scheduler at time 13h45 started executing task 14 that takes 10 mins
✅ Completed Task 14 - ' Set up necesscities like laptop, and iPad ' at time 13h55

⏰ Simple Scheduler at time 13h55 started executing task 15 that takes 5 mins
✅ Completed Task 15 - ' Remind Group members that we have meeting with meeting agenda ' at time 14h00

⏰ Simple Scheduler at time 14h00 started executing task 1 that takes 60 mins
✅ Completed Task 1 - ' Meeting Civic Partner ' at time 15h00

⏰ Simple Scheduler at time 15h00 started executing task 3 that takes 60 mins
✅ Completed Task 3 - ' Have a meal ' at time 16h00

⏰ Simple Scheduler at time 16h00 started executing task 36 that takes 5 mins
✅ Completed Task 36 - ' Login to Netflix ' at time 16h05

⏰ Simple Scheduler at time 16h05 started executing task 37 that takes 5 mins
✅ Completed Task 37 - ' Find Trending in Korea ' at time 16h10

⏰ Simple Scheduler at time 16h10 started executing task 38 that takes 5 mins
✅ Completed Task 38 - ' Start Episode 1 for the Top 1 Trending in Korea ' at time 16h15

⏰Simple Scheduler at time 16h15 started executing task 6 that takes 60 mins
✅ Completed Task 6 - ' Watch one episode of K Drama on Netflix ' at time 17h15

⏰Simple Scheduler at time 17h15 started executing task 26 that takes 5 mins
✅ Completed Task 26 - ' Pack the laundry in the laundry bag ' at time 17h20

⏰Simple Scheduler at time 17h20 started executing task 27 that takes 5 mins
✅ Completed Task 27 - ' Go to the nearest self-washing machine store ' at time 17h25

⏰Simple Scheduler at time 17h25 started executing task 28 that takes 5 mins
✅ Completed Task 28 - ' Put the laundry into the machine ' at time 17h30

⏰Simple Scheduler at time 17h30 started executing task 29 that takes 20 mins
✅ Completed Task 29 - ' Run the machine ' at time 17h50

⏰Simple Scheduler at time 17h50 started executing task 30 that takes 5 mins
✅ Completed Task 30 - ' Take out laundry ' at time 17h55

⏰Simple Scheduler at time 17h55 started executing task 31 that takes 5 mins
✅ Completed Task 31 - ' Bring laundry back to the res ' at time 18h00

Take a break for 35 minutes 😌
⏰Simple Scheduler at time 18h35 started executing task 39 that takes 5 mins
✅ Completed Task 39 - ' Find a random street vendor near Sookmyung University ' at time 18h40

⏰Simple Scheduler at time 18h40 started executing task 40 that takes 5 mins
✅ Completed Task 40 - ' Order Tteokbokki ' at time 18h45

⏰Simple Scheduler at time 18h45 started executing task 7 that takes 15 mins
✅ Completed Task 7 - ' Have a street food of Korea ' at time 19h00

⏰Simple Scheduler at time 19h00 started executing task 32 that takes 60 mins
✅ Completed Task 32 - ' Dry them on the rack ' at time 20h00

⏰Simple Scheduler at time 20h00 started executing task 4 that takes 30 mins
✅ Completed Task 4 - ' Put clean clothes in closet ' at time 20h30

⏰Simple Scheduler at time 20h30 started executing task 35 that takes 5 mins
✅ Completed Task 35 - ' Walk to the cafe ' at time 20h35

⏰Simple Scheduler at time 20h35 started executing task 5 that takes 10 mins

✅ Completed Task 5 - ' Go to a Korean Cafe near the res ' at time 20h45

🏁 Completed all planned tasks in 12h45min

Q5 / Summary

A)

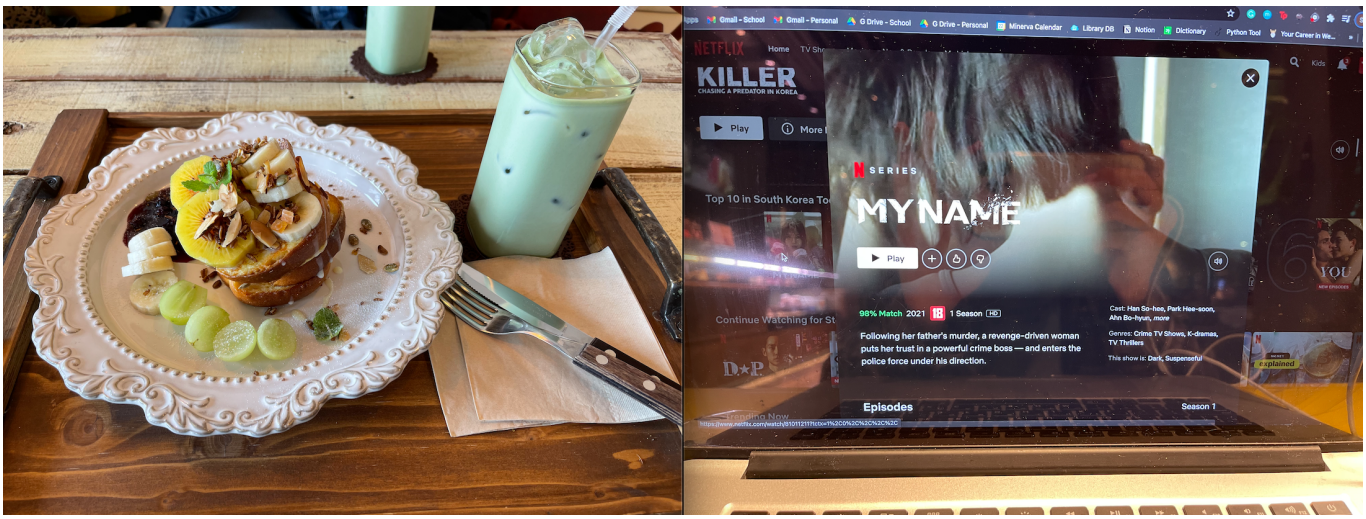
Produce a critical analysis of your scheduler, including pictures you take for this test drive highlighting:
• all the benefits in following the algorithmic directives defined in the instructions (rather than deciding on the spot where to go next!),
• and any failure modes and/or limitations you envision it running into.

B)

Examine the efficiency of your schedule (not the scheduler) and include any explicit reference to the metrics you employed to determine this.

C)

Will you start using your algorithm to schedule your day? Explain your answer in as much detail as possible.



I've been planning my days as whichever pops up in my mind. I always felt so rushed to finish all of them in a day. Of course, there are more important tasks that I have to do every day, but I planned a day without really prioritizing those tasks. This certainly made my life harder such as cramming CS110 Pre Class Work 2-3 AM. It was not a good move. Prioritizing with MaxHeap definitely uses time efficiently, and I observed that I waste less time if I plan a day accordingly. This observation comes from my experience. I had the same tasks a day before I started the assignment but didn't code and repeated the same tasks after I coded.

When I didn't have the scheduler and worked, I started the first task at 8 AM but finished at 2 AM. I spent a total of 18 hours finishing. In reality, I could multi-task, but my procrastination and personal things bother me to stay organized. Once I had a scheduler, my scheduler estimated that my tasks would be done by 8:10 PM. After following the scheduler, I observed myself putting under time constraints as a scheduler result. This made me have a more tangible scheduler, not thinking out loud. As a result, I less procrastinated and was able to finish all the tasks before 7 PM.

Since it was a basic project that gives the point of having a scheduler; however, I still see the potentials of this project and think of using this algorithm in my life.

If the following, which I tried to implement in this project but failed, are improved, I think this task scheduler should be ready to use in real life.

1. Consider geographical locations. => It is better to do activities in similar locations and then move to another location.
2. I wanted to build a code that still does every sub-task before the main task with time constraints without defining time constraints for every sub-tasks
3. Give more flexibility in scheduling. => Rather than doing tasks continuously, I want to include short 5-10 minutes break every two hours.
4. Expand this scheduler to a weekly scheduler. => In some cases, if I have too many things to do in a day and impossible to finish them in a day, I wanted to expand this scheduler that I can push some tasks to a day after and continue.

There should be more improvements that I can make toward this. I look forward to continuing work on this project to make a decent scheduler that many users can use. Stay tuned! (436 words)

LO Appendix

It is not a required section; however, I include this so that the professor can see how I kept the mind of LOs in this assignment.

cs110-DataStructures: Using Heap Data structure in this assignment, I provided a clear and detailed explanation of choice. For example, as I defined the greater number of prioritized tasks that should be done earlier than the other, I explained that the Max Heap is useful for popping the most prioritized task. Even though it was not mentioned in the assignment, it is not a good choice to use other sorts. It is because that we have some tasks with dependencies and constraints. When a task has dependencies, sub-tasks in this assignment, but those sub-tasks have lower priority than other tasks. Implementing this into code would be more complicated as we have to break the sorted array and keep breaking until all the dependencies come before the task with dependencies. In this case, we basically break the sorted array. Therefore it is not useful to do so in that way. This proves that we need a heap sort at the end; therefore, we need heap data structure.

cs110-PythonProgramming: I wrote every python code correctly, accurately, and efficiently. I successfully implemented heaps in this code and upgraded the task scheduler we had from the class that can implement the time constraint and multitasking. Even I didn't include the test cases in the notebook; I tried to verify whether the code was working based on the result I manually did. I also created two more simple task lists with dependencies and time constraints and repeated the manual steps to see if they were working correctly in those cases too.

cs110-CodeReadability: I included comments and docstrings appropriately in detail so when one reads this notebook, she/he/they will easily follow the code and understand. Some parts of the code include the error messages that can be used in debugging process. I named variables, lists, and etc., in a manner that one can easily understand what they stand for.

cs110-AlgorithmicStrategies: I explicitly explained how heap sort works based on the steps and clearly defined what each method does for heap sort. I explained why heap is useful here to implement the priority queues in detail and justified well. I explained why other sorting methods are not useful in this project in #cs110-DataStructures LO appendix.

HC Appendix

algorithms: I successfully identified how each function of MaxHeap can be used to implement the task scheduler. This assignment shows a strong understanding in Heap Structure, and all of the codes are clearly commented. I used a decision tree (mentioned below) to implement the pseudocode of this algorithm to visualize and understand.

constraints: Applied this HC on some tasks that have to be done at the exact time. This states that all the sub-tasks should be done before the big task with constraints. While designing an algorithm, I found it more logical to do all the subtasks and a big task in a row because I would be in the same location and continue working on similar things. Setting constraints is helpful to schedule a task because one can divide the day as intervals of constrained tasks. It is important to understand that life, in reality, is not flexible, and sometimes there are things that I have to do even I have other things to do. From this application of HC, I thought that I could change the way of scheduling my tasks every day from writing a to-do list to distributing tasks after setting constrained tasks first.

decisiontrees: I found that a decision tree is useful to double-check that the task scheduler is correct. Even this application does not include any numerical values to think about which decisions to make; however, the idea of decision trees that visualize the decision-making process is much easier to understand, only looking at the code to think about how the algorithm works. I included For/While loops in the decision tree to repeat the process and conditional statements to split the branches. I tried with the raw data of input that I have from the spreadsheet and manually saw every process of the algorithm. It was an effective application of this HC as it allowed me to find where my code/algorithm breaks and reduced the time I used to debug.

optimization: Even though optimization strategy is not directly applied to the codes; however, this optimizes the loss that I can potentially get from missing a task that I was supposed to do in a day. I defined the priority value as to how much I feel losing if I don't do the task. By doing high prioritized tasks first; therefore, I lose less even I could not finish other tasks. However, this HC application would be applied better if I calculated the loss with just feeling but it also depends on time. For example, I have 200 tasks with priority 1, which takes a minute, and 1 task with priority 100 but takes 24 hours to finish. Based on the implemented algorithm, priority 100 will be done first, but it ends my day. This results in -200 of loss. However, if I finish 200 tasks first, it will result that I cannot finish the task that has 100 priority in a day but still has -100 of loss. To better apply this HC, I will consider implementing code that also considers duration related to the priority value next time.

professionalism: I used grammarly to make sure there are no grammar mistakes. I followed the template that I used for Assignment 1 to make a template for this assignment is professional with correct markdowns.