CS110: Designing a Scheduler

November 17, 2024

Navigation

Click on the links below to navigate to different sections of the document:

1 Setting Up

	id	description	duration	dependencies
0	0	Write grocery checklist	50	No Dependency
1	1	Do groceries at PX Mart	50	Do groceries at PX Mart
2	2	Charge Computer before class	50	No Dependency
3	3	Do Pre-Class Work for CS110	100	Do Pre-Class Work for CS110
4	4	Vacuum the floor	20	No Dependency
5	5	Withdraw cash from 7-11 ATM	15	No Dependency
6	6	CS110 Class	90	CS110 Class
7	7	Take NTU id card from the room	5	No Dependency
8	8	Go to the NTU library through YouBike	20	Go to the NTU library through YouBike
9	9	Take the MRT train to the City Science Lab @ T...	40	Take the MRT train to the City Science Lab @ T...
10	10	Bike back to Halfway Cafe in the Daan District...	30	Bike back to Halfway Cafe in the Daan District...

Regular Tasks:

- Write grocery checklist
- Do groceries at PX Mart
- Charge Computer before class
- Do Pre-Class Work for CS110
- Vacuum the floor
- Withdraw cash from 7-11 ATM
- CS110 Class

Taipei centered tasks:

- Take NTU id card from the room
- Go to the NTU library through YouBike
- Take the MRT train to the City Science Lab @ Taipei from MIT
- Bike back to Halfway Cafe in the Daan District to study until late

2 Algorithmic Strategy

Video Link: Google Drive

3 Python Implementation

3.1 My own OOP implementation of MaxHeapq

Citation: Levay, 2023.

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

    Attributes
    ----------
    heap : list
        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):
        """
        Initializes an empty MaxHeapq instance.

        Parameters
        ----------
        None
        """
        self.heap = []
        self.heap_size = 0

    def is_max_heap(self):
        """
        Checks if the current list satisfies the max heap property.

        Returns
        -------
        bool
            True if the list is a max heap, False otherwise.
        """
        a = self.heap
        n = self.heap_size

        # Check each node to ensure it is greater than its children
        for i in range(n):
            left_child = self.left(i)
            right_child = self.right(i)

            # Check if left child exists and is greater than parent
            if left_child < n and a[i] < a[left_child]:
                return False

            # Check if right child exists and is greater than parent
```

```
            if right_child < n and a[i] < a[right_child]:
                return False
        return True


    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 the 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 the 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 the child node.

        Returns
        -------
        int
            Index of the parent node.
        """
```

```
        return (i - 1) // 2

    def heappush(self, key):
        """
        Inserts a key into the priority queue.

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

        Returns
        -------
        None
            This method modifies the heap in place.
        """

        # Append a placeholder for new key and then increase its value to correct
        ↪ position
        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.

        Raises
        ------
        ValueError
            If new key is smaller than current key.

        Returns
        -------
        None
            This method modifies the heap in place.
        """

        if key.priority_score < 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)] < self.heap[i]:
            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 by ensuring that subtree rooted at
        ↪  index i satisfies max heap property.

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

        Returns
        -------
        None
            This method modifies the heap in place.
        """
        l = self.left(i)
        r = self.right(i)
        largest = i

        # Check if left child exists and is greater than current largest
        if l < self.heap_size and self.heap[l] > self.heap[largest]:
            largest = l

        # Check if right child exists and is greater than current largest
        if r < self.heap_size and self.heap[r] > self.heap[largest]:
            largest = r

        # If largest is not current index, swap and continue heapifying
        if largest != i:
            self.heap[i], self.heap[largest] = self.heap[largest], self.heap[i]
            self.heapify(largest)

    # Novel method
    def anypop(self, i):
        """
        Returns an arbitrary element from the max priority queue and removes it from
        ↪  it.

        Parameters
        ----------
        i : int
            The index of an element to pop from the heap.

        Raises
        ------
        ValueError
            If there are no keys in priority queue (heap underflow).

        Returns
        -------
```

```
        int
            The value extracted from the heap at index i.
        """
        if self.heap_size < 1:
            raise ValueError('Heap underflow: There are no keys in priority queue.')

        any_value = self.heap[i]  # Store value to return later

        # Replace root with last element, pop last element, decrease size, then
        ↪ re-heapify
        self.heap[i] = self.heap[-1]
        self.heap.pop()
        self.heap_size -= 1

        # Re-heapify from index i to maintain max heap property
        self.heapify(i)

        return any_value

    def heappop(self):
        """
        Returns and removes the largest key in the max priority queue.

        Raises
        ------
        ValueError
            If there are no keys in priority queue (heap underflow).

        Returns
        -------
        int
            The maximum value extracted from the heap (the root).
        """

        if self.heap_size < 1:
            raise ValueError('Heap underflow: There are no keys in priority queue.')

        max_value = self.heap[0]

        self.heap[0] = self.heap[-1]
        self.heap.pop()
        self.heap_size -= 1

        self.heapify(0)

        return max_value
```

3.2 Test Cases for MaxHeap

```
heap = MaxHeapq()

# fully independent tasks
simple_tasks = [
    Task(id=0, description='Write grocery checklist', duration=50, dependencies=[]),
    Task(id=1, description='Vacuum the floor', duration=20, dependencies=[]),
    Task(id=2, description='Withdraw cash from 7-11 ATM', duration=15,
    ↪  dependencies=[]),
    Task(id=3, description='Do groceries at PX Mart', duration=50,
    ↪  dependencies=[0]),
    Task(id=4, description='Charge Computer before class', duration=50,
    ↪  dependencies=[]),
    Task(id=5, description='Do Pre-Class Work for CS110', duration=100,
    ↪  dependencies=[4]),
]

# assume current time if 8am
current_time = 8*60

for task in simple_tasks:
  task.compute_priority(current_time)

# Insert all tasks into the heap
for task in simple_tasks:
    heap.heappush(task)

assert heap.is_max_heap(), "Heap does not maintain max-heap property after task
↪  insertions."
assert heap.heap[0].description == 'Withdraw cash from 7-11 ATM', "Heappop did not
↪  remove the correct task."
```

```
heap = MaxHeapq()

# fully independent tasks
simple_tasks = [
    Task(id=0, description='Vacuum the floor', duration=20, dependencies=[]),
    Task(id=1, description='Withdraw cash from 7-11 ATM', duration=15,
    ↪  dependencies=[]),
    Task(id=2, description='Do groceries at PX Mart', duration=50, dependencies=[]),
    Task(id=3, description='Charge Computer before class', duration=50,
    ↪  dependencies=[]),
    Task(id=4, description='Do Pre-Class Work for CS110', duration=100,
    ↪  dependencies=[]),
]

# assume current time is 8am
current_time = 8*60

```

```
for task in simple_tasks:
  task.compute_priority(current_time)

for task in simple_tasks:
    heap.heappush(task)

max_priority_task = heap.heappop()
print(max_priority_task)

# Verify the task with the highest priority (shortest duration) is removed
assert max_priority_task.description == 'Withdraw cash from 7-11 ATM', "Heappop did not remove the correct task."
assert heap.is_max_heap()
```

```
heap = MaxHeapq()

# fully independent tasks
simple_tasks = [
Task(id=0, description='Vacuum the floor', duration=20, dependencies=[]),
Task(id=1, description='Withdraw cash from 7-11 ATM', duration=15, dependencies=[]),
Task(id=2, description='Do groceries at PX Mart', duration=50, dependencies=[]),
Task(id=3, description='Charge Computer before class', duration=50, dependencies=[]),
Task(id=4, description='Do Pre-Class Work for CS110', duration=100, dependencies=[]),
]

# assume current time is 8am
current_time = 8*60

for task in simple_tasks:
task.compute_priority(current_time)

for task in simple_tasks:
heap.heappush(task)

removed_task = heap.anypop(2)  # Arbitrarily remove task at index 2 (Withdraw cash from 7-11 ATM)

assert removed_task.description == 'Do groceries at PX Mart', "anypop did not remove the correct task."
assert heap.is_max_heap(), "Heap does not maintain max-heap property after anypop."
```

3.3 Simple Example of a Scheduler

```
simple_tasks = [
Task(id=0, description='Write grocery checklist', duration=50, dependencies=[]),
Task(id=1, description='Vacuum the floor', duration=20, dependencies=[]),
Task(id=2, description='Withdraw cash from 7-11 ATM', duration=15, dependencies=[]),
Task(id=3, description='Do groceries at PX Mart', duration=50, dependencies=[0]),
Task(id=4, description='Charge Computer before class', duration=50,
↪  dependencies=[]),
Task(id=5, description='Do Pre-Class Work for CS110', duration=100,
↪  dependencies=[4]),
]
```

Output with a priority queue, to demonstrate how the max-heap property is met:

```
This is my priority queue:
0) Task 2: Withdraw cash from 7-11 ATM
    Duration: 15 minutes
    Status: I
    Priority Score = 85

1) Task 0: Write grocery checklist
    Duration: 50 minutes
    Status: I
    Priority Score = 50

2) Task 1: Vacuum the floor
    Duration: 20 minutes
    Status: I
    Priority Score = 80

3) Task 4: Charge Computer before class
    Duration: 50 minutes
    Status: I
    Priority Score = 50

t=8h00: started 'Withdraw cash from 7-11 ATM' for 15 mins...
    | t=8h15, task completed!

This is my priority queue:
0) Task 1: Vacuum the floor
    Duration: 20 minutes
    Status: I
    Priority Score = 80

1) Task 0: Write grocery checklist
    Duration: 50 minutes
    Status: I
    Priority Score = 50

```

```
2) Task 4: Charge Computer before class
    Duration: 50 minutes
    Status: I
    Priority Score = 50

t=8h15: started 'Vacuum the floor' for 20 mins...
    | t=8h35, task completed!

This is my priority queue:
0) Task 4: Charge Computer before class
    Duration: 50 minutes
    Status: I
    Priority Score = 50

1) Task 0: Write grocery checklist
    Duration: 50 minutes
    Status: I
    Priority Score = 50

t=8h35: started 'Charge Computer before class' for 50 mins...
    | t=9h25, task completed!

This is my priority queue:
0) Task 0: Write grocery checklist
    Duration: 50 minutes
    Status: I
    Priority Score = 50

1) Task 5: Do Pre-Class Work for CS110
    Duration: 100 minutes
    Status: I
    Priority Score = 0

t=9h25: started 'Write grocery checklist' for 50 mins...
    | t=10h15, task completed!

This is my priority queue:
0) Task 3: Do groceries at PX Mart
    Duration: 50 minutes
    Status: I
    Priority Score = 50

1) Task 5: Do Pre-Class Work for CS110
    Duration: 100 minutes
    Status: I
    Priority Score = 0

t=10h15: started 'Do groceries at PX Mart' for 50 mins...
    | t=11h05, task completed!

This is my priority queue:
0) Task 5: Do Pre-Class Work for CS110
    Duration: 100 minutes
```

```
    Status: I
    Priority Score = 0

t=11h05: started 'Do Pre-Class Work for CS110' for 100 mins...
    | t=12h45, task completed!


Completed all planned tasks in 4h45min!
```

3.4 Changing the order of the inputs leads to the same output

```
# Original order
tasks = [
    Task(id=0, description='Get up at 8:00 AM',
         duration=30, dependencies=[]),
    Task(id=1, description='Walk to 7-11 and have breakfast',
         duration=20, dependencies=[0]),
    Task(id=2, description='Bike to the Main NTU Library',
         duration=12, dependencies=[0, 1]),
    Task(id=3, description='Do Pre-Class Work for CS111',
         duration=50, dependencies=[2]),
    Task(id=4, description='Formalize the Utility function definition for CS110
    ↪  Assignment',
         duration=90, dependencies=[2]),
    Task(id=5, description='Prepare video scripts and whiteboards for CS110
    ↪  Assignment',
         duration=80, dependencies=[2]),
    Task(id=6, description="Have lunch at JJ's Poke Bowl and bike back to the
    ↪  library",
         duration=75, dependencies=[2], scheduled_time=13*60),
    Task(id=7, description='CS111 Class',
         duration=90, dependencies=[3], scheduled_time=18*60),
    Task(id=8, description='Record CS110 Videos',
         duration=75, dependencies=[4, 5]),
    Task(id=9, description='Work on written parts of CS110 Assignment',
         duration=120, dependencies=[8]),
    Task(id=10, description='Bike back to 9Floor after the library closes at 22h',
         duration=12, dependencies=[6, 7, 8, 9], scheduled_time=22*60)
    ]

task_scheduler = TaskScheduler(tasks)

# print the scheduler's input
task_scheduler.print_self()

start_scheduler_at = 8*60
task_scheduler.run_task_scheduler(start_scheduler_at)

expected_order = task_scheduler.executed_task_descriptions
```

```
36
37  # Different order 1
38  diff_order_1 = [
39      Task(id=0, description='Bike back to 9Floor after the library closes at 22h',
40           duration=12, dependencies=[7, 8, 9, 10]),
41      Task(id=1, description='Get up at 8:00 AM',
42           duration=30, dependencies=[]),
43      Task(id=2, description='Walk to 7-11 and have breakfast',
44           duration=20, dependencies=[1]),
45      Task(id=3, description='Bike to the Main NTU Library',
46           duration=12, dependencies=[1, 2]),
47      Task(id=4, description='Do Pre-Class Work for CS111',
48           duration=50, dependencies=[3]),
49      Task(id=5, description='Formalize the Utility function definition for CS110
        ↪  Assignment',
50           duration=90, dependencies=[3]),
51      Task(id=6, description='Prepare video scripts and whiteboards for CS110
        ↪  Assignment',
52           duration=80, dependencies=[3]),
53      Task(id=7, description="Have lunch at JJ's Poke Bowl and bike back to the
        ↪  library",
54           duration=75, dependencies=[3], scheduled_time=13*60),
55      Task(id=8, description='CS111 Class',
56           duration=90, dependencies=[4], scheduled_time=18*60),
57      Task(id=9, description='Record CS110 Videos',
58           duration=75, dependencies=[5, 6]),
59      Task(id=10, description='Work on written parts of CS110 Assignment',
60           duration=120, dependencies=[9])
61  ]
62
63  task_scheduler_diff_order_1 = TaskScheduler(diff_order_1)
64
65  task_scheduler_diff_order_1.print_self()
66
67  start_scheduler_at = 8*60
68  task_scheduler_diff_order_1.run_task_scheduler(start_scheduler_at)
69
70  task_descriptions_diff_order_1 =
    ↪  task_scheduler_diff_order_1.executed_task_descriptions
71
72  assert task_descriptions_diff_order_1 == expected_order, f"Mismatch found:
    ↪  {task_descriptions_diff_order_1} != {expected_order}"
73
74  # Different order 2
75  diff_order_2 = [
76      Task(id=0, description='Get up at 8:00 AM',
77           duration=30, dependencies=[]),
78      Task(id=1, description='Walk to 7-11 and have breakfast',
79           duration=20, dependencies=[0]),
80      Task(id=2, description='Bike to the Main NTU Library',
81           duration=12, dependencies=[0, 1]),
82      Task(id=3, description='Have lunch at JJ\'s Poke Bowl and bike back to the
        ↪  library',
```

```
        duration=75, dependencies=[2], scheduled_time=13*60),
    Task(id=4, description='Do Pre-Class Work for CS111',
        duration=50, dependencies=[2]),
    Task(id=5, description='Formalize the Utility function definition for CS110
    ↪ Assignment',
        duration=90, dependencies=[2]),
    Task(id=6, description='Record CS110 Videos',
        duration=75, dependencies=[5, 10]),
    Task(id=7, description='CS111 Class',
        duration=90, dependencies=[4], scheduled_time=18*60),
    Task(id=8, description='Bike back to 9Floor after the library closes at 22h',
        duration=12, dependencies=[3, 6, 7, 9]),
    Task(id=9, description='Work on written parts of CS110 Assignment',
        duration=120, dependencies=[6]),
    Task(id=10, description='Prepare video scripts and whiteboards for CS110
    ↪ Assignment',
        duration=80, dependencies=[2])
]

task_scheduler_diff_order_2 = TaskScheduler(diff_order_2)

task_scheduler_diff_order_2.print_self()

start_scheduler_at = 8*60
task_scheduler_diff_order_2.run_task_scheduler(start_scheduler_at)

task_descriptions_diff_order_2 =
↪ task_scheduler_diff_order_2.executed_task_descriptions

assert task_descriptions_diff_order_2 == expected_order, f"Mismatch found:
↪ {task_descriptions_diff_order_2} != {expected_order}"
```

3.5 Are schedules feasible?

The current output of schedules so far leads to a sequence of tasks that can be performed in a feasible amount of time, without further conflicts. However, I do anticipate a few issues already with task rescheduling, times without any fill-in tasks, or delays due to uncontrollable factors like public transportation.

```
Running a simple scheduler:

t=8h00: started 'Get up at 8:00 AM' for 30 mins...
    | t=8h30, task completed!

t=8h30: started 'Walk to 7-11 and have breakfast' for 20 mins...
    | t=8h50, task completed!

t=8h50: started 'Bike to the Main NTU Library' for 12 mins...
    | t=9h02, task completed!
```

```
11
12  Fill-in task available! t=9h02: started 'Do Pre-Class Work for CS111' for 50 mins...
13      | t=9h52, task completed!
14
15  Fill-in task available! t=9h52: started 'Prepare video scripts and whiteboards for
    ↪  CS110 Assignment' for 80 mins...
16      | t=11h12, task completed!
17
18  Fill-in task available! t=11h12: started 'Formalize the Utility function definition
    ↪  for CS110 Assignment' for 90 mins...
19      | t=12h42, task completed!
20
21  t=12h42: No eligible fill-in tasks available, waiting...
22
23  ...
24
25  t=12h59: No eligible fill-in tasks available, waiting...
26
27  t=13h00: started 'Have lunch at JJ's Poke Bowl and bike back to the library' for 75
    ↪  mins...
28      | t=14h15, task completed!
29
30  Fill-in task available! t=14h15: started 'Record CS110 Videos' for 75 mins...
31      | t=15h30, task completed!
32
33  t=15h30: No eligible fill-in tasks available, waiting...
34
35  ...
36
37  t=17h59: No eligible fill-in tasks available, waiting...
38
39  t=18h00: started 'CS111 Class' for 90 mins...
40      | t=19h30, task completed!
41
42  t=19h30: started 'Work on written parts of CS110 Assignment' for 120 mins...
43      | t=21h30, task completed!
44
45  t=21h30: No eligible fill-in tasks available, waiting...
46
47  ...
48
49  t=21h59: No eligible fill-in tasks available, waiting...
50
51  t=22h00: started 'Bike back to 9Floor after the library closes at 22h' for 12
    ↪  mins...
52      | t=22h12, task completed!
53
54  Completed all planned tasks in 14h12min!
```

4 Test-Driving the Scheduler

4.1 My tasks

```
    tasks = [
Task(id=0, description='Get up at 9:00 AM',
     duration=30, dependencies=[]),
Task(id=1, description='Get ready for the day (organize backpack)',
     duration=15, dependencies=[0]),
Task(id=2, description='Bike to the Main NTU Library',
     duration=15, dependencies=[0, 1]),
Task(id=3, description='Do Pre-Class Work for CS111',
     duration=75, dependencies=[2]),
Task(id=4, description='Formalize the utility function definition for CS110
↪  Assignment',
     duration=60, dependencies=[2]),
Task(id=5, description='Prepare video scripts and whiteboards for CS110 Assignment',
     duration=75, dependencies=[2]),
Task(id=6, description='Bike and have lunch at JJs Poke Bowl',
     duration=75, dependencies=[2], scheduled_time=14*60),
Task(id=7, description='CS111 Class',
     duration=90, dependencies=[3, 6], scheduled_time=18*60),
Task(id=8, description='Fix bugs in the additional fill-task methods for CS110
↪  Assignment',
     duration=90, dependencies=[5]),
Task(id=9, description='Work on LBA part of the upcoming CS111 Assignment',
     duration=60, dependencies=[3]),
Task(id=9, description='Bike back from JJs to NTU Library before CS111 class',
     duration=10, dependencies=[6]),
Task(id=10, description='Bike back to Res Hall after class finishes at 7:30pm',
     duration=12, dependencies=[6, 7, 8, 9], scheduled_time=1170) # 19.5*60
     ↪  expressed as an integer
]

task_scheduler = TaskScheduler(tasks)

task_scheduler.print_self()

start_scheduler_at = 9*60
task_scheduler.run_task_scheduler(start_scheduler_at)
task_scheduler.print_executed_task_descriptions()
```

4.2 My schedule

```
    Running a simple scheduler:

t=9h00: started 'Get up at 9:00 AM' for 30 mins...
```

```
    | t=9h30, task completed!

t=9h30: started 'Get ready for the day (organize backpack)' for 15 mins...
    | t=9h45, task completed!

t=9h45: started 'Bike to the Main NTU Library' for 15 mins...
    | t=10h00, task completed!

Fill-in task available! t=10h00: started 'Formalize the utility function definition
↪  for CS110 Assignment' for 60 mins...
    | t=11h00, task completed!

Fill-in task available! t=11h00: started 'Prepare video scripts and whiteboards for
↪  CS110 Assignment' for 75 mins...
    | t=12h15, task completed!

Fill-in task available! t=12h15: started 'Do Pre-Class Work for CS111' for 75
↪  mins...
    | t=13h30, task completed!

t=13h30: No eligible fill-in tasks available, waiting...

...

t=13h59: No eligible fill-in tasks available, waiting...

t=14h00: started 'Bike and have lunch at JJs Poke Bowl' for 75 mins...
    | t=15h15, task completed!

Fill-in task available! t=15h15: started 'Bike back from JJs to NTU Library before
↪  CS111 class' for 10 mins...
    | t=15h25, task completed!

Fill-in task available! t=15h25: started 'Work on LBA part of the upcoming CS111
↪  Assignment' for 60 mins...
    | t=16h25, task completed!

Fill-in task available! t=16h25: started 'Fix bugs in the additional fill-task
↪  methods for CS110 Assignment' for 90 mins...
    | t=17h55, task completed!

t=17h55: No eligible fill-in tasks available, waiting...

...

t=17h59: No eligible fill-in tasks available, waiting...

t=18h00: started 'CS111 Class' for 90 mins...
    | t=19h30, task completed!

t=19h30: started 'Bike back to Res Hall after class finishes at 7:30pm' for 12
↪  mins...
    | t=19h42, task completed!
```

```
50
51  Completed all planned tasks in 10h42min!
```

4.3 My pictures

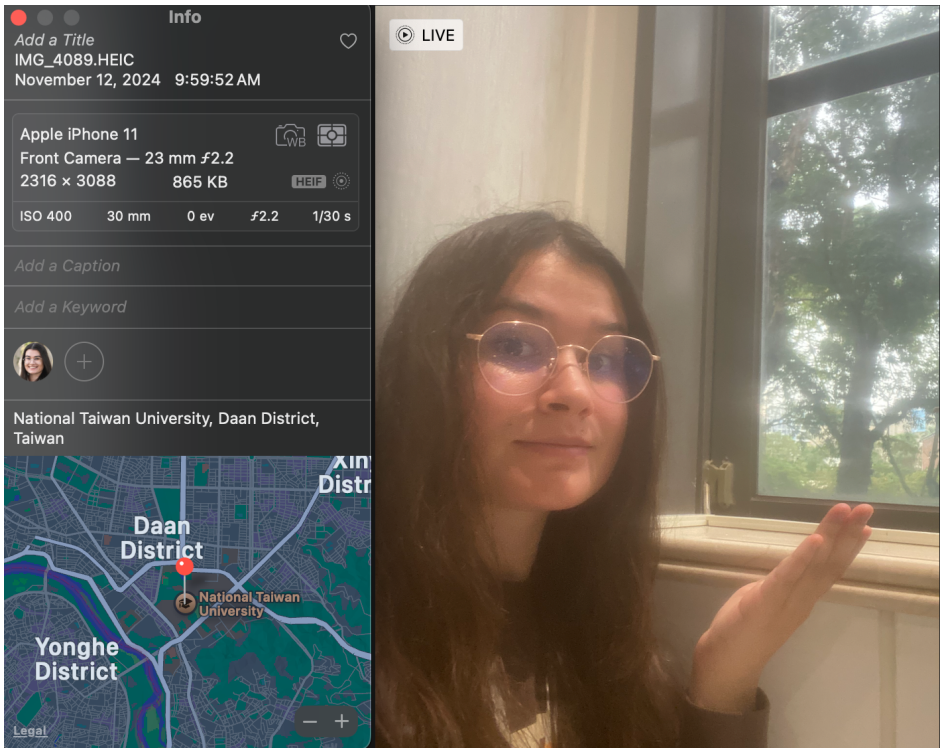


Figure 1: Arriving at the NTU Library in the morning, slightly earlier than scheduled

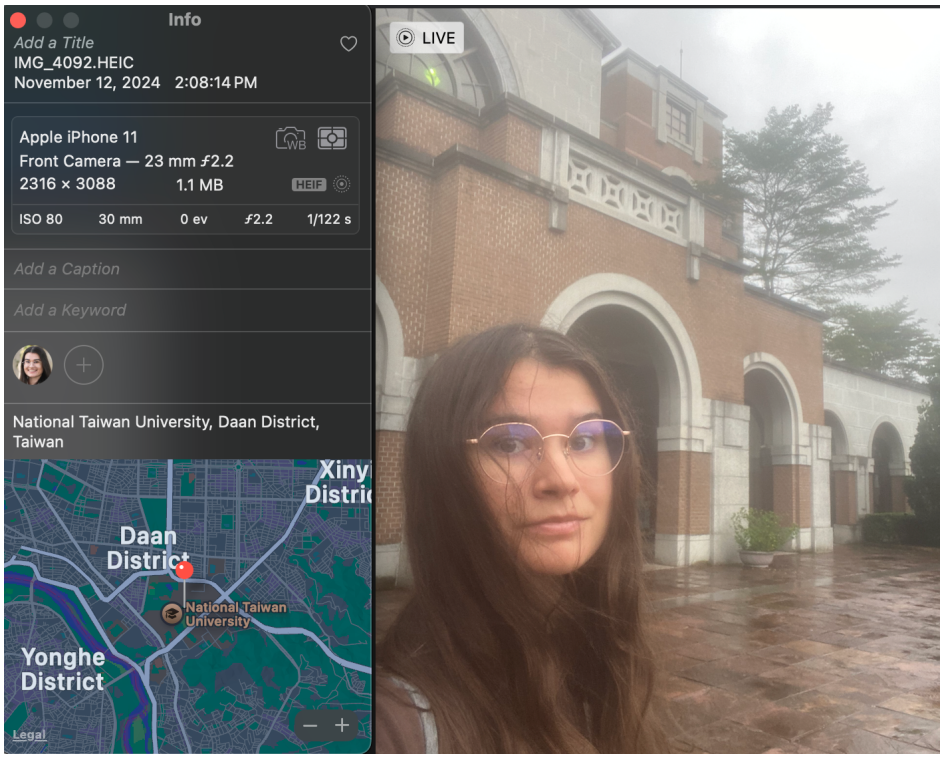


Figure 2: Leaving the NTU Library to bike to JJ's Poke

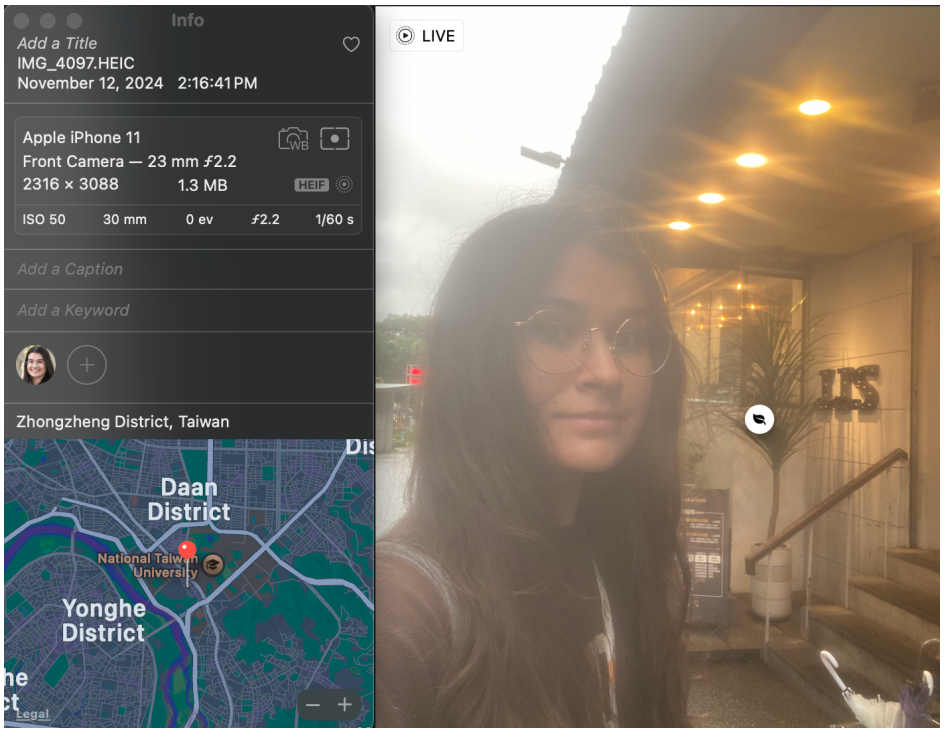


Figure 3: Arriving at JJs Poke for lunch

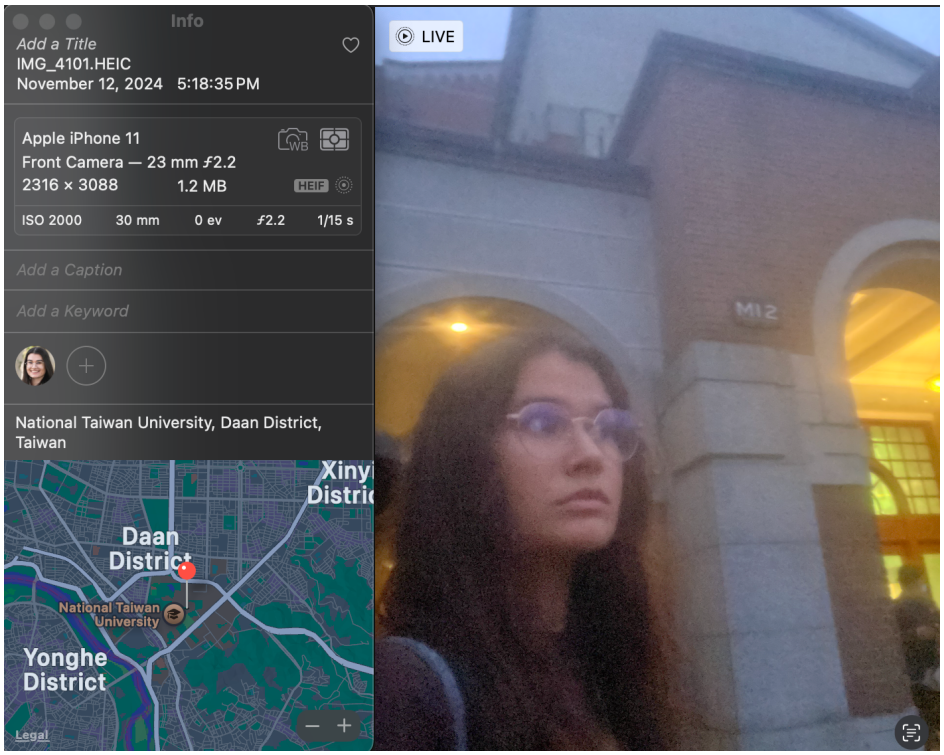


Figure 4: Arriving back at NTU Library with some additional time before my 6 pm CS111 Class. My schedule recommended I biked first to NTU since this would take less time, but I decided, for convenience, to switch the order of the tasks related to the CS110 assignment and perform them at JJs Poke before returning to NTU for my class (a potential idea for scheduler improvement)

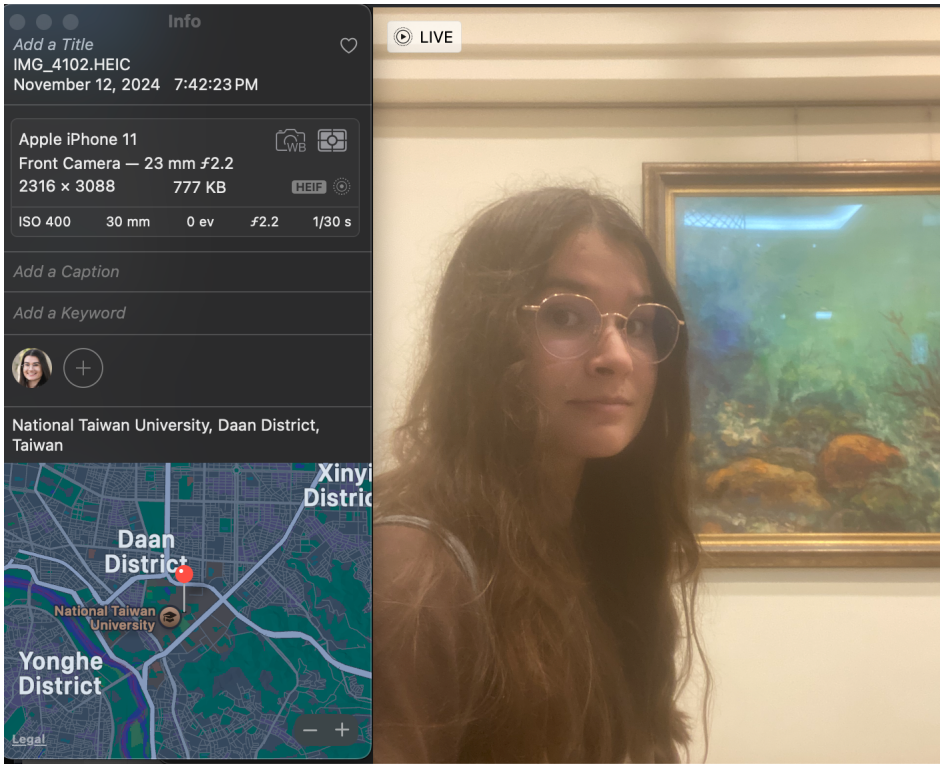


Figure 5: Leaving the NTU Library, for which I was 12 minutes late since I spontaneously decided to stay back for Office Hours after class

5 Analysis of the Scheduler

A) Overall, based on my experience, following the scheduler was useful, but there is room for improvement. Namely, the schedule was useful for following the tasks sorted by priority during time slots where I did not have a scheduled task, and all tasks were dependent on the same task. For example, the tasks of "Do Pre-Class Work for CS111", "Formalize the utility function definition for CS110 Assignment", and "Prepare video scripts and whiteboards for CS110 Assignment", all dependent on arriving at the NTU library, had no scheduled time. In this case, per my utility function, I would derive higher utility from completing tasks with lower duration. And indeed, I preferred to have concluded quicker tasks first.

However, as mentioned previously, I realized as I moved locations that I could have performed tasks in a different order than defined by the scheduler, and still ended up with the same outcome (i.e. arriving at the NTU Library at a similar time of 17:40). In this case, I decided to first fulfill the "Work on LBA part of the upcoming CS111 Assignment" and "Fix bugs in the additional fill-task methods for CS110 Assignments" tasks in the restaurant, which was quieter and it was convenient to use my computer inside. Because of this, the scheduler, if tailored to my needs, could also allow for changes in the order of the tasks, or also take a metric of place convenience into account. Namely, include a metric to decrease utility for any tasks identified for relocation.

Furthermore, the scheduler has an additional issue of not reorganizing tasks in case one task is delayed or missed. In my schedule, no significant delays, besides biking back to the residence hall 12 minutes after the scheduled time, occurred. However, if my tasks involved taking public transportation before a scheduled task, then I should account for potential delays or missed trains or buses (i.e. the utility function could increase the priority score of transportation tasks in which scheduled tasks depend on, that is, if the id of a transportation task in in the list of dependencies of a task with a scheduled time)

B) To examine the general efficiency of the scheduler qualitatively, we can perform sample tests on a particular schedule scenario. Here, we will evaluate the following metrics:

- **Number of tasks completed in an average day**. In this scenario, let us set a constraint of a maximum of 12 hours of task execution. If there are any delayed tasks for the next day, we'll mark them with the status of delayed and they will not be executed.
- **Idle time**, which corresponds to the time when no tasks are allocated. This is directly related to the new attribute of scheduled time and measures our schedule's efficiency in terms of how well it fills in non-scheduled tasks before a scheduled task.

A schedule is considered efficient if it can maximize the number of tasks executed and minimize the idle time between tasks with and without a scheduled time. We can analyze if a schedule is efficient by comparing a sample optimized schedule by hand against the schedule developed by our scheduler:

Table 1: Optimized Task Schedule by Hand

Task Description	Start Time	End Time
Get up and have breakfast	9:00 AM	9:30 AM
Do Pre-Class Work for CS111	9:30 AM	11:30 AM
Idle Time	11:30 AM	12:00 PM
Meeting for AI Sustainability Lab	12:00 PM	1:15 PM
Daily Exercise	1:15 PM	2:00 PM
Online Discussion with CCP Team	2:00 PM	3:00 PM
Revise last CS113 Content	3:00 PM	4:10 PM
Idle Time	4:10 PM	6:00 PM
CS111 Class	6:00 PM	7:30 PM
Idle Time	7:30 PM	8:00 PM
Bake Cookies with Friends	8:00 PM	10:00 PM
Skill Builder for CS113	Delayed	Delayed
Planning for the Rest of the Week	Delayed	Delayed

Total number of tasks completed: 8. Total idle time: 2h20 min

Table 2: Schedule Developed by the Task Scheduler

Task Description	Start Time	End Time
Get up and have breakfast	9:00 AM	9:30 AM
Meeting for AI Sustainability Lab	9:30 AM	10:45 AM
Daily Exercise	10:45 AM	11:30 AM
Do Pre-Class Work for CS111	11:30 PM	1:30 PM
Idle Time	1:30 PM	2:00 PM
Online Discussion with CCP Team	2:00 PM	3:00 PM
Revise last CS113 Content	3:00 PM	4:10 PM
Idle Time	4:10 PM	6:00 PM
CS111 Class	6:00 PM	7:30 PM
Idle Time	7:30 PM	8:00 PM
Bake Cookies with Friends	8:00 PM	10:00 PM
Skill Builder for CS113	Delayed	Delayed
Planning for the Rest of the Week	Delayed	Delayed

Total number of tasks completed: 8. Total idle time: Total idle time: 2h20 min.

Based on the example above, considering the dependencies, durations, and scheduled times of the input tasks, we can conclude the current scheduler meets both optimized metrics, maximizing the number of tasks completed and minimizing the idle time. Thus, it produces an efficient schedule for a collection of dependent, independent, scheduled, and non-scheduled tasks. Nevertheless, it is also important to point out that the original scheduler changed the scheduled time of the AI Sustainability Lab Meeting, which would not be an ideal behavior, considering that rescheduling a group task could be detrimental not only to the user's utility but also to its team's utility.

C) In order to measure the algorithmic efficiency experimentally we have to include a few abstractions to measure how the algorithm scales. Namely, we have to remove the constraint of the 12 hours of daytime we defined before and also assume that our scheduler can accommodate a large number of tasks which is not practically tractable (e.g. over 30 tasks) unless we schedule it for multiple days. We will also consider the tasks are fully independent of each other. The metric used to measure the scheduler's efficiency is time, in seconds, measured by running the task scheduler using the time library. In these experiments, we are computing the total time spent on the operations of the priority queue and the Task scheduler itself.

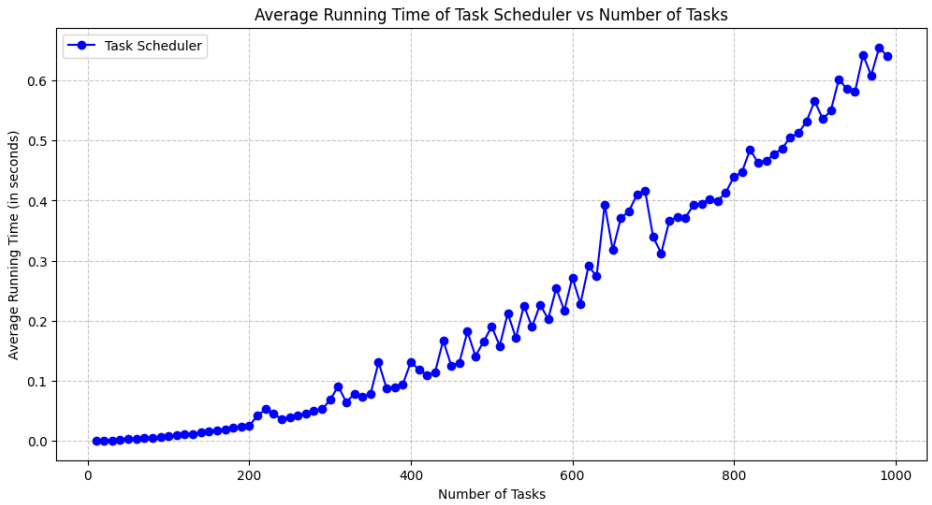
In this scenario, we will experimentally determine the asymptotic running time using a log-log plot, on increasing input sizes. A plot on a logarithmic scale is preferred in this case because the slope of the fitted line in a log-log plot directly indicates the growth rate of the dependent variable relative to the independent variable. For instance, if the slope is 1, it indicates a linear relationship $O(n)$; if it's greater than 1, it suggests super-linear growth (e.g. $O(n^2)$, $O(n \log n)$), while a slope less than 1 indicates sub-linear growth (e.g. $O(n \log n)$).

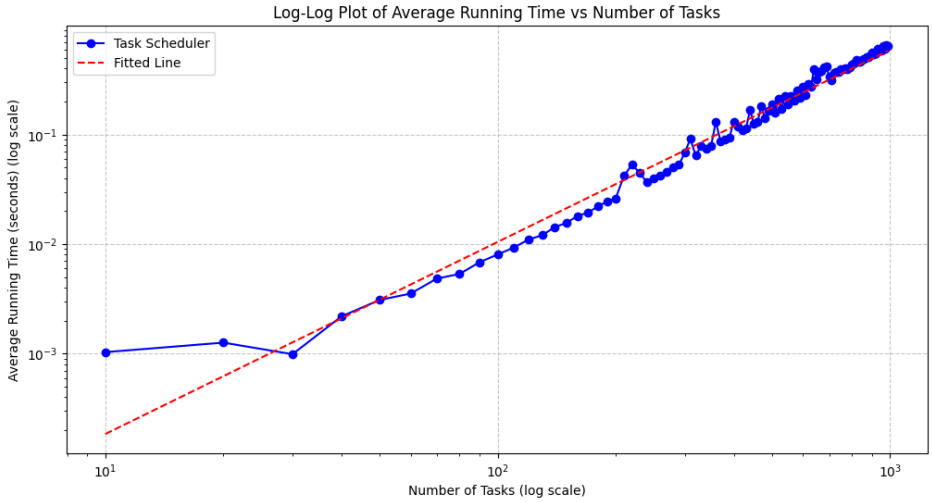
We can assume the dominant term of the running time is in the form:

$$T(n) = cn^k$$

Where c is the y-intercept, since the line intersects the vertical axis at $n = 10^0$, or $T(1) = c$, and k is the slope of the line.

Our experiments produced the following plots:





Our calculated slope is $k = 1.7532$, which indicates a super linear growth which is is in between $O(n)$ and $O(n^2)$, which we can roughly estimate as a $O(n \log n)$ growth, however closely bordering $O(n^2)$.

We can compare this against the theoretical analysis of our code by calculating the complexity of the following operations within the Task Scheduler:

1. **Adding tasks to the priority queue method:** We iterate through all the tasks to check for those that are ready to be added to the queue (i.e. have no dependencies). For each ready task: we update its status and priority and then push it into the Max Heap. In the heap push operation, adding a new element has a complexity of $O(1)$, and reorganizing the elements to meet the heap property during the heapify operation has a complexity of $O(h)$, where h is the current height of the heap, which is equal to $log(n)$, where n is the number of nodes. Our worst-case scenario is achieved when all tasks in the list of tasks have no dependencies, therefore, the worst-case complexity for our algorithm is $O(n \log n)$.

2. **Removing dependencies of a task:** here, we are iterating through all tasks to check is a completed task's ID is in the dependency list. If so, we remove the ID. Our worst-overall complexity is, therefore, of $O(n \cdot d)$, where d is the average number of dependencies per task. However for dense dependency schedules where (d = n), for each task, we will check dependencies of all other tasks, leading to a worst time complexity of $O(n^2)$

3. **Checking unscheduled tasks:** we iterate through all tasks to check if there are any which are still left to the scheduled, which also scales as $O(n)$

4. **Deleting the top node of the queue:** we remove the maximum value from the heap, move the last element to the root, and rebalance the heap using heapify. Replacing the root takes $O(1)$ while heapify takes $O(\log n)$. For n tasks, this scales up to $O(n \log n)$.

5. **Deleting an arbitrary node from the queue:** we remove an arbitrary index i from the heap, replace it with the last element and then rebalance the heap using heapify. Replacing the element while iterating through a heap of n nodes (or tasks) takes $O(n)$ (considering a worst case where the index of the selected element is the last) and heapify scales to $O(\log n)$,

which leads to a combined complexity of $O(n + \log n)$ that scales to $O(\log n)$ per fill-in. If there are m fill-in tasks executed in total, the complexity would scale as $O(m \cdot n)$, where n is the number of tasks in the heap during each fill-in. When there are many fill-in tasks ($m \approx n$), the worst-case complexity becomes $O(n^2)$.

Our overall time complexity is defined by the dominant term, among the addition of all the aforementioned complexities.

1. If there are few dependencies or few fill-in tasks:
   (a) Our overall complexity will be of $O(n \log n)$
2. If dependencies or the number of fill-in tasks are dense ($d \approx n$):
   (a) Our overall complexity will be of $O(n^2)$

Our experiments consider a scenario where the tasks are completely independent and where there are no scheduled tasks. Therefore, our theoretical results most roughly match with our experimental results mentioned in the figures above (although it is still important to point out our experimental results lean more towards a $O(n^2)$, which would have a slope of $k = 2$).

6 How can the scheduler be improved?

Based on the previous critiques, my current version still does not allow for the user to reschedule a task in case they miss the timing of a particular task. Further, the scheduler only changes the status of the task as delayed if it doesn't meet the 12-hour constraint. An improved version of the scheduler would already allocate the tasks for the following day, allowing for multi-day schedules.

Proposed enhancements:

1. **Dynamic Rescheduling Capability:**
    - Data Structure: Implement a hash map (dictionary) to store tasks by their IDs for quick access. This will allow users to easily locate and update tasks if they miss the timing.
    - Usage: When a task is delayed, instead of just marking it as such, the system can automatically propose a new scheduled time based on user-defined rules (e.g. next available slot).
2. **Multi-Day Scheduling:**
    - Data Structure: Use multiple priority queues to manage tasks across multiple days. Each day can have its own priority queue, allowing for efficient retrieval of tasks based on their urgency and scheduled times.
    - Usage: When tasks are delayed, they can be pushed into the next day's priority queue based on their priority score and dependencies. This ensures that tasks are not lost but are instead rescheduled effectively.
3. **Dependency Optimization:**
    - Data Structure: Implement a graph structure to manage task dependencies more effectively. Each task can point to its dependencies, allowing for easier traversal and checking of dependent tasks.
    - Usage: As mentioned in the complexity discussion above, our current traversal of dependencies is costly, and could eventually lead to a performance bottleneck for a significantly large number of tasks. Using a graph structure instead of a list of dependencies could be an appropriate optimization.

Word count: 293

7 Appendices

7.1 Part I: Reflection

In section **Session 13 - [7.2] Heaps and priority queues**, during the breakout I realized one key aspect of this assignment would be to define how Task objects are compared clearly. That is, I should define which attribute of the object would serve as a comparison key in the priority queue. I realized this could be achieved in the breakout by altering the lt() method. something that I wrote about in my reflection poll and implemented in this assignment. At the time, I selected the attribute of duration as a comparison key; in this assignment, I selected the priority score as an attribute for comparison.

7.2 Part II: LO and HC Applications

- *#professionalism:* The assignment has included all parts, including videos, explanations, code, and visualizations. I have included my Appendix both as a combined PDF and an .ipynb file in the .zip format. Completed the AI statement. (34 words)
- *#cs110-AlgoDataStruct:* In my video, I explained the priority queue structure implemented as a MaxHeap, comparing it to a list. I implemented its main operations through the MaxHeap class. In the last section, described additional data structures (e.g. hash maps and graphs) to be included in an improved version of the scheduler. (50 words)
- *#cs110-CodeReadability:* Throughout the assignment, I wrote code that uses appropriate function and variable naming, comments, and docstrings. Added inline comments to clarify details of the implementation. Included test cases using assert statements and implemented a wide variety of printing formats to improve the readability of several operations in my scheduler. (49 words)
- *#cs110-ComplexityAnalysis:* In my video, I referred to the time complexity of priority queues (implemented through MaxHeap) operations compared to lists. Later in the analysis, I provided a thorough analysis of the overall task scheduler combined with the priority queue operations by breaking down different operations. (44 words)
- *#cs110-ComputationalCritique:* I provided a critique of the practical advantages and disadvantages of my scheduler through my location-based experiment. Later in the analysis section, I provided a thorough analysis, with examples and experiments, which evaluate efficiency metrics in terms of both the schedule and scheduler (as a program). (46 words)
- *#cs110-PythonProgramming:* Ensured my code was functional and robust by including the described number of test cases, especially while developing a consistent schedule regardless of the input order. With the experiments, I made sure that the visualizations were plotted for a broad range of inputs and included titles, axis descriptions, and legends. (50 words)

Additional HC *#breakitdown:* This assignment was particularly complex and required a wide variety of elements to be commented on in enough depth. To fully conclude it, it was required to group tasks according to each subsection of the assignment, from creating a table to filming a video

to providing experimental results. Some sections, such as the location-based implementation, were also sequential, which also required additional organization and completion of previous parts. (68 words)

7.3 Part III: Python Code

Included as a merged PDF and through an additional .ipynb file.

8 AI Statement

I used Grammarly AI to support my writing and prevent potential typos and grammatical mistakes.

CS110_LBA_Assignment

November 17, 2024

1 Implementation of the Task Class

[1]:
```
class Task:
    """
    Represents a task with its details and current status.

    Attributes:
        id (int): Unique identifier for the task.
        description (str): A brief description of the task.
        duration (int): Duration of the task in minutes.
        dependencies (list): List of task IDs that must be completed before␣
↪this task.
        scheduled_time (int, optional): The specific time (in minutes from the␣
↪start of the day)
                                        when the task is scheduled to start.␣
↪Default is None.
        status (str): Current status of the task. Possible values:
                      - 'N' (NOT_STARTED): The task has not been started.
                      - 'I' (IN_PRIORITY_QUEUE): The task is in the priority␣
↪queue.
                      - 'C' (COMPLETED): The task has been completed.
                      - 'D' (DELAYED): The task has been delayed.
        priority_score (int): A score representing the urgency or importance of␣
↪the task.
                              Higher values indicate higher priority.
    """

    # Status constants for clarity and consistency
    NOT_STARTED = 'N'
    IN_PRIORITY_QUEUE = 'I'
    COMPLETED = 'C'

    def __init__(self, id, description, duration, dependencies,␣
↪scheduled_time=None, status='N', priority_score=0):
        """
        Initializes a Task instance.
```

```
        Args:
            id (int): Unique identifier for the task.
            description (str): A brief description of the task.
            duration (int): Duration of the task in minutes.
            dependencies (list): List of task IDs that must be completed before␣
↪this task.
            scheduled_time (int, optional): The specific time (in minutes from␣
↪the start of the day)
                                            when the task is scheduled to␣
↪start. Default is None.
            status (str): Initial status of the task. Default is 'N'␣
↪(NOT_STARTED).
            priority_score (int): Initial priority score of the task. Default␣
↪is 0.
        """
        self.id = id
        self.description = description
        self.duration = duration
        self.dependencies = dependencies
        self.scheduled_time = scheduled_time
        self.status = status
        self.priority_score = priority_score

    def __str__(self):
        """
        Returns a human-readable string representation of the Task.

        Returns:
            str: A string describing the task, including its ID, description,
                 duration, status, and priority score.
        """
        return (
            f"Task {self.id}: {self.description}\n"
            f"\tDuration: {self.duration} minutes\n"
            f"\tStatus: {self.status}\n"
            f"\tPriority Score = {self.priority_score}\n"
        )

    def compute_priority(self, current_time):
        """
        Computes and updates the task's priority score based on its attributes␣
↪and the current time.

        Priority is calculated using the following factors:
        - Scheduled Time: Tasks closer to their scheduled time have higher␣
↪priority.
```

```
        - Dependencies: Tasks with many dependencies have slightly reduced priority.
        - Duration: Shorter tasks may be prioritized for scheduling efficiency.

        Args:
            current_time (int): The current time (in minutes from the start of the day).
        """
        # Determine time priority based on the scheduled time of the task
        time_priority = 0 # Default
        if self.scheduled_time is not None:  # Check if the task has a predefined scheduled time
            if self.scheduled_time == current_time:
                time_priority = 100  # Assign maximum priority if the task is scheduled for the current time
            elif self.scheduled_time < current_time:
                time_priority = 0  # Task loses priority if its scheduled time has already passed
            else:
                # Increase priority as the task's scheduled time approaches the current time
                time_priority = (self.scheduled_time - current_time) * 10

        # Penalize tasks with many dependencies
        dependency_score = len(self.dependencies)

        # Total priority score calculation
        total_priority = (
            time_priority -  # Directly use time_priority, which can be positive
            dependency_score + # Dependence score contributes negatively to the utility
            abs(100 - self.duration)  # Longer tasks have lower scores
        )

        self.priority_score = total_priority

    def __lt__(self, other):
        """
        Compares tasks for sorting, based on their priority score.

        Args:
            other (Task): Another Task instance to compare with.

        Returns:
            bool: True if this task has a lower priority score than the other task.
```

```
        """
        return self.priority_score < other.priority_score

    def to_dict(self):
        """
        Converts the Task instance into a dictionary representation.

        Returns:
            dict: A dictionary containing the task's key attributes:
                  - id (int): Task ID.
                  - description (str): Task description.
                  - duration (int): Task duration in minutes.
                  - dependency (list): List of task dependencies.
        """
        return {
            "id": self.id,
            "description": self.description,
            "duration": self.duration,
            "dependencies": self.dependencies,
        }
```

2 Task Table

[ ]:
```
# Text input for 11 tasks:
Write grocery checklist, 50, 0
Do groceries at PX Mart, 50, Write grocery checklist
Charge Computer before class, 50, 0
Do Pre-Class Work for CS110, 100, Charge Computer before class
Vacuum the floor, 20, 0
Withdraw cash from 7-11 ATM, 15, 0
CS110 Class, 90, Do Pre-Class Work for CS110
Take NTU id card from the room, 5, 0
Go to the NTU library through YouBike, 20, Take NTU id card
Take the MRT train to the City Science Lab @ Taipei from MIT, 40, Take NTU id␣
↪card
Bike back to Halfway Cafe in the Daan District to study until late, 30, Take␣
↪NTU id card
```

[ ]:
```
import pandas as pd
import numpy as np

def create_table():
    """
    Creates a table of tasks based on user input and stores them in a Pandas␣
↪DataFrame.
```

```
    The user is prompted to enter the number of tasks and details for each␣
↪task, including:
    description, duration (in minutes), and dependencies. The dependencies␣
↪should be
    specified as task descriptions, with '0' indicating no dependencies.
    """

    # Prompt user for the number of tasks to add
    n_tasks = int(input("How many tasks will you add to the table? "))

    id = 0
    task_list = []
    task_df = pd.DataFrame(columns=['id', 'description', 'duration',␣
↪'dependencies'])

    # Loop to collect task details from user input
    for i in range(n_tasks):
        description, duration, dependency = input(
            "Type in comma separated values: description, duration (in␣
↪minutes), "
            "and the name of the task which needs to be completed before that.␣
↪If there are no dependencies, type 0: "
        ).split(', ')
        print(description, duration, dependency)

        # Create a new Task instance with provided details
        new_task = Task(
            id=id,
            description= description.strip(),
            duration= int(duration.strip()),
            dependencies= description.strip() if dependency != '0' else 'No␣
↪Dependency'  # Convert dependency to list or empty if '0'
        )

        # Convert the new task to a dictionary format for DataFrame creation
        new_row = new_task.to_dict()

        # Append new row to existing DataFrame
        task_df = pd.concat([task_df, pd.DataFrame([new_row])],␣
↪ignore_index=True)

        task_list.append(new_task)  # Add the new Task instance to the list

        id += 1  # Increment task ID for next task

    return task_df  # Print final DataFrame containing all tasks
```

```
# Call the function to execute it
create_table()
```

```
How many tasks will you add to the table? 11
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: Write grocery checklist, 50, 0
Write grocery checklist 50 0
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: Do groceries at PX Mart, 50, Write grocery checklist
Do groceries at PX Mart 50 Write grocery checklist
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: Charge Computer before class, 50, 0
Charge Computer before class 50 0
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: Do Pre-Class Work for CS110, 100, Charge Computer before
class
Do Pre-Class Work for CS110 100 Charge Computer before class
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: Vacuum the floor, 20, 0
Vacuum the floor 20 0
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: Withdraw cash from 7-11 ATM, 15, 0
Withdraw cash from 7-11 ATM 15 0
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: CS110 Class, 90, Do Pre-Class Work for CS110
CS110 Class 90 Do Pre-Class Work for CS110
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: Take NTU id card from the room, 5, 0
Take NTU id card from the room 5 0
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: Go to the NTU library through YouBike, 20, Take NTU id
card
Go to the NTU library through YouBike 20 Take NTU id card
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: Take the MRT train to the City Science Lab @ Taipei from
MIT, 40, Take NTU id card
Take the MRT train to the City Science Lab @ Taipei from MIT 40 Take NTU id card
```

```
Type in comma separated values: description, duration (in minutes), and the name
of the task which needs to be completed before that. If there are no
dependencies, type 0: Bike back to Halfway Cafe in the Daan District to study
until late, 30, Take NTU id card
Bike back to Halfway Cafe in the Daan District to study until late 30 Take NTU
id card
```

[ ]:
```
    id                                        description duration  \
0    0                          Write grocery checklist       50
1    1                          Do groceries at PX Mart       50
2    2                     Charge Computer before class       50
3    3                      Do Pre-Class Work for CS110      100
4    4                                  Vacuum the floor       20
5    5                      Withdraw cash from 7-11 ATM       15
6    6                                    CS110 Class       90
7    7                   Take NTU id card from the room        5
8    8            Go to the NTU library through YouBike       20
9    9  Take the MRT train to the City Science Lab @ T…       40
10  10  Bike back to Halfway Cafe in the Daan District…       30

                                        dependencies
0                                       No Dependency
1                           Do groceries at PX Mart
2                                       No Dependency
3                     Do Pre-Class Work for CS110
4                                       No Dependency
5                                       No Dependency
6                                         CS110 Class
7                                       No Dependency
8             Go to the NTU library through YouBike
9   Take the MRT train to the City Science Lab @ T…
10  Bike back to Halfway Cafe in the Daan District…
```

3 Implementation of the MaxHeap

Most of the code was extracted from Session Session 13 - [7.2] Heaps and priority queues (Levay, 2024)

[2]:
```python
class MaxHeapq:
    """
    A class that implements properties and methods
    that support a max priority queue data structure.

    Attributes
    ----------
    heap : list
        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):
        """
        Initializes an empty MaxHeapq instance.

        Parameters
        ----------
        None
        """
        self.heap = []
        self.heap_size = 0

    def is_max_heap(self):
        """
        Checks if the current list satisfies the max heap property.

        Returns
        -------
        bool
            True if the list is a max heap, False otherwise.
        """
        a = self.heap
        n = self.heap_size

        # Check each node to ensure it is greater than its children
        for i in range(n):
            left_child = self.left(i)
            right_child = self.right(i)

            # Check if left child exists and is greater than parent
            if left_child < n and a[i] < a[left_child]:
                return False

            # Check if right child exists and is greater than parent
            if right_child < n and a[i] < a[right_child]:
                return False
        return True


    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 the 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 the 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 the child node.

        Returns
        -------
        int
            Index of the parent node.
        """
        return (i - 1) // 2

    def heappush(self, key):
```

```
        """
        Inserts a key into the priority queue.

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

        Returns
        -------
        None
            This method modifies the heap in place.
        """

        # Append a placeholder for new key and then increase its value to␣
↪correct position
        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.

        Raises
        ------
        ValueError
            If new key is smaller than current key.

        Returns
        -------
        None
            This method modifies the heap in place.
        """

        if key.priority_score < 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)] < self.heap[i]:
```

```
            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 by ensuring that subtree rooted at index i satisfies max heap property.

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

        Returns
        -------
        None
            This method modifies the heap in place.
        """
        l = self.left(i)
        r = self.right(i)
        largest = i

        # Check if left child exists and is greater than current largest
        if l < self.heap_size and self.heap[l] > self.heap[largest]:
            largest = l

        # Check if right child exists and is greater than current largest
        if r < self.heap_size and self.heap[r] > self.heap[largest]:
            largest = r

        # If largest is not current index, swap and continue heapifying
        if largest != i:
            self.heap[i], self.heap[largest] = self.heap[largest], self.heap[i]
            self.heapify(largest)

    # Novel method
    def anypop(self, i):
        """
        Returns an arbitrary element from the max priority queue and removes it from it.

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

```
            The index of an element to pop from the heap.

        Raises
        ------
        ValueError
            If there are no keys in priority queue (heap underflow).

        Returns
        -------
        int
            The value extracted from the heap at index i.
        """
        if self.heap_size < 1:
            raise ValueError('Heap underflow: There are no keys in priority␣
↪queue.')

        any_value = self.heap[i]  # Store value to return later

        # Replace root with last element, pop last element, decrease size, then␣
↪re-heapify
        self.heap[i] = self.heap[-1]
        self.heap.pop()
        self.heap_size -= 1

        # Re-heapify from index i to maintain max heap property
        self.heapify(i)

        return any_value

    def heappop(self):
        """
        Returns and removes the largest key in the max priority queue.

        Raises
        ------
        ValueError
            If there are no keys in priority queue (heap underflow).

        Returns
        -------
        int
            The maximum value extracted from the heap (the root).
        """

        if self.heap_size < 1:
            raise ValueError('Heap underflow: There are no keys in priority␣
↪queue.')
```

```
        max_value = self.heap[0]

        self.heap[0] = self.heap[-1]
        self.heap.pop()
        self.heap_size -= 1

        self.heapify(0)

        return max_value
```

4 Implementation of the Task Scheduler Class

[3]:
```
class TaskScheduler:
    """
    A simple daily task scheduler using a priority queue.

    Attributes:
        tasks (list): List of tasks to be managed by the scheduler.
        priority_queue (MaxHeapq): Priority queue for managing task execution␣
↪order.
    """

    NOT_STARTED = 'N'  # Task has not been started
    IN_PRIORITY_QUEUE = 'I'  # Task is in the priority queue
    COMPLETED = 'C'  # Task has been completed

    def __init__(self, tasks):
        """
        Initializes an instance of TaskScheduler.

        Args:
            tasks (list): List of Task instances to schedule.
        """
        self.tasks = tasks  # Store the list of tasks
        self.priority_queue = MaxHeapq()  # Initialize the priority queue
        self.executed_task_descriptions = []  # Stores descriptions of executed␣
↪tasks

    def print_self(self):
        """Prints all tasks added to the scheduler with their details."""
        print("Tasks added to the simple scheduler:")
        print("---------------------------------------")

        for t in self.tasks:
            print(f"  '{t.description}', duration = {t.duration} mins.")
```

```
            if len(t.dependencies) > 0:
                print(f"\t   This task depends on others!")
                print(f"These are the dependencies: {t.dependencies}")

    def print_priority_queue(self):
        """Prints the current state of the priority queue."""
        print('This is my priority queue: ')

        for order, task in enumerate(self.priority_queue.heap):
            print(f'{order}) {task}')

    def remove_dependency(self, id):
        """
        Removes a specified task ID from the dependencies of other tasks.

        Args:
            id (int): The ID of the task whose dependencies should be removed.
        """
        for t in self.tasks:
            if t.id != id and id in t.dependencies:
                t.dependencies.remove(id)

    def get_tasks_ready(self, current_time):
        """
        Updates the status and computes priorities for tasks that are ready to execute.

        Args:
            current_time (int): The current time used to check readiness.
        """

        for task in self.tasks:
            # If the task has no dependencies and is not yet in the queue
            if task.status == self.NOT_STARTED and not task.dependencies:
                # Update status of the task
                task.status = self.IN_PRIORITY_QUEUE
                task.compute_priority(current_time)  # Compute priority based on current time
                # Push task into the priority queue
                self.priority_queue.heappush(task)

    def check_unscheduled_tasks(self):
        """Checks if there are any unscheduled tasks remaining."""

        for task in self.tasks:
            if task.status == self.NOT_STARTED:  # If any task is not started, return True
```

```
                return True

        return False  # No unscheduled tasks

    def format_time(self, time):
        """Formats time from minutes into hours and minutes string format."""

        return f"{time // 60}h{time % 60:02d}"

    def validate_heap(self):
        """Validates that the priority queue maintains its max-heap property."""
        assert self.priority_queue.is_max_heap(), "Heap property violated!"

    def run_task_scheduler(self, starting_time):
        """
        Executes the scheduler starting from a specified time.

        Args:
            starting_time (int): The initial time from which scheduling begins.
        """
        day_ended = False  # Flag to indicate if the day has ended
        current_time = starting_time  # Initialize current time
        #print("Running a simple scheduler:\n")

        while self.check_unscheduled_tasks() or self.priority_queue.heap:
            self.get_tasks_ready(current_time)  # Prepare tasks ready to execute
            # NOTE: Make sure to comment the print values when running the␣
↪quantitative experiment and avoid memory overflow
            #self.print_priority_queue()

            if self.priority_queue.heap_size > 0:  # If there are tasks in the␣
↪queue
                current_task = self.priority_queue.heappop()  # Get highest␣
↪priority task
                self.validate_heap()  # Validate heap property

                scheduled_time = current_task.scheduled_time

                if scheduled_time is not None and scheduled_time > current_time:
                    wait_until = scheduled_time

                    while current_time < wait_until:  # Wait until scheduled␣
↪time
                        fill_in_executed = False

                        for current_index in range(self.priority_queue.
↪heap_size):
```

```
                            peek_fill_in_task = self.priority_queue.
↪heap[current_index]

                            if peek_fill_in_task.scheduled_time is None or␣
↪peek_fill_in_task.scheduled_time < scheduled_time:
                                if current_time + peek_fill_in_task.duration <=␣
↪wait_until:
                                    fill_in_task = self.priority_queue.
↪anypop(current_index)
                                    self.validate_heap()

                                    # NOTE: Make sure to comment the print␣
↪values when running the quantitative experiment and avoid memory overflow
                                    #print(f" Fill-in task available! t={self.
↪format_time(current_time)}: started '{fill_in_task.description}' for␣
↪{fill_in_task.duration} mins...")
                                    current_time += fill_in_task.duration  #␣
↪Update current time after executing fill-in task
                                    #print(f"\t  t={self.
↪format_time(current_time)}, task completed!\n")
                                    self.remove_dependency(fill_in_task.id)  #␣
↪Remove completed dependencies
                                    fill_in_task.status = fill_in_task.
↪COMPLETED  # Mark as completed
                                    self.executed_task_descriptions.
↪append(fill_in_task.description)  # Log description
                                    fill_in_executed = True
                                    break

                        if not fill_in_executed:  # If no fill-in tasks were␣
↪executed, wait longer
                            #print(f"  t={self.format_time(current_time)}: No␣
↪eligible fill-in tasks available, waiting...\n")
                            current_time += 1

                # NOTE: Make sure to comment the print values when running the␣
↪quantitative experiment and avoid memory overflow
                #print(f" t={self.format_time(current_time)}: started␣
↪'{current_task.description}' for {current_task.duration} mins...")
                current_time += current_task.duration  # Update time after␣
↪executing main task
                #print(f"\t  t={self.format_time(current_time)}, task completed!
↪\n")
                self.remove_dependency(current_task.id)  # Remove completed␣
↪dependencies
```

```
                current_task.status = current_task.COMPLETED  # Mark as␣
↪completed
                self.executed_task_descriptions.append(current_task.
↪description)  # Store description

                total_time_so_far = current_time - starting_time

        total_time = current_time - starting_time
        #print(f"\n  Completed all planned tasks in {total_time //␣
↪60}h{total_time % 60:02d}min!")

    def print_executed_task_descriptions(self):
        """
        Prints the final descriptions of executed tasks as a list of strings.
        """
        print("\nFinal descriptions of executed tasks:")
        print("-------------------------------------")
        for i, description in enumerate(self.executed_task_descriptions,␣
↪start=1):
            print(f"{i}. {description}")
```

4.1 Test Case 1: Insert and Check Max Heap property

With all tasks independent and non-scheduled, priority is mainly computed by duration. Tasks with smaller duration are priorittized and should come first.

[79]:
```
heap = MaxHeapq()

# fully independent tasks
simple_tasks = [
    Task(id=0, description='Write grocery checklist', duration=50,␣
↪dependencies=[]),
    Task(id=1, description='Vacuum the floor', duration=20, dependencies=[]),
    Task(id=2, description='Withdraw cash from 7-11 ATM', duration=15,␣
↪dependencies=[]),
    Task(id=3, description='Do groceries at PX Mart', duration=50,␣
↪dependencies=[0]),
    Task(id=4, description='Charge Computer before class', duration=50,␣
↪dependencies=[]),
    Task(id=5, description='Do Pre-Class Work for CS110', duration=100,␣
↪dependencies=[4]),
]

# assume current time if 8am
current_time = 8*60

for task in simple_tasks:
```

```
  task.compute_priority(current_time)

# Insert all tasks into the heap
for task in simple_tasks:
    heap.heappush(task)

assert heap.is_max_heap(), "Heap does not maintain max-heap property after task␣
 ↪insertions."
assert heap.heap[0].description == 'Withdraw cash from 7-11 ATM', "Heappop did␣
 ↪not remove the correct task."
```

4.2 Test Case 2: Remove the Maximum Element to test heappop

[82]:
```
heap = MaxHeapq()

# fully independent tasks
simple_tasks = [
    Task(id=0, description='Vacuum the floor', duration=20, dependencies=[]),
    Task(id=1, description='Withdraw cash from 7-11 ATM', duration=15,␣
 ↪dependencies=[]),
    Task(id=2, description='Do groceries at PX Mart', duration=50,␣
 ↪dependencies=[]),
    Task(id=3, description='Charge Computer before class', duration=50,␣
 ↪dependencies=[]),
    Task(id=4, description='Do Pre-Class Work for CS110', duration=100,␣
 ↪dependencies=[]),
]

# assume current time is 8am
current_time = 8*60

for task in simple_tasks:
  task.compute_priority(current_time)

for task in simple_tasks:
    heap.heappush(task)

max_priority_task = heap.heappop()
print(max_priority_task)

# Verify the task with the highest priority (shortest duration) is removed
assert max_priority_task.description == 'Withdraw cash from 7-11 ATM', "Heappop␣
 ↪did not remove the correct task."
assert heap.is_max_heap()
```

```
Task 1: Withdraw cash from 7-11 ATM
        Duration: 15 minutes
```

```
Status: N
Priority Score = 85
```

4.3 Test Case 3: Remove Arbitrary Element to test anypop()

[81]:
```
heap = MaxHeapq()

# fully independent tasks
simple_tasks = [
    Task(id=0, description='Vacuum the floor', duration=20, dependencies=[]),
    Task(id=1, description='Withdraw cash from 7-11 ATM', duration=15,␣
 ↪dependencies=[]),
    Task(id=2, description='Do groceries at PX Mart', duration=50,␣
 ↪dependencies=[]),
    Task(id=3, description='Charge Computer before class', duration=50,␣
 ↪dependencies=[]),
    Task(id=4, description='Do Pre-Class Work for CS110', duration=100,␣
 ↪dependencies=[]),
]

# assume current time is 8am
current_time = 8*60

for task in simple_tasks:
  task.compute_priority(current_time)

for task in simple_tasks:
    heap.heappush(task)

removed_task = heap.anypop(2)  # Arbitrarily remove task at index 2 (Withdraw␣
 ↪cash from 7-11 ATM)

assert removed_task.description == 'Do groceries at PX Mart', "anypop did not␣
 ↪remove the correct task."
assert heap.is_max_heap(), "Heap does not maintain max-heap property after␣
 ↪anypop."
```

#Tests

4.4 Simple Sample Schedule

[87]:
```
simple_tasks = [
    Task(id=0, description='Write grocery checklist', duration=50,␣
 ↪dependencies=[]),
    Task(id=1, description='Vacuum the floor', duration=20, dependencies=[]),
```

```
    Task(id=2, description='Withdraw cash from 7-11 ATM', duration=15,␣
 ↪dependencies=[]),
    Task(id=3, description='Do groceries at PX Mart', duration=50,␣
 ↪dependencies=[0]),
    Task(id=4, description='Charge Computer before class', duration=50,␣
 ↪dependencies=[]),
    Task(id=5, description='Do Pre-Class Work for CS110', duration=100,␣
 ↪dependencies=[4]),
]
```

[88]:
```
simple_task_scheduler = TaskScheduler(simple_tasks)

simple_task_scheduler.print_self()

start_scheduler_at = 8*60
simple_task_scheduler.run_task_scheduler(start_scheduler_at)
```

```
Tasks added to the simple scheduler:
--------------------------------------
  'Write grocery checklist', duration = 50 mins.
  'Vacuum the floor', duration = 20 mins.
  'Withdraw cash from 7-11 ATM', duration = 15 mins.
  'Do groceries at PX Mart', duration = 50 mins.
           This task depends on others!
These are the dependencies: [0]
  'Charge Computer before class', duration = 50 mins.
  'Do Pre-Class Work for CS110', duration = 100 mins.
           This task depends on others!
These are the dependencies: [4]
Running a simple scheduler:

This is my priority queue:
0) Task 2: Withdraw cash from 7-11 ATM
        Duration: 15 minutes
        Status: I
        Priority Score = 85

1) Task 0: Write grocery checklist
        Duration: 50 minutes
        Status: I
        Priority Score = 50

2) Task 1: Vacuum the floor
        Duration: 20 minutes
        Status: I
        Priority Score = 80
```

```
3) Task 4: Charge Computer before class
        Duration: 50 minutes
        Status: I
        Priority Score = 50

 t=8h00: started 'Withdraw cash from 7-11 ATM' for 15 mins…
          t=8h15, task completed!

This is my priority queue:
0) Task 1: Vacuum the floor
        Duration: 20 minutes
        Status: I
        Priority Score = 80

1) Task 0: Write grocery checklist
        Duration: 50 minutes
        Status: I
        Priority Score = 50

2) Task 4: Charge Computer before class
        Duration: 50 minutes
        Status: I
        Priority Score = 50

 t=8h15: started 'Vacuum the floor' for 20 mins…
          t=8h35, task completed!

This is my priority queue:
0) Task 4: Charge Computer before class
        Duration: 50 minutes
        Status: I
        Priority Score = 50

1) Task 0: Write grocery checklist
        Duration: 50 minutes
        Status: I
        Priority Score = 50

 t=8h35: started 'Charge Computer before class' for 50 mins…
          t=9h25, task completed!

This is my priority queue:
0) Task 0: Write grocery checklist
        Duration: 50 minutes
        Status: I
        Priority Score = 50

1) Task 5: Do Pre-Class Work for CS110
```

```
        Duration: 100 minutes
        Status: I
        Priority Score = 0

 t=9h25: started 'Write grocery checklist' for 50 mins…
         t=10h15, task completed!

This is my priority queue:
0) Task 3: Do groceries at PX Mart
        Duration: 50 minutes
        Status: I
        Priority Score = 50

1) Task 5: Do Pre-Class Work for CS110
        Duration: 100 minutes
        Status: I
        Priority Score = 0

 t=10h15: started 'Do groceries at PX Mart' for 50 mins…
          t=11h05, task completed!

This is my priority queue:
0) Task 5: Do Pre-Class Work for CS110
        Duration: 100 minutes
        Status: I
        Priority Score = 0

 t=11h05: started 'Do Pre-Class Work for CS110' for 100 mins…
          t=12h45, task completed!


  Completed all planned tasks in 4h45min!
```

4.5 Verifiying Heap Property

[94]:
```
tasks = [
    Task(id=0, description='Get up at 8:00 AM',
         duration=30, dependencies=[]),
    Task(id=1, description='Walk to 7-11 and have breakfast',
         duration=20, dependencies=[0]),
    Task(id=2, description='Bike to the Main NTU Library',
         duration=12, dependencies=[0, 1]),
    Task(id=3, description='Do Pre-Class Work for CS111',
         duration=50, dependencies=[2]),
    Task(id=4, description='Formalize the Utility function definition for CS110␣
 ↪Assignment',
         duration=90, dependencies=[2]),
```

```
    Task(id=5, description='Prepare video scripts and whiteboards for CS110␣
 ↪Assignment',
         duration=80, dependencies=[2]),
    Task(id=6, description="Have lunch at JJ's Poke Bowl and bike back to the␣
 ↪library",
         duration=75, dependencies=[2], scheduled_time=13*60),
    Task(id=7, description='CS111 Class',
         duration=90, dependencies=[3], scheduled_time=18*60),
    Task(id=8, description='Record CS110 Videos',
         duration=75, dependencies=[4, 5]),
    Task(id=9, description='Work on written parts of CS110 Assignment',
         duration=120, dependencies=[8]),
    Task(id=10, description='Bike back to 9Floor after the library closes at␣
 ↪22h',
         duration=12, dependencies=[6, 7, 8, 9], scheduled_time=22*60)
    ]

task_scheduler = TaskScheduler(tasks)

task_scheduler.print_self()

start_scheduler_at = 8*60
task_scheduler.run_task_scheduler(start_scheduler_at)
```

```
Tasks added to the simple scheduler:
-------------------------------------
  'Get up at 8:00 AM', duration = 30 mins.
  'Walk to 7-11 and have breakfast', duration = 20 mins.
           This task depends on others!
These are the dependencies: [0]
  'Bike to the Main NTU Library', duration = 12 mins.
           This task depends on others!
These are the dependencies: [0, 1]
  'Do Pre-Class Work for CS111', duration = 50 mins.
           This task depends on others!
These are the dependencies: [2]
  'Formalize the Utility function definition for CS110 Assignment', duration =
90 mins.
           This task depends on others!
These are the dependencies: [2]
  'Prepare video scripts and whiteboards for CS110 Assignment', duration = 80
mins.
           This task depends on others!
These are the dependencies: [2]
  'Have lunch at JJ's Poke Bowl and bike back to the library', duration = 75
mins.
           This task depends on others!
```

```
These are the dependencies: [2]
  'CS111 Class', duration = 90 mins.
           This task depends on others!
These are the dependencies: [3]
  'Record CS110 Videos', duration = 75 mins.
           This task depends on others!
These are the dependencies: [4, 5]
  'Work on written parts of CS110 Assignment', duration = 120 mins.
           This task depends on others!
These are the dependencies: [8]
  'Bike back to 9Floor after the library closes at 22h', duration = 12 mins.
           This task depends on others!
These are the dependencies: [6, 7, 8, 9]
Running a simple scheduler:

 t=8h00: started 'Get up at 8:00 AM' for 30 mins…
          t=8h30, task completed!

 t=8h30: started 'Walk to 7-11 and have breakfast' for 20 mins…
          t=8h50, task completed!

 t=8h50: started 'Bike to the Main NTU Library' for 12 mins…
          t=9h02, task completed!

  Fill-in task available! t=9h02: started 'Do Pre-Class Work for CS111' for 50
mins…
          t=9h52, task completed!

  Fill-in task available! t=9h52: started 'Prepare video scripts and whiteboards
for CS110 Assignment' for 80 mins…
          t=11h12, task completed!

  Fill-in task available! t=11h12: started 'Formalize the Utility function
definition for CS110 Assignment' for 90 mins…
          t=12h42, task completed!

  t=12h42: No eligible fill-in tasks available, waiting…

  t=12h43: No eligible fill-in tasks available, waiting…

  t=12h44: No eligible fill-in tasks available, waiting…

  t=12h45: No eligible fill-in tasks available, waiting…

  t=12h46: No eligible fill-in tasks available, waiting…

  t=12h47: No eligible fill-in tasks available, waiting…
```

```
 t=12h48: No eligible fill-in tasks available, waiting…

 t=12h49: No eligible fill-in tasks available, waiting…

 t=12h50: No eligible fill-in tasks available, waiting…

 t=12h51: No eligible fill-in tasks available, waiting…

 t=12h52: No eligible fill-in tasks available, waiting…

 t=12h53: No eligible fill-in tasks available, waiting…

 t=12h54: No eligible fill-in tasks available, waiting…

 t=12h55: No eligible fill-in tasks available, waiting…

 t=12h56: No eligible fill-in tasks available, waiting…

 t=12h57: No eligible fill-in tasks available, waiting…

 t=12h58: No eligible fill-in tasks available, waiting…

 t=12h59: No eligible fill-in tasks available, waiting…

t=13h00: started 'Have lunch at JJ's Poke Bowl and bike back to the library'
for 75 mins…
         t=14h15, task completed!

 Fill-in task available! t=14h15: started 'Record CS110 Videos' for 75 mins…
         t=15h30, task completed!

 t=15h30: No eligible fill-in tasks available, waiting…

 t=15h31: No eligible fill-in tasks available, waiting…

 t=15h32: No eligible fill-in tasks available, waiting…

 t=15h33: No eligible fill-in tasks available, waiting…

 t=15h34: No eligible fill-in tasks available, waiting…

 t=15h35: No eligible fill-in tasks available, waiting…

 t=15h36: No eligible fill-in tasks available, waiting…

 t=15h37: No eligible fill-in tasks available, waiting…

 t=15h38: No eligible fill-in tasks available, waiting…
```

```
t=15h39: No eligible fill-in tasks available, waiting…
t=15h40: No eligible fill-in tasks available, waiting…
t=15h41: No eligible fill-in tasks available, waiting…
t=15h42: No eligible fill-in tasks available, waiting…
t=15h43: No eligible fill-in tasks available, waiting…
t=15h44: No eligible fill-in tasks available, waiting…
t=15h45: No eligible fill-in tasks available, waiting…
t=15h46: No eligible fill-in tasks available, waiting…
t=15h47: No eligible fill-in tasks available, waiting…
t=15h48: No eligible fill-in tasks available, waiting…
t=15h49: No eligible fill-in tasks available, waiting…
t=15h50: No eligible fill-in tasks available, waiting…
t=15h51: No eligible fill-in tasks available, waiting…
t=15h52: No eligible fill-in tasks available, waiting…
t=15h53: No eligible fill-in tasks available, waiting…
t=15h54: No eligible fill-in tasks available, waiting…
t=15h55: No eligible fill-in tasks available, waiting…
t=15h56: No eligible fill-in tasks available, waiting…
t=15h57: No eligible fill-in tasks available, waiting…
t=15h58: No eligible fill-in tasks available, waiting…
t=15h59: No eligible fill-in tasks available, waiting…
t=16h00: No eligible fill-in tasks available, waiting…
t=16h01: No eligible fill-in tasks available, waiting…
t=16h02: No eligible fill-in tasks available, waiting…
```

```
t=16h03: No eligible fill-in tasks available, waiting…
t=16h04: No eligible fill-in tasks available, waiting…
t=16h05: No eligible fill-in tasks available, waiting…
t=16h06: No eligible fill-in tasks available, waiting…
t=16h07: No eligible fill-in tasks available, waiting…
t=16h08: No eligible fill-in tasks available, waiting…
t=16h09: No eligible fill-in tasks available, waiting…
t=16h10: No eligible fill-in tasks available, waiting…
t=16h11: No eligible fill-in tasks available, waiting…
t=16h12: No eligible fill-in tasks available, waiting…
t=16h13: No eligible fill-in tasks available, waiting…
t=16h14: No eligible fill-in tasks available, waiting…
t=16h15: No eligible fill-in tasks available, waiting…
t=16h16: No eligible fill-in tasks available, waiting…
t=16h17: No eligible fill-in tasks available, waiting…
t=16h18: No eligible fill-in tasks available, waiting…
t=16h19: No eligible fill-in tasks available, waiting…
t=16h20: No eligible fill-in tasks available, waiting…
t=16h21: No eligible fill-in tasks available, waiting…
t=16h22: No eligible fill-in tasks available, waiting…
t=16h23: No eligible fill-in tasks available, waiting…
t=16h24: No eligible fill-in tasks available, waiting…
t=16h25: No eligible fill-in tasks available, waiting…
t=16h26: No eligible fill-in tasks available, waiting…
```

```
t=16h27: No eligible fill-in tasks available, waiting…
t=16h28: No eligible fill-in tasks available, waiting…
t=16h29: No eligible fill-in tasks available, waiting…
t=16h30: No eligible fill-in tasks available, waiting…
t=16h31: No eligible fill-in tasks available, waiting…
t=16h32: No eligible fill-in tasks available, waiting…
t=16h33: No eligible fill-in tasks available, waiting…
t=16h34: No eligible fill-in tasks available, waiting…
t=16h35: No eligible fill-in tasks available, waiting…
t=16h36: No eligible fill-in tasks available, waiting…
t=16h37: No eligible fill-in tasks available, waiting…
t=16h38: No eligible fill-in tasks available, waiting…
t=16h39: No eligible fill-in tasks available, waiting…
t=16h40: No eligible fill-in tasks available, waiting…
t=16h41: No eligible fill-in tasks available, waiting…
t=16h42: No eligible fill-in tasks available, waiting…
t=16h43: No eligible fill-in tasks available, waiting…
t=16h44: No eligible fill-in tasks available, waiting…
t=16h45: No eligible fill-in tasks available, waiting…
t=16h46: No eligible fill-in tasks available, waiting…
t=16h47: No eligible fill-in tasks available, waiting…
t=16h48: No eligible fill-in tasks available, waiting…
t=16h49: No eligible fill-in tasks available, waiting…
t=16h50: No eligible fill-in tasks available, waiting…
```

```
t=16h51: No eligible fill-in tasks available, waiting…
t=16h52: No eligible fill-in tasks available, waiting…
t=16h53: No eligible fill-in tasks available, waiting…
t=16h54: No eligible fill-in tasks available, waiting…
t=16h55: No eligible fill-in tasks available, waiting…
t=16h56: No eligible fill-in tasks available, waiting…
t=16h57: No eligible fill-in tasks available, waiting…
t=16h58: No eligible fill-in tasks available, waiting…
t=16h59: No eligible fill-in tasks available, waiting…
t=17h00: No eligible fill-in tasks available, waiting…
t=17h01: No eligible fill-in tasks available, waiting…
t=17h02: No eligible fill-in tasks available, waiting…
t=17h03: No eligible fill-in tasks available, waiting…
t=17h04: No eligible fill-in tasks available, waiting…
t=17h05: No eligible fill-in tasks available, waiting…
t=17h06: No eligible fill-in tasks available, waiting…
t=17h07: No eligible fill-in tasks available, waiting…
t=17h08: No eligible fill-in tasks available, waiting…
t=17h09: No eligible fill-in tasks available, waiting…
t=17h10: No eligible fill-in tasks available, waiting…
t=17h11: No eligible fill-in tasks available, waiting…
t=17h12: No eligible fill-in tasks available, waiting…
t=17h13: No eligible fill-in tasks available, waiting…
t=17h14: No eligible fill-in tasks available, waiting…
```

```
t=17h15: No eligible fill-in tasks available, waiting…
t=17h16: No eligible fill-in tasks available, waiting…
t=17h17: No eligible fill-in tasks available, waiting…
t=17h18: No eligible fill-in tasks available, waiting…
t=17h19: No eligible fill-in tasks available, waiting…
t=17h20: No eligible fill-in tasks available, waiting…
t=17h21: No eligible fill-in tasks available, waiting…
t=17h22: No eligible fill-in tasks available, waiting…
t=17h23: No eligible fill-in tasks available, waiting…
t=17h24: No eligible fill-in tasks available, waiting…
t=17h25: No eligible fill-in tasks available, waiting…
t=17h26: No eligible fill-in tasks available, waiting…
t=17h27: No eligible fill-in tasks available, waiting…
t=17h28: No eligible fill-in tasks available, waiting…
t=17h29: No eligible fill-in tasks available, waiting…
t=17h30: No eligible fill-in tasks available, waiting…
t=17h31: No eligible fill-in tasks available, waiting…
t=17h32: No eligible fill-in tasks available, waiting…
t=17h33: No eligible fill-in tasks available, waiting…
t=17h34: No eligible fill-in tasks available, waiting…
t=17h35: No eligible fill-in tasks available, waiting…
t=17h36: No eligible fill-in tasks available, waiting…
t=17h37: No eligible fill-in tasks available, waiting…
t=17h38: No eligible fill-in tasks available, waiting…
```

```
 t=17h39: No eligible fill-in tasks available, waiting…

 t=17h40: No eligible fill-in tasks available, waiting…

 t=17h41: No eligible fill-in tasks available, waiting…

 t=17h42: No eligible fill-in tasks available, waiting…

 t=17h43: No eligible fill-in tasks available, waiting…

 t=17h44: No eligible fill-in tasks available, waiting…

 t=17h45: No eligible fill-in tasks available, waiting…

 t=17h46: No eligible fill-in tasks available, waiting…

 t=17h47: No eligible fill-in tasks available, waiting…

 t=17h48: No eligible fill-in tasks available, waiting…

 t=17h49: No eligible fill-in tasks available, waiting…

 t=17h50: No eligible fill-in tasks available, waiting…

 t=17h51: No eligible fill-in tasks available, waiting…

 t=17h52: No eligible fill-in tasks available, waiting…

 t=17h53: No eligible fill-in tasks available, waiting…

 t=17h54: No eligible fill-in tasks available, waiting…

 t=17h55: No eligible fill-in tasks available, waiting…

 t=17h56: No eligible fill-in tasks available, waiting…

 t=17h57: No eligible fill-in tasks available, waiting…

 t=17h58: No eligible fill-in tasks available, waiting…

 t=17h59: No eligible fill-in tasks available, waiting…

t=18h00: started 'CS111 Class' for 90 mins…
         t=19h30, task completed!

t=19h30: started 'Work on written parts of CS110 Assignment' for 120 mins…
         t=21h30, task completed!
```

```
t=21h30: No eligible fill-in tasks available, waiting…
t=21h31: No eligible fill-in tasks available, waiting…
t=21h32: No eligible fill-in tasks available, waiting…
t=21h33: No eligible fill-in tasks available, waiting…
t=21h34: No eligible fill-in tasks available, waiting…
t=21h35: No eligible fill-in tasks available, waiting…
t=21h36: No eligible fill-in tasks available, waiting…
t=21h37: No eligible fill-in tasks available, waiting…
t=21h38: No eligible fill-in tasks available, waiting…
t=21h39: No eligible fill-in tasks available, waiting…
t=21h40: No eligible fill-in tasks available, waiting…
t=21h41: No eligible fill-in tasks available, waiting…
t=21h42: No eligible fill-in tasks available, waiting…
t=21h43: No eligible fill-in tasks available, waiting…
t=21h44: No eligible fill-in tasks available, waiting…
t=21h45: No eligible fill-in tasks available, waiting…
t=21h46: No eligible fill-in tasks available, waiting…
t=21h47: No eligible fill-in tasks available, waiting…
t=21h48: No eligible fill-in tasks available, waiting…
t=21h49: No eligible fill-in tasks available, waiting…
t=21h50: No eligible fill-in tasks available, waiting…
t=21h51: No eligible fill-in tasks available, waiting…
t=21h52: No eligible fill-in tasks available, waiting…
t=21h53: No eligible fill-in tasks available, waiting…
```

```
 t=21h54: No eligible fill-in tasks available, waiting…

 t=21h55: No eligible fill-in tasks available, waiting…

 t=21h56: No eligible fill-in tasks available, waiting…

 t=21h57: No eligible fill-in tasks available, waiting…

 t=21h58: No eligible fill-in tasks available, waiting…

 t=21h59: No eligible fill-in tasks available, waiting…

 t=22h00: started 'Bike back to 9Floor after the library closes at 22h' for 12
mins…
          t=22h12, task completed!


 Completed all planned tasks in 14h12min!
```

4.6 Switching order

[ ]:
```
# Original order
tasks = [
    Task(id=0, description='Get up at 8:00 AM',
         duration=30, dependencies=[]),
    Task(id=1, description='Walk to 7-11 and have breakfast',
         duration=20, dependencies=[0]),
    Task(id=2, description='Bike to the Main NTU Library',
         duration=12, dependencies=[0, 1]),
    Task(id=3, description='Do Pre-Class Work for CS111',
         duration=50, dependencies=[2]),
    Task(id=4, description='Formalize the Utility function definition for CS110␣
 ↪Assignment',
         duration=90, dependencies=[2]),
    Task(id=5, description='Prepare video scripts and whiteboards for CS110␣
 ↪Assignment',
         duration=80, dependencies=[2]),
    Task(id=6, description="Have lunch at JJ's Poke Bowl and bike back to the␣
 ↪library",
         duration=75, dependencies=[2], scheduled_time=13*60),
    Task(id=7, description='CS111 Class',
         duration=90, dependencies=[3], scheduled_time=18*60),
    Task(id=8, description='Record CS110 Videos',
         duration=75, dependencies=[4, 5]),
    Task(id=9, description='Work on written parts of CS110 Assignment',
         duration=120, dependencies=[8]),
```

```
    Task(id=10, description='Bike back to 9Floor after the library closes at␣
 ↪22h',
         duration=12, dependencies=[6, 7, 8, 9], scheduled_time=22*60)
    ]

task_scheduler = TaskScheduler(tasks)

# print the scheduler's input
task_scheduler.print_self()

start_scheduler_at = 8*60
task_scheduler.run_task_scheduler(start_scheduler_at)

expected_order = task_scheduler.executed_task_descriptions

# Different order 1
diff_order_1 = [
    Task(id=0, description='Bike back to 9Floor after the library closes at␣
 ↪22h',
         duration=12, dependencies=[7, 8, 9, 10]),
    Task(id=1, description='Get up at 8:00 AM',
         duration=30, dependencies=[]),
    Task(id=2, description='Walk to 7-11 and have breakfast',
         duration=20, dependencies=[1]),
    Task(id=3, description='Bike to the Main NTU Library',
         duration=12, dependencies=[1, 2]),
    Task(id=4, description='Do Pre-Class Work for CS111',
         duration=50, dependencies=[3]),
    Task(id=5, description='Formalize the Utility function definition for CS110␣
 ↪Assignment',
         duration=90, dependencies=[3]),
    Task(id=6, description='Prepare video scripts and whiteboards for CS110␣
 ↪Assignment',
         duration=80, dependencies=[3]),
    Task(id=7, description="Have lunch at JJ's Poke Bowl and bike back to the␣
 ↪library",
         duration=75, dependencies=[3], scheduled_time=13*60),
    Task(id=8, description='CS111 Class',
         duration=90, dependencies=[4], scheduled_time=18*60),
    Task(id=9, description='Record CS110 Videos',
         duration=75, dependencies=[5, 6]),
    Task(id=10, description='Work on written parts of CS110 Assignment',
         duration=120, dependencies=[9])
]

task_scheduler_diff_order_1 = TaskScheduler(diff_order_1)
```

```
task_scheduler_diff_order_1.print_self()

start_scheduler_at = 8*60
task_scheduler_diff_order_1.run_task_scheduler(start_scheduler_at)

task_descriptions_diff_order_1 = task_scheduler_diff_order_1.
 ↪executed_task_descriptions

assert task_descriptions_diff_order_1 == expected_order, f"Mismatch found:␣
 ↪{task_descriptions_diff_order_1} != {expected_order}"

# Different order 2
diff_order_2 = [
    Task(id=0, description='Get up at 8:00 AM',
         duration=30, dependencies=[]),
    Task(id=1, description='Walk to 7-11 and have breakfast',
         duration=20, dependencies=[0]),
    Task(id=2, description='Bike to the Main NTU Library',
         duration=12, dependencies=[0, 1]),
    Task(id=3, description='Have lunch at JJ\'s Poke Bowl and bike back to the␣
 ↪library',
         duration=75, dependencies=[2], scheduled_time=13*60),
    Task(id=4, description='Do Pre-Class Work for CS111',
         duration=50, dependencies=[2]),
    Task(id=5, description='Formalize the Utility function definition for CS110␣
 ↪Assignment',
         duration=90, dependencies=[2]),
    Task(id=6, description='Record CS110 Videos',
         duration=75, dependencies=[5, 10]),
    Task(id=7, description='CS111 Class',
         duration=90, dependencies=[4], scheduled_time=18*60),
    Task(id=8, description='Bike back to 9Floor after the library closes at␣
 ↪22h',
         duration=12, dependencies=[3, 6, 7, 9]),
    Task(id=9, description='Work on written parts of CS110 Assignment',
         duration=120, dependencies=[6]),
    Task(id=10, description='Prepare video scripts and whiteboards for CS110␣
 ↪Assignment',
         duration=80, dependencies=[2])
]

task_scheduler_diff_order_2 = TaskScheduler(diff_order_2)

task_scheduler_diff_order_2.print_self()

start_scheduler_at = 8*60
```

```
task_scheduler_diff_order_2.run_task_scheduler(start_scheduler_at)

task_descriptions_diff_order_2 = task_scheduler_diff_order_2.
 ↪executed_task_descriptions

assert task_descriptions_diff_order_2 == expected_order, f"Mismatch found:␣
 ↪{task_descriptions_diff_order_2} != {expected_order}"
```

```
Tasks added to the simple scheduler:
--------------------------------------
  'Get up at 8:00 AM', duration = 30 mins.
  'Walk to 7-11 and have breakfast', duration = 20 mins.
          This task depends on others!
These are the dependencies: [0]
  'Bike to the Main NTU Library', duration = 12 mins.
          This task depends on others!
These are the dependencies: [0, 1]
  'Do Pre-Class Work for CS111', duration = 50 mins.
          This task depends on others!
These are the dependencies: [2]
  'Formalize the Utility function definition for CS110 Assignment', duration =
90 mins.
          This task depends on others!
These are the dependencies: [2]
  'Prepare video scripts and whiteboards for CS110 Assignment', duration = 80
mins.
          This task depends on others!
These are the dependencies: [2]
  'Have lunch at JJ's Poke Bowl and bike back to the library', duration = 75
mins.
          This task depends on others!
These are the dependencies: [2]
  'CS111 Class', duration = 90 mins.
          This task depends on others!
These are the dependencies: [3]
  'Record CS110 Videos', duration = 75 mins.
          This task depends on others!
These are the dependencies: [4, 5]
  'Work on written parts of CS110 Assignment', duration = 120 mins.
          This task depends on others!
These are the dependencies: [8]
  'Bike back to 9Floor after the library closes at 22h', duration = 12 mins.
          This task depends on others!
These are the dependencies: [6, 7, 8, 9]
Running a simple scheduler:

This is my priority queue:
0) Task 0: Get up at 8:00 AM
```

```
        Duration: 30 minutes
        Status: I
        Priority Score = 70

After adding tasks to the queue:
 t=8h00: started 'Get up at 8:00 AM' for 30 mins…
          t=8h30, task completed!

This is my priority queue:
0) Task 1: Walk to 7-11 and have breakfast
        Duration: 20 minutes
        Status: I
        Priority Score = 80

After adding tasks to the queue:
 t=8h30: started 'Walk to 7-11 and have breakfast' for 20 mins…
          t=8h50, task completed!

This is my priority queue:
0) Task 2: Bike to the Main NTU Library
        Duration: 12 minutes
        Status: I
        Priority Score = 88

After adding tasks to the queue:
 t=8h50: started 'Bike to the Main NTU Library' for 12 mins…
          t=9h02, task completed!

This is my priority queue:
0) Task 6: Have lunch at JJ's Poke Bowl and bike back to the library
        Duration: 75 minutes
        Status: I
        Priority Score = 2405

1) Task 3: Do Pre-Class Work for CS111
        Duration: 50 minutes
        Status: I
        Priority Score = 50

2) Task 5: Prepare video scripts and whiteboards for CS110 Assignment
        Duration: 80 minutes
        Status: I
        Priority Score = 20

3) Task 4: Formalize the Utility function definition for CS110 Assignment
        Duration: 90 minutes
        Status: I
        Priority Score = 10
```

```
After adding tasks to the queue:
  Fill-in task available! t=9h02: started 'Do Pre-Class Work for CS111' for 50
mins…
          t=9h52, task completed!

  Fill-in task available! t=9h52: started 'Prepare video scripts and whiteboards
for CS110 Assignment' for 80 mins…
          t=11h12, task completed!

  Fill-in task available! t=11h12: started 'Formalize the Utility function
definition for CS110 Assignment' for 90 mins…
          t=12h42, task completed!

  t=12h42: No eligible fill-in tasks available, waiting…

  t=12h43: No eligible fill-in tasks available, waiting…

  t=12h44: No eligible fill-in tasks available, waiting…

  t=12h45: No eligible fill-in tasks available, waiting…

  t=12h46: No eligible fill-in tasks available, waiting…

  t=12h47: No eligible fill-in tasks available, waiting…

  t=12h48: No eligible fill-in tasks available, waiting…

  t=12h49: No eligible fill-in tasks available, waiting…

  t=12h50: No eligible fill-in tasks available, waiting…

  t=12h51: No eligible fill-in tasks available, waiting…

  t=12h52: No eligible fill-in tasks available, waiting…

  t=12h53: No eligible fill-in tasks available, waiting…

  t=12h54: No eligible fill-in tasks available, waiting…

  t=12h55: No eligible fill-in tasks available, waiting…

  t=12h56: No eligible fill-in tasks available, waiting…

  t=12h57: No eligible fill-in tasks available, waiting…

  t=12h58: No eligible fill-in tasks available, waiting…
```

```
  t=12h59: No eligible fill-in tasks available, waiting…

 t=13h00: started 'Have lunch at JJ's Poke Bowl and bike back to the library'
for 75 mins…
          t=14h15, task completed!

This is my priority queue:
0) Task 7: CS111 Class
        Duration: 90 minutes
        Status: I
        Priority Score = 2260

1) Task 8: Record CS110 Videos
        Duration: 75 minutes
        Status: I
        Priority Score = 25

After adding tasks to the queue:
  Fill-in task available! t=14h15: started 'Record CS110 Videos' for 75 mins…
          t=15h30, task completed!

  t=15h30: No eligible fill-in tasks available, waiting…

  t=15h31: No eligible fill-in tasks available, waiting…

  t=15h32: No eligible fill-in tasks available, waiting…

  t=15h33: No eligible fill-in tasks available, waiting…

  t=15h34: No eligible fill-in tasks available, waiting…

  t=15h35: No eligible fill-in tasks available, waiting…

  t=15h36: No eligible fill-in tasks available, waiting…

  t=15h37: No eligible fill-in tasks available, waiting…

  t=15h38: No eligible fill-in tasks available, waiting…

  t=15h39: No eligible fill-in tasks available, waiting…

  t=15h40: No eligible fill-in tasks available, waiting…

  t=15h41: No eligible fill-in tasks available, waiting…

  t=15h42: No eligible fill-in tasks available, waiting…

  t=15h43: No eligible fill-in tasks available, waiting…
```

```
t=15h44: No eligible fill-in tasks available, waiting…
t=15h45: No eligible fill-in tasks available, waiting…
t=15h46: No eligible fill-in tasks available, waiting…
t=15h47: No eligible fill-in tasks available, waiting…
t=15h48: No eligible fill-in tasks available, waiting…
t=15h49: No eligible fill-in tasks available, waiting…
t=15h50: No eligible fill-in tasks available, waiting…
t=15h51: No eligible fill-in tasks available, waiting…
t=15h52: No eligible fill-in tasks available, waiting…
t=15h53: No eligible fill-in tasks available, waiting…
t=15h54: No eligible fill-in tasks available, waiting…
t=15h55: No eligible fill-in tasks available, waiting…
t=15h56: No eligible fill-in tasks available, waiting…
t=15h57: No eligible fill-in tasks available, waiting…
t=15h58: No eligible fill-in tasks available, waiting…
t=15h59: No eligible fill-in tasks available, waiting…
t=16h00: No eligible fill-in tasks available, waiting…
t=16h01: No eligible fill-in tasks available, waiting…
t=16h02: No eligible fill-in tasks available, waiting…
t=16h03: No eligible fill-in tasks available, waiting…
t=16h04: No eligible fill-in tasks available, waiting…
t=16h05: No eligible fill-in tasks available, waiting…
t=16h06: No eligible fill-in tasks available, waiting…
t=16h07: No eligible fill-in tasks available, waiting…
```

```
t=16h08: No eligible fill-in tasks available, waiting…
t=16h09: No eligible fill-in tasks available, waiting…
t=16h10: No eligible fill-in tasks available, waiting…
t=16h11: No eligible fill-in tasks available, waiting…
t=16h12: No eligible fill-in tasks available, waiting…
t=16h13: No eligible fill-in tasks available, waiting…
t=16h14: No eligible fill-in tasks available, waiting…
t=16h15: No eligible fill-in tasks available, waiting…
t=16h16: No eligible fill-in tasks available, waiting…
t=16h17: No eligible fill-in tasks available, waiting…
t=16h18: No eligible fill-in tasks available, waiting…
t=16h19: No eligible fill-in tasks available, waiting…
t=16h20: No eligible fill-in tasks available, waiting…
t=16h21: No eligible fill-in tasks available, waiting…
t=16h22: No eligible fill-in tasks available, waiting…
t=16h23: No eligible fill-in tasks available, waiting…
t=16h24: No eligible fill-in tasks available, waiting…
t=16h25: No eligible fill-in tasks available, waiting…
t=16h26: No eligible fill-in tasks available, waiting…
t=16h27: No eligible fill-in tasks available, waiting…
t=16h28: No eligible fill-in tasks available, waiting…
t=16h29: No eligible fill-in tasks available, waiting…
t=16h30: No eligible fill-in tasks available, waiting…
t=16h31: No eligible fill-in tasks available, waiting…
```

```
t=16h32: No eligible fill-in tasks available, waiting…
t=16h33: No eligible fill-in tasks available, waiting…
t=16h34: No eligible fill-in tasks available, waiting…
t=16h35: No eligible fill-in tasks available, waiting…
t=16h36: No eligible fill-in tasks available, waiting…
t=16h37: No eligible fill-in tasks available, waiting…
t=16h38: No eligible fill-in tasks available, waiting…
t=16h39: No eligible fill-in tasks available, waiting…
t=16h40: No eligible fill-in tasks available, waiting…
t=16h41: No eligible fill-in tasks available, waiting…
t=16h42: No eligible fill-in tasks available, waiting…
t=16h43: No eligible fill-in tasks available, waiting…
t=16h44: No eligible fill-in tasks available, waiting…
t=16h45: No eligible fill-in tasks available, waiting…
t=16h46: No eligible fill-in tasks available, waiting…
t=16h47: No eligible fill-in tasks available, waiting…
t=16h48: No eligible fill-in tasks available, waiting…
t=16h49: No eligible fill-in tasks available, waiting…
t=16h50: No eligible fill-in tasks available, waiting…
t=16h51: No eligible fill-in tasks available, waiting…
t=16h52: No eligible fill-in tasks available, waiting…
t=16h53: No eligible fill-in tasks available, waiting…
t=16h54: No eligible fill-in tasks available, waiting…
t=16h55: No eligible fill-in tasks available, waiting…
```

```
t=16h56: No eligible fill-in tasks available, waiting…
t=16h57: No eligible fill-in tasks available, waiting…
t=16h58: No eligible fill-in tasks available, waiting…
t=16h59: No eligible fill-in tasks available, waiting…
t=17h00: No eligible fill-in tasks available, waiting…
t=17h01: No eligible fill-in tasks available, waiting…
t=17h02: No eligible fill-in tasks available, waiting…
t=17h03: No eligible fill-in tasks available, waiting…
t=17h04: No eligible fill-in tasks available, waiting…
t=17h05: No eligible fill-in tasks available, waiting…
t=17h06: No eligible fill-in tasks available, waiting…
t=17h07: No eligible fill-in tasks available, waiting…
t=17h08: No eligible fill-in tasks available, waiting…
t=17h09: No eligible fill-in tasks available, waiting…
t=17h10: No eligible fill-in tasks available, waiting…
t=17h11: No eligible fill-in tasks available, waiting…
t=17h12: No eligible fill-in tasks available, waiting…
t=17h13: No eligible fill-in tasks available, waiting…
t=17h14: No eligible fill-in tasks available, waiting…
t=17h15: No eligible fill-in tasks available, waiting…
t=17h16: No eligible fill-in tasks available, waiting…
t=17h17: No eligible fill-in tasks available, waiting…
t=17h18: No eligible fill-in tasks available, waiting…
t=17h19: No eligible fill-in tasks available, waiting…
```

```
t=17h20: No eligible fill-in tasks available, waiting…
t=17h21: No eligible fill-in tasks available, waiting…
t=17h22: No eligible fill-in tasks available, waiting…
t=17h23: No eligible fill-in tasks available, waiting…
t=17h24: No eligible fill-in tasks available, waiting…
t=17h25: No eligible fill-in tasks available, waiting…
t=17h26: No eligible fill-in tasks available, waiting…
t=17h27: No eligible fill-in tasks available, waiting…
t=17h28: No eligible fill-in tasks available, waiting…
t=17h29: No eligible fill-in tasks available, waiting…
t=17h30: No eligible fill-in tasks available, waiting…
t=17h31: No eligible fill-in tasks available, waiting…
t=17h32: No eligible fill-in tasks available, waiting…
t=17h33: No eligible fill-in tasks available, waiting…
t=17h34: No eligible fill-in tasks available, waiting…
t=17h35: No eligible fill-in tasks available, waiting…
t=17h36: No eligible fill-in tasks available, waiting…
t=17h37: No eligible fill-in tasks available, waiting…
t=17h38: No eligible fill-in tasks available, waiting…
t=17h39: No eligible fill-in tasks available, waiting…
t=17h40: No eligible fill-in tasks available, waiting…
t=17h41: No eligible fill-in tasks available, waiting…
t=17h42: No eligible fill-in tasks available, waiting…
t=17h43: No eligible fill-in tasks available, waiting…
```

```
 t=17h44: No eligible fill-in tasks available, waiting…

 t=17h45: No eligible fill-in tasks available, waiting…

 t=17h46: No eligible fill-in tasks available, waiting…

 t=17h47: No eligible fill-in tasks available, waiting…

 t=17h48: No eligible fill-in tasks available, waiting…

 t=17h49: No eligible fill-in tasks available, waiting…

 t=17h50: No eligible fill-in tasks available, waiting…

 t=17h51: No eligible fill-in tasks available, waiting…

 t=17h52: No eligible fill-in tasks available, waiting…

 t=17h53: No eligible fill-in tasks available, waiting…

 t=17h54: No eligible fill-in tasks available, waiting…

 t=17h55: No eligible fill-in tasks available, waiting…

 t=17h56: No eligible fill-in tasks available, waiting…

 t=17h57: No eligible fill-in tasks available, waiting…

 t=17h58: No eligible fill-in tasks available, waiting…

 t=17h59: No eligible fill-in tasks available, waiting…

t=18h00: started 'CS111 Class' for 90 mins…
         t=19h30, task completed!

This is my priority queue:
0) Task 9: Work on written parts of CS110 Assignment
        Duration: 120 minutes
        Status: I
        Priority Score = 20

After adding tasks to the queue:
t=19h30: started 'Work on written parts of CS110 Assignment' for 120 mins…
         t=21h30, task completed!

This is my priority queue:
0) Task 10: Bike back to 9Floor after the library closes at 22h
```

```
        Duration: 12 minutes
        Status: I
        Priority Score = 388

After adding tasks to the queue:
  t=21h30: No eligible fill-in tasks available, waiting…

  t=21h31: No eligible fill-in tasks available, waiting…

  t=21h32: No eligible fill-in tasks available, waiting…

  t=21h33: No eligible fill-in tasks available, waiting…

  t=21h34: No eligible fill-in tasks available, waiting…

  t=21h35: No eligible fill-in tasks available, waiting…

  t=21h36: No eligible fill-in tasks available, waiting…

  t=21h37: No eligible fill-in tasks available, waiting…

  t=21h38: No eligible fill-in tasks available, waiting…

  t=21h39: No eligible fill-in tasks available, waiting…

  t=21h40: No eligible fill-in tasks available, waiting…

  t=21h41: No eligible fill-in tasks available, waiting…

  t=21h42: No eligible fill-in tasks available, waiting…

  t=21h43: No eligible fill-in tasks available, waiting…

  t=21h44: No eligible fill-in tasks available, waiting…

  t=21h45: No eligible fill-in tasks available, waiting…

  t=21h46: No eligible fill-in tasks available, waiting…

  t=21h47: No eligible fill-in tasks available, waiting…

  t=21h48: No eligible fill-in tasks available, waiting…

  t=21h49: No eligible fill-in tasks available, waiting…

  t=21h50: No eligible fill-in tasks available, waiting…

  t=21h51: No eligible fill-in tasks available, waiting…
```

```
  t=21h52: No eligible fill-in tasks available, waiting…

  t=21h53: No eligible fill-in tasks available, waiting…

  t=21h54: No eligible fill-in tasks available, waiting…

  t=21h55: No eligible fill-in tasks available, waiting…

  t=21h56: No eligible fill-in tasks available, waiting…

  t=21h57: No eligible fill-in tasks available, waiting…

  t=21h58: No eligible fill-in tasks available, waiting…

  t=21h59: No eligible fill-in tasks available, waiting…

 t=22h00: started 'Bike back to 9Floor after the library closes at 22h' for 12
mins…
          t=22h12, task completed!


  Completed all planned tasks in 14h12min!
Tasks added to the simple scheduler:
--------------------------------------
  'Bike back to 9Floor after the library closes at 22h', duration = 12 mins.
           This task depends on others!
These are the dependencies: [7, 8, 9, 10]
  'Get up at 8:00 AM', duration = 30 mins.
  'Walk to 7-11 and have breakfast', duration = 20 mins.
           This task depends on others!
These are the dependencies: [1]
  'Bike to the Main NTU Library', duration = 12 mins.
           This task depends on others!
These are the dependencies: [1, 2]
  'Do Pre-Class Work for CS111', duration = 50 mins.
           This task depends on others!
These are the dependencies: [3]
  'Formalize the Utility function definition for CS110 Assignment', duration =
90 mins.
           This task depends on others!
These are the dependencies: [3]
  'Prepare video scripts and whiteboards for CS110 Assignment', duration = 80
mins.
           This task depends on others!
These are the dependencies: [3]
  'Have lunch at JJ's Poke Bowl and bike back to the library', duration = 75
mins.
```

```
          This task depends on others!
These are the dependencies: [3]
  'CS111 Class', duration = 90 mins.
          This task depends on others!
These are the dependencies: [4]
  'Record CS110 Videos', duration = 75 mins.
          This task depends on others!
These are the dependencies: [5, 6]
  'Work on written parts of CS110 Assignment', duration = 120 mins.
          This task depends on others!
These are the dependencies: [9]
Running a simple scheduler:

This is my priority queue:
0) Task 1: Get up at 8:00 AM
        Duration: 30 minutes
        Status: I
        Priority Score = 70

After adding tasks to the queue:
 t=8h00: started 'Get up at 8:00 AM' for 30 mins…
          t=8h30, task completed!

This is my priority queue:
0) Task 2: Walk to 7-11 and have breakfast
        Duration: 20 minutes
        Status: I
        Priority Score = 80

After adding tasks to the queue:
 t=8h30: started 'Walk to 7-11 and have breakfast' for 20 mins…
          t=8h50, task completed!

This is my priority queue:
0) Task 3: Bike to the Main NTU Library
        Duration: 12 minutes
        Status: I
        Priority Score = 88

After adding tasks to the queue:
 t=8h50: started 'Bike to the Main NTU Library' for 12 mins…
          t=9h02, task completed!

This is my priority queue:
0) Task 7: Have lunch at JJ's Poke Bowl and bike back to the library
        Duration: 75 minutes
        Status: I
        Priority Score = 2405
```

```
1) Task 4: Do Pre-Class Work for CS111
        Duration: 50 minutes
        Status: I
        Priority Score = 50

2) Task 6: Prepare video scripts and whiteboards for CS110 Assignment
        Duration: 80 minutes
        Status: I
        Priority Score = 20

3) Task 5: Formalize the Utility function definition for CS110 Assignment
        Duration: 90 minutes
        Status: I
        Priority Score = 10

After adding tasks to the queue:
  Fill-in task available! t=9h02: started 'Do Pre-Class Work for CS111' for 50
mins…
          t=9h52, task completed!

  Fill-in task available! t=9h52: started 'Prepare video scripts and whiteboards
for CS110 Assignment' for 80 mins…
          t=11h12, task completed!

  Fill-in task available! t=11h12: started 'Formalize the Utility function
definition for CS110 Assignment' for 90 mins…
          t=12h42, task completed!

  t=12h42: No eligible fill-in tasks available, waiting…

  t=12h43: No eligible fill-in tasks available, waiting…

  t=12h44: No eligible fill-in tasks available, waiting…

  t=12h45: No eligible fill-in tasks available, waiting…

  t=12h46: No eligible fill-in tasks available, waiting…

  t=12h47: No eligible fill-in tasks available, waiting…

  t=12h48: No eligible fill-in tasks available, waiting…

  t=12h49: No eligible fill-in tasks available, waiting…

  t=12h50: No eligible fill-in tasks available, waiting…

  t=12h51: No eligible fill-in tasks available, waiting…
```

```
  t=12h52: No eligible fill-in tasks available, waiting…

  t=12h53: No eligible fill-in tasks available, waiting…

  t=12h54: No eligible fill-in tasks available, waiting…

  t=12h55: No eligible fill-in tasks available, waiting…

  t=12h56: No eligible fill-in tasks available, waiting…

  t=12h57: No eligible fill-in tasks available, waiting…

  t=12h58: No eligible fill-in tasks available, waiting…

  t=12h59: No eligible fill-in tasks available, waiting…

 t=13h00: started 'Have lunch at JJ's Poke Bowl and bike back to the library'
for 75 mins…
          t=14h15, task completed!

This is my priority queue:
0) Task 8: CS111 Class
        Duration: 90 minutes
        Status: I
        Priority Score = 2260

1) Task 9: Record CS110 Videos
        Duration: 75 minutes
        Status: I
        Priority Score = 25

After adding tasks to the queue:
  Fill-in task available! t=14h15: started 'Record CS110 Videos' for 75 mins…
          t=15h30, task completed!

  t=15h30: No eligible fill-in tasks available, waiting…

  t=15h31: No eligible fill-in tasks available, waiting…

  t=15h32: No eligible fill-in tasks available, waiting…

  t=15h33: No eligible fill-in tasks available, waiting…

  t=15h34: No eligible fill-in tasks available, waiting…

  t=15h35: No eligible fill-in tasks available, waiting…
```

t=15h36: No eligible fill-in tasks available, waiting…

t=15h37: No eligible fill-in tasks available, waiting…

t=15h38: No eligible fill-in tasks available, waiting…

t=15h39: No eligible fill-in tasks available, waiting…

t=15h40: No eligible fill-in tasks available, waiting…

t=15h41: No eligible fill-in tasks available, waiting…

t=15h42: No eligible fill-in tasks available, waiting…

t=15h43: No eligible fill-in tasks available, waiting…

t=15h44: No eligible fill-in tasks available, waiting…

t=15h45: No eligible fill-in tasks available, waiting…

t=15h46: No eligible fill-in tasks available, waiting…

t=15h47: No eligible fill-in tasks available, waiting…

t=15h48: No eligible fill-in tasks available, waiting…

t=15h49: No eligible fill-in tasks available, waiting…

t=15h50: No eligible fill-in tasks available, waiting…

t=15h51: No eligible fill-in tasks available, waiting…

t=15h52: No eligible fill-in tasks available, waiting…

t=15h53: No eligible fill-in tasks available, waiting…

t=15h54: No eligible fill-in tasks available, waiting…

t=15h55: No eligible fill-in tasks available, waiting…

t=15h56: No eligible fill-in tasks available, waiting…

t=15h57: No eligible fill-in tasks available, waiting…

t=15h58: No eligible fill-in tasks available, waiting…

t=15h59: No eligible fill-in tasks available, waiting…

```
t=16h00: No eligible fill-in tasks available, waiting…
t=16h01: No eligible fill-in tasks available, waiting…
t=16h02: No eligible fill-in tasks available, waiting…
t=16h03: No eligible fill-in tasks available, waiting…
t=16h04: No eligible fill-in tasks available, waiting…
t=16h05: No eligible fill-in tasks available, waiting…
t=16h06: No eligible fill-in tasks available, waiting…
t=16h07: No eligible fill-in tasks available, waiting…
t=16h08: No eligible fill-in tasks available, waiting…
t=16h09: No eligible fill-in tasks available, waiting…
t=16h10: No eligible fill-in tasks available, waiting…
t=16h11: No eligible fill-in tasks available, waiting…
t=16h12: No eligible fill-in tasks available, waiting…
t=16h13: No eligible fill-in tasks available, waiting…
t=16h14: No eligible fill-in tasks available, waiting…
t=16h15: No eligible fill-in tasks available, waiting…
t=16h16: No eligible fill-in tasks available, waiting…
t=16h17: No eligible fill-in tasks available, waiting…
t=16h18: No eligible fill-in tasks available, waiting…
t=16h19: No eligible fill-in tasks available, waiting…
t=16h20: No eligible fill-in tasks available, waiting…
t=16h21: No eligible fill-in tasks available, waiting…
t=16h22: No eligible fill-in tasks available, waiting…
t=16h23: No eligible fill-in tasks available, waiting…
```

```
t=16h24: No eligible fill-in tasks available, waiting…
t=16h25: No eligible fill-in tasks available, waiting…
t=16h26: No eligible fill-in tasks available, waiting…
t=16h27: No eligible fill-in tasks available, waiting…
t=16h28: No eligible fill-in tasks available, waiting…
t=16h29: No eligible fill-in tasks available, waiting…
t=16h30: No eligible fill-in tasks available, waiting…
t=16h31: No eligible fill-in tasks available, waiting…
t=16h32: No eligible fill-in tasks available, waiting…
t=16h33: No eligible fill-in tasks available, waiting…
t=16h34: No eligible fill-in tasks available, waiting…
t=16h35: No eligible fill-in tasks available, waiting…
t=16h36: No eligible fill-in tasks available, waiting…
t=16h37: No eligible fill-in tasks available, waiting…
t=16h38: No eligible fill-in tasks available, waiting…
t=16h39: No eligible fill-in tasks available, waiting…
t=16h40: No eligible fill-in tasks available, waiting…
t=16h41: No eligible fill-in tasks available, waiting…
t=16h42: No eligible fill-in tasks available, waiting…
t=16h43: No eligible fill-in tasks available, waiting…
t=16h44: No eligible fill-in tasks available, waiting…
t=16h45: No eligible fill-in tasks available, waiting…
t=16h46: No eligible fill-in tasks available, waiting…
t=16h47: No eligible fill-in tasks available, waiting…
```

t=16h48: No eligible fill-in tasks available, waiting…

t=16h49: No eligible fill-in tasks available, waiting…

t=16h50: No eligible fill-in tasks available, waiting…

t=16h51: No eligible fill-in tasks available, waiting…

t=16h52: No eligible fill-in tasks available, waiting…

t=16h53: No eligible fill-in tasks available, waiting…

t=16h54: No eligible fill-in tasks available, waiting…

t=16h55: No eligible fill-in tasks available, waiting…

t=16h56: No eligible fill-in tasks available, waiting…

t=16h57: No eligible fill-in tasks available, waiting…

t=16h58: No eligible fill-in tasks available, waiting…

t=16h59: No eligible fill-in tasks available, waiting…

t=17h00: No eligible fill-in tasks available, waiting…

t=17h01: No eligible fill-in tasks available, waiting…

t=17h02: No eligible fill-in tasks available, waiting…

t=17h03: No eligible fill-in tasks available, waiting…

t=17h04: No eligible fill-in tasks available, waiting…

t=17h05: No eligible fill-in tasks available, waiting…

t=17h06: No eligible fill-in tasks available, waiting…

t=17h07: No eligible fill-in tasks available, waiting…

t=17h08: No eligible fill-in tasks available, waiting…

t=17h09: No eligible fill-in tasks available, waiting…

t=17h10: No eligible fill-in tasks available, waiting…

t=17h11: No eligible fill-in tasks available, waiting…

```
t=17h12: No eligible fill-in tasks available, waiting…
t=17h13: No eligible fill-in tasks available, waiting…
t=17h14: No eligible fill-in tasks available, waiting…
t=17h15: No eligible fill-in tasks available, waiting…
t=17h16: No eligible fill-in tasks available, waiting…
t=17h17: No eligible fill-in tasks available, waiting…
t=17h18: No eligible fill-in tasks available, waiting…
t=17h19: No eligible fill-in tasks available, waiting…
t=17h20: No eligible fill-in tasks available, waiting…
t=17h21: No eligible fill-in tasks available, waiting…
t=17h22: No eligible fill-in tasks available, waiting…
t=17h23: No eligible fill-in tasks available, waiting…
t=17h24: No eligible fill-in tasks available, waiting…
t=17h25: No eligible fill-in tasks available, waiting…
t=17h26: No eligible fill-in tasks available, waiting…
t=17h27: No eligible fill-in tasks available, waiting…
t=17h28: No eligible fill-in tasks available, waiting…
t=17h29: No eligible fill-in tasks available, waiting…
t=17h30: No eligible fill-in tasks available, waiting…
t=17h31: No eligible fill-in tasks available, waiting…
t=17h32: No eligible fill-in tasks available, waiting…
t=17h33: No eligible fill-in tasks available, waiting…
t=17h34: No eligible fill-in tasks available, waiting…
t=17h35: No eligible fill-in tasks available, waiting…
```

```
t=17h36: No eligible fill-in tasks available, waiting…
t=17h37: No eligible fill-in tasks available, waiting…
t=17h38: No eligible fill-in tasks available, waiting…
t=17h39: No eligible fill-in tasks available, waiting…
t=17h40: No eligible fill-in tasks available, waiting…
t=17h41: No eligible fill-in tasks available, waiting…
t=17h42: No eligible fill-in tasks available, waiting…
t=17h43: No eligible fill-in tasks available, waiting…
t=17h44: No eligible fill-in tasks available, waiting…
t=17h45: No eligible fill-in tasks available, waiting…
t=17h46: No eligible fill-in tasks available, waiting…
t=17h47: No eligible fill-in tasks available, waiting…
t=17h48: No eligible fill-in tasks available, waiting…
t=17h49: No eligible fill-in tasks available, waiting…
t=17h50: No eligible fill-in tasks available, waiting…
t=17h51: No eligible fill-in tasks available, waiting…
t=17h52: No eligible fill-in tasks available, waiting…
t=17h53: No eligible fill-in tasks available, waiting…
t=17h54: No eligible fill-in tasks available, waiting…
t=17h55: No eligible fill-in tasks available, waiting…
t=17h56: No eligible fill-in tasks available, waiting…
t=17h57: No eligible fill-in tasks available, waiting…
t=17h58: No eligible fill-in tasks available, waiting…
t=17h59: No eligible fill-in tasks available, waiting…
```

```
 t=18h00: started 'CS111 Class' for 90 mins…
          t=19h30, task completed!

This is my priority queue:
0) Task 10: Work on written parts of CS110 Assignment
        Duration: 120 minutes
        Status: I
        Priority Score = 20

After adding tasks to the queue:
 t=19h30: started 'Work on written parts of CS110 Assignment' for 120 mins…
          t=21h30, task completed!

This is my priority queue:
0) Task 0: Bike back to 9Floor after the library closes at 22h
        Duration: 12 minutes
        Status: I
        Priority Score = 88

After adding tasks to the queue:
 t=21h30: started 'Bike back to 9Floor after the library closes at 22h' for 12
mins…
          t=21h42, task completed!


  Completed all planned tasks in 13h42min!
Tasks added to the simple scheduler:
--------------------------------------
  'Get up at 8:00 AM', duration = 30 mins.
  'Walk to 7-11 and have breakfast', duration = 20 mins.
           This task depends on others!
These are the dependencies: [0]
  'Bike to the Main NTU Library', duration = 12 mins.
           This task depends on others!
These are the dependencies: [0, 1]
  'Have lunch at JJ's Poke Bowl and bike back to the library', duration = 75
mins.
           This task depends on others!
These are the dependencies: [2]
  'Do Pre-Class Work for CS111', duration = 50 mins.
           This task depends on others!
These are the dependencies: [2]
  'Formalize the Utility function definition for CS110 Assignment', duration =
90 mins.
           This task depends on others!
These are the dependencies: [2]
  'Record CS110 Videos', duration = 75 mins.
           This task depends on others!
```

```
These are the dependencies: [5, 10]
  'CS111 Class', duration = 90 mins.
           This task depends on others!
These are the dependencies: [4]
  'Bike back to 9Floor after the library closes at 22h', duration = 12 mins.
           This task depends on others!
These are the dependencies: [3, 6, 7, 9]
  'Work on written parts of CS110 Assignment', duration = 120 mins.
           This task depends on others!
These are the dependencies: [6]
  'Prepare video scripts and whiteboards for CS110 Assignment', duration = 80
mins.
           This task depends on others!
These are the dependencies: [2]
Running a simple scheduler:

This is my priority queue:
0) Task 0: Get up at 8:00 AM
        Duration: 30 minutes
        Status: I
        Priority Score = 70

After adding tasks to the queue:
 t=8h00: started 'Get up at 8:00 AM' for 30 mins…
          t=8h30, task completed!

This is my priority queue:
0) Task 1: Walk to 7-11 and have breakfast
        Duration: 20 minutes
        Status: I
        Priority Score = 80

After adding tasks to the queue:
 t=8h30: started 'Walk to 7-11 and have breakfast' for 20 mins…
          t=8h50, task completed!

This is my priority queue:
0) Task 2: Bike to the Main NTU Library
        Duration: 12 minutes
        Status: I
        Priority Score = 88

After adding tasks to the queue:
 t=8h50: started 'Bike to the Main NTU Library' for 12 mins…
          t=9h02, task completed!

This is my priority queue:
0) Task 3: Have lunch at JJ's Poke Bowl and bike back to the library
```

```
        Duration: 75 minutes
        Status: I
        Priority Score = 2405

1) Task 4: Do Pre-Class Work for CS111
        Duration: 50 minutes
        Status: I
        Priority Score = 50

2) Task 5: Formalize the Utility function definition for CS110 Assignment
        Duration: 90 minutes
        Status: I
        Priority Score = 10

3) Task 10: Prepare video scripts and whiteboards for CS110 Assignment
        Duration: 80 minutes
        Status: I
        Priority Score = 20

After adding tasks to the queue:
  Fill-in task available! t=9h02: started 'Do Pre-Class Work for CS111' for 50
mins…
          t=9h52, task completed!

  Fill-in task available! t=9h52: started 'Prepare video scripts and whiteboards
for CS110 Assignment' for 80 mins…
          t=11h12, task completed!

  Fill-in task available! t=11h12: started 'Formalize the Utility function
definition for CS110 Assignment' for 90 mins…
          t=12h42, task completed!

  t=12h42: No eligible fill-in tasks available, waiting…

  t=12h43: No eligible fill-in tasks available, waiting…

  t=12h44: No eligible fill-in tasks available, waiting…

  t=12h45: No eligible fill-in tasks available, waiting…

  t=12h46: No eligible fill-in tasks available, waiting…

  t=12h47: No eligible fill-in tasks available, waiting…

  t=12h48: No eligible fill-in tasks available, waiting…

  t=12h49: No eligible fill-in tasks available, waiting…
```

```
  t=12h50: No eligible fill-in tasks available, waiting…

  t=12h51: No eligible fill-in tasks available, waiting…

  t=12h52: No eligible fill-in tasks available, waiting…

  t=12h53: No eligible fill-in tasks available, waiting…

  t=12h54: No eligible fill-in tasks available, waiting…

  t=12h55: No eligible fill-in tasks available, waiting…

  t=12h56: No eligible fill-in tasks available, waiting…

  t=12h57: No eligible fill-in tasks available, waiting…

  t=12h58: No eligible fill-in tasks available, waiting…

  t=12h59: No eligible fill-in tasks available, waiting…

 t=13h00: started 'Have lunch at JJ's Poke Bowl and bike back to the library'
for 75 mins…
          t=14h15, task completed!

This is my priority queue:
0) Task 7: CS111 Class
        Duration: 90 minutes
        Status: I
        Priority Score = 2260

1) Task 6: Record CS110 Videos
        Duration: 75 minutes
        Status: I
        Priority Score = 25

After adding tasks to the queue:
  Fill-in task available! t=14h15: started 'Record CS110 Videos' for 75 mins…
          t=15h30, task completed!

  t=15h30: No eligible fill-in tasks available, waiting…

  t=15h31: No eligible fill-in tasks available, waiting…

  t=15h32: No eligible fill-in tasks available, waiting…

  t=15h33: No eligible fill-in tasks available, waiting…

  t=15h34: No eligible fill-in tasks available, waiting…
```

```
t=15h35: No eligible fill-in tasks available, waiting…
t=15h36: No eligible fill-in tasks available, waiting…
t=15h37: No eligible fill-in tasks available, waiting…
t=15h38: No eligible fill-in tasks available, waiting…
t=15h39: No eligible fill-in tasks available, waiting…
t=15h40: No eligible fill-in tasks available, waiting…
t=15h41: No eligible fill-in tasks available, waiting…
t=15h42: No eligible fill-in tasks available, waiting…
t=15h43: No eligible fill-in tasks available, waiting…
t=15h44: No eligible fill-in tasks available, waiting…
t=15h45: No eligible fill-in tasks available, waiting…
t=15h46: No eligible fill-in tasks available, waiting…
t=15h47: No eligible fill-in tasks available, waiting…
t=15h48: No eligible fill-in tasks available, waiting…
t=15h49: No eligible fill-in tasks available, waiting…
t=15h50: No eligible fill-in tasks available, waiting…
t=15h51: No eligible fill-in tasks available, waiting…
t=15h52: No eligible fill-in tasks available, waiting…
t=15h53: No eligible fill-in tasks available, waiting…
t=15h54: No eligible fill-in tasks available, waiting…
t=15h55: No eligible fill-in tasks available, waiting…
t=15h56: No eligible fill-in tasks available, waiting…
t=15h57: No eligible fill-in tasks available, waiting…
t=15h58: No eligible fill-in tasks available, waiting…
```

```
t=15h59: No eligible fill-in tasks available, waiting…
t=16h00: No eligible fill-in tasks available, waiting…
t=16h01: No eligible fill-in tasks available, waiting…
t=16h02: No eligible fill-in tasks available, waiting…
t=16h03: No eligible fill-in tasks available, waiting…
t=16h04: No eligible fill-in tasks available, waiting…
t=16h05: No eligible fill-in tasks available, waiting…
t=16h06: No eligible fill-in tasks available, waiting…
t=16h07: No eligible fill-in tasks available, waiting…
t=16h08: No eligible fill-in tasks available, waiting…
t=16h09: No eligible fill-in tasks available, waiting…
t=16h10: No eligible fill-in tasks available, waiting…
t=16h11: No eligible fill-in tasks available, waiting…
t=16h12: No eligible fill-in tasks available, waiting…
t=16h13: No eligible fill-in tasks available, waiting…
t=16h14: No eligible fill-in tasks available, waiting…
t=16h15: No eligible fill-in tasks available, waiting…
t=16h16: No eligible fill-in tasks available, waiting…
t=16h17: No eligible fill-in tasks available, waiting…
t=16h18: No eligible fill-in tasks available, waiting…
t=16h19: No eligible fill-in tasks available, waiting…
t=16h20: No eligible fill-in tasks available, waiting…
t=16h21: No eligible fill-in tasks available, waiting…
t=16h22: No eligible fill-in tasks available, waiting…
```

```
t=16h23: No eligible fill-in tasks available, waiting…
t=16h24: No eligible fill-in tasks available, waiting…
t=16h25: No eligible fill-in tasks available, waiting…
t=16h26: No eligible fill-in tasks available, waiting…
t=16h27: No eligible fill-in tasks available, waiting…
t=16h28: No eligible fill-in tasks available, waiting…
t=16h29: No eligible fill-in tasks available, waiting…
t=16h30: No eligible fill-in tasks available, waiting…
t=16h31: No eligible fill-in tasks available, waiting…
t=16h32: No eligible fill-in tasks available, waiting…
t=16h33: No eligible fill-in tasks available, waiting…
t=16h34: No eligible fill-in tasks available, waiting…
t=16h35: No eligible fill-in tasks available, waiting…
t=16h36: No eligible fill-in tasks available, waiting…
t=16h37: No eligible fill-in tasks available, waiting…
t=16h38: No eligible fill-in tasks available, waiting…
t=16h39: No eligible fill-in tasks available, waiting…
t=16h40: No eligible fill-in tasks available, waiting…
t=16h41: No eligible fill-in tasks available, waiting…
t=16h42: No eligible fill-in tasks available, waiting…
t=16h43: No eligible fill-in tasks available, waiting…
t=16h44: No eligible fill-in tasks available, waiting…
t=16h45: No eligible fill-in tasks available, waiting…
t=16h46: No eligible fill-in tasks available, waiting…
```

```
t=16h47: No eligible fill-in tasks available, waiting…
t=16h48: No eligible fill-in tasks available, waiting…
t=16h49: No eligible fill-in tasks available, waiting…
t=16h50: No eligible fill-in tasks available, waiting…
t=16h51: No eligible fill-in tasks available, waiting…
t=16h52: No eligible fill-in tasks available, waiting…
t=16h53: No eligible fill-in tasks available, waiting…
t=16h54: No eligible fill-in tasks available, waiting…
t=16h55: No eligible fill-in tasks available, waiting…
t=16h56: No eligible fill-in tasks available, waiting…
t=16h57: No eligible fill-in tasks available, waiting…
t=16h58: No eligible fill-in tasks available, waiting…
t=16h59: No eligible fill-in tasks available, waiting…
t=17h00: No eligible fill-in tasks available, waiting…
t=17h01: No eligible fill-in tasks available, waiting…
t=17h02: No eligible fill-in tasks available, waiting…
t=17h03: No eligible fill-in tasks available, waiting…
t=17h04: No eligible fill-in tasks available, waiting…
t=17h05: No eligible fill-in tasks available, waiting…
t=17h06: No eligible fill-in tasks available, waiting…
t=17h07: No eligible fill-in tasks available, waiting…
t=17h08: No eligible fill-in tasks available, waiting…
t=17h09: No eligible fill-in tasks available, waiting…
t=17h10: No eligible fill-in tasks available, waiting…
```

```
t=17h11: No eligible fill-in tasks available, waiting…
t=17h12: No eligible fill-in tasks available, waiting…
t=17h13: No eligible fill-in tasks available, waiting…
t=17h14: No eligible fill-in tasks available, waiting…
t=17h15: No eligible fill-in tasks available, waiting…
t=17h16: No eligible fill-in tasks available, waiting…
t=17h17: No eligible fill-in tasks available, waiting…
t=17h18: No eligible fill-in tasks available, waiting…
t=17h19: No eligible fill-in tasks available, waiting…
t=17h20: No eligible fill-in tasks available, waiting…
t=17h21: No eligible fill-in tasks available, waiting…
t=17h22: No eligible fill-in tasks available, waiting…
t=17h23: No eligible fill-in tasks available, waiting…
t=17h24: No eligible fill-in tasks available, waiting…
t=17h25: No eligible fill-in tasks available, waiting…
t=17h26: No eligible fill-in tasks available, waiting…
t=17h27: No eligible fill-in tasks available, waiting…
t=17h28: No eligible fill-in tasks available, waiting…
t=17h29: No eligible fill-in tasks available, waiting…
t=17h30: No eligible fill-in tasks available, waiting…
t=17h31: No eligible fill-in tasks available, waiting…
t=17h32: No eligible fill-in tasks available, waiting…
t=17h33: No eligible fill-in tasks available, waiting…
t=17h34: No eligible fill-in tasks available, waiting…
```

```
t=17h35: No eligible fill-in tasks available, waiting…
t=17h36: No eligible fill-in tasks available, waiting…
t=17h37: No eligible fill-in tasks available, waiting…
t=17h38: No eligible fill-in tasks available, waiting…
t=17h39: No eligible fill-in tasks available, waiting…
t=17h40: No eligible fill-in tasks available, waiting…
t=17h41: No eligible fill-in tasks available, waiting…
t=17h42: No eligible fill-in tasks available, waiting…
t=17h43: No eligible fill-in tasks available, waiting…
t=17h44: No eligible fill-in tasks available, waiting…
t=17h45: No eligible fill-in tasks available, waiting…
t=17h46: No eligible fill-in tasks available, waiting…
t=17h47: No eligible fill-in tasks available, waiting…
t=17h48: No eligible fill-in tasks available, waiting…
t=17h49: No eligible fill-in tasks available, waiting…
t=17h50: No eligible fill-in tasks available, waiting…
t=17h51: No eligible fill-in tasks available, waiting…
t=17h52: No eligible fill-in tasks available, waiting…
t=17h53: No eligible fill-in tasks available, waiting…
t=17h54: No eligible fill-in tasks available, waiting…
t=17h55: No eligible fill-in tasks available, waiting…
t=17h56: No eligible fill-in tasks available, waiting…
t=17h57: No eligible fill-in tasks available, waiting…
t=17h58: No eligible fill-in tasks available, waiting…
```

```
 t=17h59: No eligible fill-in tasks available, waiting…

t=18h00: started 'CS111 Class' for 90 mins…
         t=19h30, task completed!

This is my priority queue:
0) Task 9: Work on written parts of CS110 Assignment
        Duration: 120 minutes
        Status: I
        Priority Score = 20

After adding tasks to the queue:
t=19h30: started 'Work on written parts of CS110 Assignment' for 120 mins…
         t=21h30, task completed!

This is my priority queue:
0) Task 8: Bike back to 9Floor after the library closes at 22h
        Duration: 12 minutes
        Status: I
        Priority Score = 88

After adding tasks to the queue:
t=21h30: started 'Bike back to 9Floor after the library closes at 22h' for 12
mins…
         t=21h42, task completed!


 Completed all planned tasks in 13h42min!
```

5 My Performed Schedule

[93]:
```
tasks = [
Task(id=0, description='Get up at 9:00 AM',
     duration=30, dependencies=[]),
Task(id=1, description='Get ready for the day (organize backpack)',
     duration=15, dependencies=[0]),
Task(id=2, description='Bike to the Main NTU Library',
     duration=15, dependencies=[0, 1]),
Task(id=3, description='Do Pre-Class Work for CS111',
     duration=75, dependencies=[2]),
Task(id=4, description='Formalize the utility function definition for CS110␣
 ↪Assignment',
     duration=60, dependencies=[2]),
Task(id=5, description='Prepare video scripts and whiteboards for CS110␣
 ↪Assignment',
```

```
    duration=75, dependencies=[2]),
Task(id=6, description='Bike and have lunch at JJs Poke Bowl',
    duration=75, dependencies=[2], scheduled_time=14*60),
Task(id=7, description='CS111 Class',
    duration=90, dependencies=[3, 6], scheduled_time=18*60),
Task(id=8, description='Fix bugs in the additional fill-task methods for CS110␣
 ↪Assignment',
    duration=90, dependencies=[5]),
Task(id=9, description='Work on LBA part of the upcoming CS111 Assignment',
    duration=60, dependencies=[3]),
Task(id=9, description='Bike back from JJs to NTU Library before CS111 class',
    duration=10, dependencies=[6]),
Task(id=10, description='Bike back to Res Hall after class finishes at 7:30pm',
    duration=12, dependencies=[6, 7, 8, 9], scheduled_time=1170) # 19.5*60␣
 ↪expressed as an integer
]

task_scheduler = TaskScheduler(tasks)

task_scheduler.print_self()

start_scheduler_at = 9*60
task_scheduler.run_task_scheduler(start_scheduler_at)
task_scheduler.print_executed_task_descriptions()
```

```
Tasks added to the simple scheduler:
-------------------------------------
  'Get up at 9:00 AM', duration = 30 mins.
  'Get ready for the day (organize backpack)', duration = 15 mins.
          This task depends on others!
These are the dependencies: [0]
  'Bike to the Main NTU Library', duration = 15 mins.
          This task depends on others!
These are the dependencies: [0, 1]
  'Do Pre-Class Work for CS111', duration = 75 mins.
          This task depends on others!
These are the dependencies: [2]
  'Formalize the utility function definition for CS110 Assignment', duration =
60 mins.
          This task depends on others!
These are the dependencies: [2]
  'Prepare video scripts and whiteboards for CS110 Assignment', duration = 75
mins.
          This task depends on others!
These are the dependencies: [2]
  'Bike and have lunch at JJs Poke Bowl', duration = 75 mins.
          This task depends on others!
```

```
These are the dependencies: [2]
  'CS111 Class', duration = 90 mins.
          This task depends on others!
These are the dependencies: [3, 6]
  'Fix bugs in the additional fill-task methods for CS110 Assignment', duration
= 90 mins.
          This task depends on others!
These are the dependencies: [5]
  'Work on LBA part of the upcoming CS111 Assignment', duration = 60 mins.
          This task depends on others!
These are the dependencies: [3]
  'Bike back from JJs to NTU Library before CS111 class', duration = 10 mins.
          This task depends on others!
These are the dependencies: [6]
  'Bike back to Res Hall after class finishes at 7:30pm', duration = 12 mins.
          This task depends on others!
These are the dependencies: [6, 7, 8, 9]
Running a simple scheduler:

 t=9h00: started 'Get up at 9:00 AM' for 30 mins…
          t=9h30, task completed!

 t=9h30: started 'Get ready for the day (organize backpack)' for 15 mins…
          t=9h45, task completed!

 t=9h45: started 'Bike to the Main NTU Library' for 15 mins…
          t=10h00, task completed!

  Fill-in task available! t=10h00: started 'Formalize the utility function
definition for CS110 Assignment' for 60 mins…
          t=11h00, task completed!

  Fill-in task available! t=11h00: started 'Prepare video scripts and
whiteboards for CS110 Assignment' for 75 mins…
          t=12h15, task completed!

  Fill-in task available! t=12h15: started 'Do Pre-Class Work for CS111' for 75
mins…
          t=13h30, task completed!

  t=13h30: No eligible fill-in tasks available, waiting…

  t=13h31: No eligible fill-in tasks available, waiting…

  t=13h32: No eligible fill-in tasks available, waiting…

  t=13h33: No eligible fill-in tasks available, waiting…
```

```
t=13h34: No eligible fill-in tasks available, waiting…
t=13h35: No eligible fill-in tasks available, waiting…
t=13h36: No eligible fill-in tasks available, waiting…
t=13h37: No eligible fill-in tasks available, waiting…
t=13h38: No eligible fill-in tasks available, waiting…
t=13h39: No eligible fill-in tasks available, waiting…
t=13h40: No eligible fill-in tasks available, waiting…
t=13h41: No eligible fill-in tasks available, waiting…
t=13h42: No eligible fill-in tasks available, waiting…
t=13h43: No eligible fill-in tasks available, waiting…
t=13h44: No eligible fill-in tasks available, waiting…
t=13h45: No eligible fill-in tasks available, waiting…
t=13h46: No eligible fill-in tasks available, waiting…
t=13h47: No eligible fill-in tasks available, waiting…
t=13h48: No eligible fill-in tasks available, waiting…
t=13h49: No eligible fill-in tasks available, waiting…
t=13h50: No eligible fill-in tasks available, waiting…
t=13h51: No eligible fill-in tasks available, waiting…
t=13h52: No eligible fill-in tasks available, waiting…
t=13h53: No eligible fill-in tasks available, waiting…
t=13h54: No eligible fill-in tasks available, waiting…
t=13h55: No eligible fill-in tasks available, waiting…
t=13h56: No eligible fill-in tasks available, waiting…
t=13h57: No eligible fill-in tasks available, waiting…
```

```
  t=13h58: No eligible fill-in tasks available, waiting…

  t=13h59: No eligible fill-in tasks available, waiting…

 t=14h00: started 'Bike and have lunch at JJs Poke Bowl' for 75 mins…
          t=15h15, task completed!

  Fill-in task available! t=15h15: started 'Bike back from JJs to NTU Library
before CS111 class' for 10 mins…
          t=15h25, task completed!

  Fill-in task available! t=15h25: started 'Work on LBA part of the upcoming
CS111 Assignment' for 60 mins…
          t=16h25, task completed!

  Fill-in task available! t=16h25: started 'Fix bugs in the additional fill-task
methods for CS110 Assignment' for 90 mins…
          t=17h55, task completed!

  t=17h55: No eligible fill-in tasks available, waiting…

  t=17h56: No eligible fill-in tasks available, waiting…

  t=17h57: No eligible fill-in tasks available, waiting…

  t=17h58: No eligible fill-in tasks available, waiting…

  t=17h59: No eligible fill-in tasks available, waiting…

 t=18h00: started 'CS111 Class' for 90 mins…
          t=19h30, task completed!

 t=19h30: started 'Bike back to Res Hall after class finishes at 7:30pm' for 12
mins…
          t=19h42, task completed!


  Completed all planned tasks in 10h42min!

Final descriptions of executed tasks:
-------------------------------------
1. Get up at 9:00 AM
2. Get ready for the day (organize backpack)
3. Bike to the Main NTU Library
4. Formalize the utility function definition for CS110 Assignment
5. Prepare video scripts and whiteboards for CS110 Assignment
6. Do Pre-Class Work for CS111
7. Bike and have lunch at JJs Poke Bowl
```

```
8. Bike back from JJs to NTU Library before CS111 class
9. Work on LBA part of the upcoming CS111 Assignment
10. Fix bugs in the additional fill-task methods for CS110 Assignment
11. CS111 Class
12. Bike back to Res Hall after class finishes at 7:30pm
```

6 Efficiency Analysis

6.1 Quantitative Analysis

References

Geeks for Geeks: Log Log function in Python https://www.geeksforgeeks.org/matplotlib-pyplot-loglog-function-in-python/

Statology: Scatterplot with regression line in Python https://www.statology.org/scatterplot-with-regression-line-python/

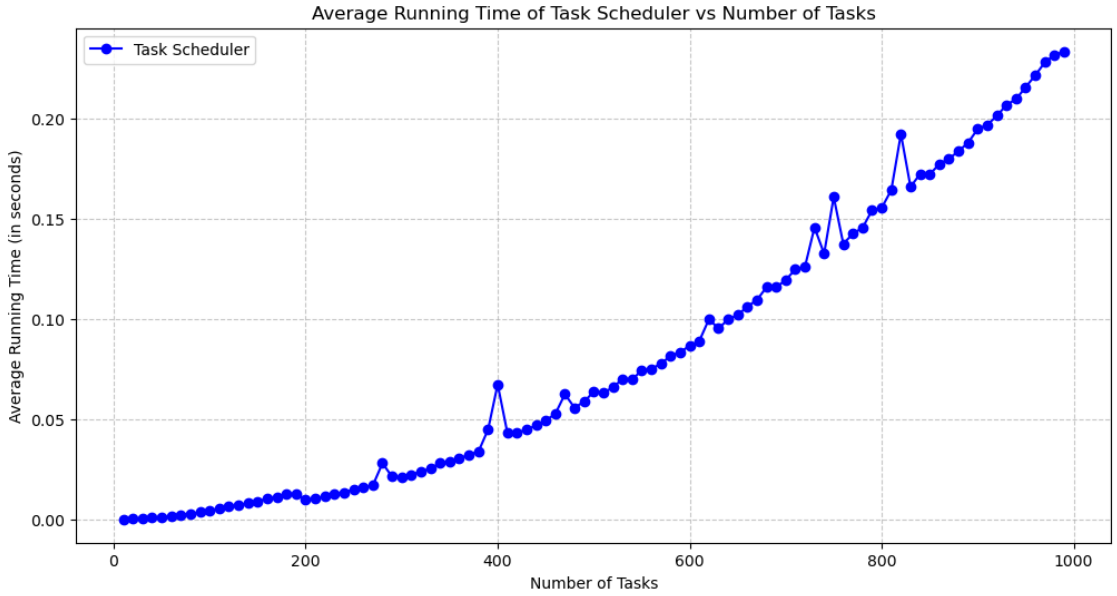
[5]:
```python
import pandas as pd
import numpy as np
import random
import time

def generate_tasks(num_tasks):
    """
    Generates a list of randomly created tasks.

    Args:
        num_tasks (int): The number of tasks to generate.

    Returns:
        list: A list of Task instances with random durations and no dependencies.
    """
    tasks = []  # Initialize an empty list to hold the tasks

    # Loop to create the specified number of tasks
    for i in range(num_tasks):
        duration = random.randint(1, 60)  # Random duration between 1 and 60 minutes
        dependencies = []  # Consider all tasks as independent (no dependencies)

        # Create a new Task instance with generated parameters
        task = Task(id=i, description=f"Task {i}", duration=duration, dependencies=dependencies)

        # Append the newly created task to the task list
        tasks.append(task)
```

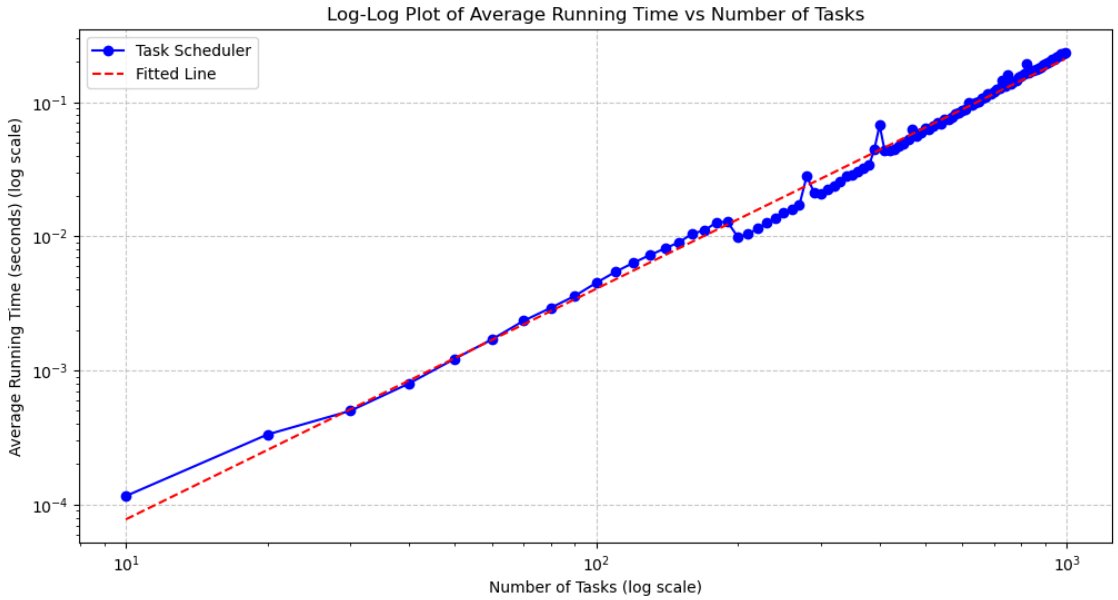
```
    return tasks  # Return the list of generated tasks

def run_experiments_avg_running_time(input_sizes, trials=30):
    """
    Runs experiments to measure the average running time of the task scheduler.

    Args:
        input_sizes (list): A list of integers representing different sizes of␣
 ↪task inputs.
        trials (int): The number of trials to run for each input size (default␣
 ↪is 30).

    Returns:
        list: A list containing average running times for each input size.
    """
    experimental_results = []  # List to store average running times for each␣
 ↪input size

    # Loop through each specified input size
    for size in input_sizes:
        average_time = 0

        # Perform multiple trials for the current input size
        for _ in range(trials):
            tasks = generate_tasks(size)  # Generate a new set of tasks
            scheduler = TaskScheduler(tasks)  # Initialize the task scheduler␣
 ↪with generated tasks

            start_time = time.process_time()  # Start timing
            scheduler.run_task_scheduler(starting_time=8*60)
            end_time = time.process_time()  # End timing

            elapsed = end_time - start_time  # Calculate elapsed time for this␣
 ↪trial
            average_time += elapsed  # Accumulate elapsed time

        average_time /= trials  # Calculate the average over all trials
        experimental_results.append(average_time)

    return experimental_results
```

[7]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Parameters for the experiment
```

```
num_tasks_list = [x for x in range(10, 1000, 10)]  # Number of tasks to test

# Run experiments and collect results
results = run_experiments_avg_running_time(num_tasks_list)

# Calculate slopes between points
slopes = []
for i in range(1, len(results)):
    slope = (results[i] - results[i-1]) / (num_tasks_list[i] -␣
 ↪num_tasks_list[i-1])
    slopes.append(slope)

# Prepare data for table display
table_data = list(zip(num_tasks_list[1:], results[1:], slopes))

# Create a figure for the main plot
fig, ax = plt.subplots(figsize=(12, 6))

# Plotting results in linear scale
ax.plot(num_tasks_list, results, label='Task Scheduler', color='blue',␣
 ↪marker='o')
ax.set_title('Average Running Time of Task Scheduler vs Number of Tasks')
ax.set_xlabel('Number of Tasks')
ax.set_ylabel('Average Running Time (in seconds)')
ax.legend()
ax.grid(True, linestyle='--', alpha=0.7)

# Create a log-log plot
plt.figure(figsize=(12, 6))
plt.loglog(num_tasks_list, results, label='Task Scheduler', color='blue',␣
 ↪marker='o')

# Fit a line to the log-log data
X = np.log(np.array(num_tasks_list)).reshape(-1, 1)  # reshape for sklearn
y = np.log(np.array(results))

model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)

# Plot the fitted line on the log-log plot
plt.loglog(num_tasks_list, np.exp(y_pred), color='red', linestyle='--',␣
 ↪label='Fitted Line')

# Get the slope from the linear regression model (the coefficient)
slope = model.coef_[0]
intercept = model.intercept_
```

```
plt.title('Log-Log Plot of Average Running Time vs Number of Tasks')
plt.xlabel('Number of Tasks (log scale)')
plt.ylabel('Average Running Time (seconds) (log scale)')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()

# Print the slope of the fitted line
print(f"Slope of the fitted line: {slope:.4f}")
print(f"Intercept of the fitted line: {intercept:.4f}")
```






```
Slope of the fitted line: 1.7195
Intercept of the fitted line: -13.4235
```

6.2 Qualitative Analysis

Introducing the 12-hour constraint, and delayed tasks.

[59]:
```python
class Task:
    """
    Represents a task with its details and current status.

    Attributes:
        id (int): Unique identifier for the task.
        description (str): A brief description of the task.
        duration (int): Duration of the task in minutes.
        dependencies (list): List of task IDs that must be completed before␣
↪this task.
        scheduled_time (int, optional): The specific time (in minutes from the␣
↪start of the day)
                                        when the task is scheduled to start.␣
↪Default is None.
        status (str): Current status of the task. Possible values:
                      - 'N' (NOT_STARTED): The task has not been started.
                      - 'I' (IN_PRIORITY_QUEUE): The task is in the priority␣
↪queue.
                      - 'C' (COMPLETED): The task has been completed.
                      - 'D' (DELAYED): The task has been delayed.
```

```
        priority_score (int): A score representing the urgency or importance of␣
↪the task.
                              Higher values indicate higher priority.
    """

    # Status constants for clarity and consistency
    NOT_STARTED = 'N'
    IN_PRIORITY_QUEUE = 'I'
    COMPLETED = 'C'
    DELAYED = 'D'

    def __init__(self, id, description, duration, dependencies,␣
↪scheduled_time=None, status='N', priority_score=0):
        """
        Initializes a Task instance.

        """
        self.id = id
        self.description = description
        self.duration = duration
        self.dependencies = dependencies
        self.scheduled_time = scheduled_time
        self.status = status
        self.priority_score = priority_score

    def __str__(self):
        """
        Returns a human-readable string representation of the Task.

        Returns:
            str: A string describing the task, including its ID, description,
                 duration, status, and priority score.
        """
        return (
            f"Task {self.id}: {self.description}\n"
            f"\tDuration: {self.duration} minutes\n"
            f"\tStatus: {self.status}\n"
            f"\tPriority Score = {self.priority_score}\n"
        )

    def compute_priority(self, current_time):
        """
        Computes and updates the task's priority score based on its attributes␣
↪and the current time.

        Priority is calculated using the following factors:
```

```
        - Scheduled Time: Tasks closer to their scheduled time have higher␣
↪priority.
        - Dependencies: Tasks with many dependencies have slightly reduced␣
↪priority.
        - Duration: Shorter tasks may be prioritized for scheduling efficiency.

        Args:
            current_time (int): The current time (in minutes from the start of␣
↪the day).
        """
        # Determine time priority based on the scheduled time of the task
        time_priority = 0 # Default
        if self.scheduled_time is not None:  # Check if the task has a␣
↪predefined scheduled time
            if self.scheduled_time == current_time:
                time_priority = 100  # Assign maximum priority if the task is␣
↪scheduled for the current time
            elif self.scheduled_time < current_time:
                time_priority = 0  # Task loses priority if its scheduled time␣
↪has already passed
            else:
                # Increase priority as the task's scheduled time approaches the␣
↪current time
                time_priority = (self.scheduled_time - current_time) * 10

        # Penalize tasks with many dependencies
        dependency_score = len(self.dependencies)

        # Total priority score calculation
        total_priority = (
            time_priority -  # Directly use time_priority, which can be positive
            dependency_score + # Dependence score contributes negatively to the␣
↪utility
            abs(100 - self.duration)  # Longer tasks have lower scores
        )

        self.priority_score = total_priority

    def __lt__(self, other):
        """
        Compares tasks for sorting, based on their priority score.

        Args:
            other (Task): Another Task instance to compare with.

        Returns:
```

```
            bool: True if this task has a lower priority score than the other␣
↪task.
        """
        return self.priority_score < other.priority_score

    def to_dict(self):
        """
        Converts the Task instance into a dictionary representation.

        Returns:
            dict: A dictionary containing the task's key attributes:
                  - id (int): Task ID.
                  - title (str): Task description.
                  - duration (int): Task duration in minutes.
                  - dependency (list): List of task dependencies.
        """
        return {
            "id": self.id,
            "title": self.description,
            "duration": self.duration,
            "dependency": self.dependencies,
        }
```

[62]:
```
class TaskScheduler:
    """
    A simple daily task scheduler using a priority queue.

    Attributes:
        tasks (list): List of tasks to be managed by the scheduler.
        priority_queue (MaxHeapq): Priority queue for managing task execution␣
↪order.
    """

    NOT_STARTED = 'N'  # Task has not been started
    IN_PRIORITY_QUEUE = 'I'  # Task is in the priority queue
    COMPLETED = 'C'  # Task has been completed
    DELAYED = 'D'  # Task has been delayed

    def __init__(self, tasks):
        """
        Initializes an instance of TaskScheduler.

        Args:
            tasks (list): List of Task instances to schedule.
        """
        self.tasks = tasks  # Store the list of tasks
        self.priority_queue = MaxHeapq()  # Initialize the priority queue
```

```
        self.executed_task_descriptions = []  # Stores descriptions of executed␣
↪tasks

    def print_self(self):
        """Prints all tasks added to the scheduler with their details."""
        print("Tasks added to the simple scheduler:")
        print("---------------------------------------")

        for t in self.tasks:
            print(f"  '{t.description}', duration = {t.duration} mins.")
            if len(t.dependencies) > 0:
                print(f"\t  This task depends on others!")
                print(f"These are the dependencies: {t.dependencies}")

    def print_priority_queue(self):
        """Prints the current state of the priority queue."""
        print('This is my priority queue: ')

        for order, task in enumerate(self.priority_queue.heap):
            print(f'{order}) {task}')

    def remove_dependency(self, id):
        """
        Removes a specified task ID from the dependencies of other tasks.

        Args:
            id (int): The ID of the task whose dependencies should be removed.
        """
        for t in self.tasks:
            if t.id != id and id in t.dependencies:
                t.dependencies.remove(id)

    def get_tasks_ready(self, current_time):
        """
        Updates the status and computes priorities for tasks that are ready to␣
↪execute.

        Args:
            current_time (int): The current time used to check readiness.
        """

        for task in self.tasks:
            # If the task has no dependencies and is not yet in the queue
            if task.status == self.NOT_STARTED and not task.dependencies:
                # Update status of the task
                task.status = self.IN_PRIORITY_QUEUE
```

```
                task.compute_priority(current_time)  # Compute priority based␣
↪on current time
                # Push task into the priority queue
                self.priority_queue.heappush(task)

    def check_unscheduled_tasks(self):
        """Checks if there are any unscheduled tasks remaining."""

        for task in self.tasks:
            if task.status == self.NOT_STARTED:  # If any task is not started,␣
↪return True
                return True

        return False  # No unscheduled tasks

    def format_time(self, time):
        """Formats time from minutes into hours and minutes string format."""

        return f"{time // 60}h{time % 60:02d}"

    def get_tasks_delayed(self):
        """
        Marks all tasks as delayed that are not completed or already in progress,
        and prints them out.

        This method is called when the day ends without completing all tasks.
        """

        # Change all relevant task statuses to delayed
        for task in self.tasks:
            if task.status == self.IN_PRIORITY_QUEUE or task.status == self.
↪NOT_STARTED:
                task.status = self.DELAYED

        print('  The day has ended! The following tasks have been delayed for␣
↪tomorrow:\n')
        i = 1
        for task in self.tasks:
            if task.status == self.DELAYED:
                print(f'{i}) {task}')
                i += 1

    def validate_heap(self):
        """Validates that the priority queue maintains its max-heap property."""
        assert self.priority_queue.is_max_heap(), "Heap property violated!"

    def run_task_scheduler(self, starting_time):
```

```
        """
        Executes the scheduler starting from a specified time.

        Args:
            starting_time (int): The initial time from which scheduling begins.
        """
        day_ended = False  # Flag to indicate if the day has ended
        current_time = starting_time  # Initialize current time
        print("Running a simple scheduler:\n")

        while self.check_unscheduled_tasks() or self.priority_queue.heap:
            self.get_tasks_ready(current_time)  # Prepare tasks ready to execute
            #self.print_priority_queue() Uncommment to see the priority queue
            print("After adding tasks to the queue:")

            if self.priority_queue.heap_size > 0:  # If there are tasks in the␣
↪queue
                current_task = self.priority_queue.heappop()  # Get highest␣
↪priority task
                self.validate_heap()  # Validate heap property

                scheduled_time = current_task.scheduled_time

                if scheduled_time is not None and scheduled_time > current_time:
                    wait_until = scheduled_time

                    while current_time < wait_until:  # Wait until scheduled␣
↪time
                        fill_in_executed = False

                        for current_index in range(self.priority_queue.
↪heap_size):
                            peek_fill_in_task = self.priority_queue.
↪heap[current_index]

                            if peek_fill_in_task.scheduled_time is None or␣
↪peek_fill_in_task.scheduled_time < scheduled_time:
                                if current_time + peek_fill_in_task.duration <=␣
↪wait_until:
                                    fill_in_task = self.priority_queue.
↪anypop(current_index)
                                    self.validate_heap()

                                    print(f" Fill-in task available! t={self.
↪format_time(current_time)}: started '{fill_in_task.description}' for␣
↪{fill_in_task.duration} mins...")
```

```
                                    current_time += fill_in_task.duration  #␣
↪Update current time after executing fill-in task
                                    print(f"\t  t={self.
↪format_time(current_time)}, task completed!\n")
                                    self.remove_dependency(fill_in_task.id)  #␣
↪Remove completed dependencies
                                    fill_in_task.status = fill_in_task.
↪COMPLETED  # Mark as completed
                                    self.executed_task_descriptions.
↪append(fill_in_task.description)  # Log description
                                    fill_in_executed = True
                                    break

                        if not fill_in_executed:  # If no fill-in tasks were␣
↪executed, wait longer
                            print(f"  t={self.format_time(current_time)}: No␣
↪eligible fill-in tasks available, waiting...\n")
                            current_time += 1

                print(f" t={self.format_time(current_time)}: started␣
↪'{current_task.description}' for {current_task.duration} mins...")
                current_time += current_task.duration  # Update time after␣
↪executing main task
                print(f"\t  t={self.format_time(current_time)}, task completed!
↪\n")
                self.remove_dependency(current_task.id)  # Remove completed␣
↪dependencies
                current_task.status = current_task.COMPLETED  # Mark as␣
↪completed
                self.executed_task_descriptions.append(current_task.
↪description)  # Store description

                total_time_so_far = current_time - starting_time

                if total_time_so_far >= 12 * 60:  # End day after working for␣
↪12 hours
                    day_ended = True
                    break

        if day_ended:
            self.get_tasks_delayed()  # Handle any delayed tasks at end of day

        total_time = current_time - starting_time
        print(f"\n Completed all planned tasks in {total_time //␣
↪60}h{total_time % 60:02d}min!")
```

```
    def print_executed_task_descriptions(self):
        """
        Prints the final descriptions of executed tasks as a list of strings.
        """
        print("\nFinal descriptions of executed tasks:")
        print("-------------------------------------")
        for i, description in enumerate(self.executed_task_descriptions,␣
↪start=1):
            print(f"{i}. {description}")
```

[63]:
```
qualitative_test_tasks = [
    # Independent tasks
    Task(id=0, description='Get up and have breakfast', duration=30,␣
↪dependencies=[]),
    Task(id=1, description='Daily Exercise', duration=45, dependencies=[]),

    # Pre-scheduled tasks
    Task(id=2, description='Meeting for AI Sustainability Lab', duration=75,␣
↪dependencies=[0], scheduled_time=10*60),  # Scheduled at 12:00 PM
    Task(id=3, description='CS111 Class', duration=90, dependencies=[0, 4],␣
↪scheduled_time=18*60),  # Scheduled at 6:00 PM

    # Dependent tasks
    Task(id=4, description='Do Pre-Class Work for CS111', duration=120,␣
↪dependencies=[0]),  # Dependent on Breakfast and Meeting
    Task(id=5, description='Revise last CS113 Content', duration=70,␣
↪dependencies=[0, 4]),  # No dependencies
    Task(id=6, description='Online Discussion with CCP Team', duration=60,␣
↪dependencies=[0], scheduled_time=14*60),  # Scheduled at 2:00 PM

    # Time-consuming tasks
    Task(id=7, description='Skill Builder for CS113', duration=150,␣
↪dependencies=[0, 5]),

    # Tasks with fixed deadlines
    Task(id=8, description='Bake Cookies with Friends', duration=120,␣
↪dependencies=[0, 3, 4, 5], scheduled_time=20*60),  # Scheduled at 8:00 PM

    # Another dependent task
    Task(id=9, description='Planning for the Rest of the Week', duration=45,␣
↪dependencies=[0, 8])
]

task_scheduler_qualitative_test = TaskScheduler(qualitative_test_tasks)

task_scheduler_qualitative_test.print_self()
```

```
start_scheduler_at = 9*60
task_scheduler_qualitative_test.run_task_scheduler(start_scheduler_at)
task_scheduler_qualitative_test.print_executed_task_descriptions()
```

```
Tasks added to the simple scheduler:
-------------------------------------
  'Get up and have breakfast', duration = 30 mins.
  'Daily Exercise', duration = 45 mins.
  'Meeting for AI Sustainability Lab', duration = 75 mins.
           This task depends on others!
These are the dependencies: [0]
  'CS111 Class', duration = 90 mins.
           This task depends on others!
These are the dependencies: [0, 4]
  'Do Pre-Class Work for CS111', duration = 120 mins.
           This task depends on others!
These are the dependencies: [0]
  'Revise last CS113 Content', duration = 70 mins.
           This task depends on others!
These are the dependencies: [0, 4]
  'Online Discussion with CCP Team', duration = 60 mins.
           This task depends on others!
These are the dependencies: [0]
  'Skill Builder for CS113', duration = 150 mins.
           This task depends on others!
These are the dependencies: [0, 5]
  'Bake Cookies with Friends', duration = 120 mins.
           This task depends on others!
These are the dependencies: [0, 3, 4, 5]
  'Planning for the Rest of the Week', duration = 45 mins.
           This task depends on others!
These are the dependencies: [0, 8]
Running a simple scheduler:

After adding tasks to the queue:
 t=9h00: started 'Get up and have breakfast' for 30 mins…
          t=9h30, task completed!

After adding tasks to the queue:
  Fill-in task available! t=9h30: started 'Meeting for AI Sustainability Lab'
for 75 mins…
          t=10h45, task completed!

  Fill-in task available! t=10h45: started 'Daily Exercise' for 45 mins…
          t=11h30, task completed!

  Fill-in task available! t=11h30: started 'Do Pre-Class Work for CS111' for 120
```

```
mins…
          t=13h30, task completed!

  t=13h30: No eligible fill-in tasks available, waiting…

  t=13h31: No eligible fill-in tasks available, waiting…

  t=13h32: No eligible fill-in tasks available, waiting…

  t=13h33: No eligible fill-in tasks available, waiting…

  t=13h34: No eligible fill-in tasks available, waiting…

  t=13h35: No eligible fill-in tasks available, waiting…

  t=13h36: No eligible fill-in tasks available, waiting…

  t=13h37: No eligible fill-in tasks available, waiting…

  t=13h38: No eligible fill-in tasks available, waiting…

  t=13h39: No eligible fill-in tasks available, waiting…

  t=13h40: No eligible fill-in tasks available, waiting…

  t=13h41: No eligible fill-in tasks available, waiting…

  t=13h42: No eligible fill-in tasks available, waiting…

  t=13h43: No eligible fill-in tasks available, waiting…

  t=13h44: No eligible fill-in tasks available, waiting…

  t=13h45: No eligible fill-in tasks available, waiting…

  t=13h46: No eligible fill-in tasks available, waiting…

  t=13h47: No eligible fill-in tasks available, waiting…

  t=13h48: No eligible fill-in tasks available, waiting…

  t=13h49: No eligible fill-in tasks available, waiting…

  t=13h50: No eligible fill-in tasks available, waiting…

  t=13h51: No eligible fill-in tasks available, waiting…

  t=13h52: No eligible fill-in tasks available, waiting…
```

```
  t=13h53: No eligible fill-in tasks available, waiting…

  t=13h54: No eligible fill-in tasks available, waiting…

  t=13h55: No eligible fill-in tasks available, waiting…

  t=13h56: No eligible fill-in tasks available, waiting…

  t=13h57: No eligible fill-in tasks available, waiting…

  t=13h58: No eligible fill-in tasks available, waiting…

  t=13h59: No eligible fill-in tasks available, waiting…

 t=14h00: started 'Online Discussion with CCP Team' for 60 mins…
          t=15h00, task completed!

After adding tasks to the queue:
  Fill-in task available! t=15h00: started 'Revise last CS113 Content' for 70
mins…
          t=16h10, task completed!

  t=16h10: No eligible fill-in tasks available, waiting…

  t=16h11: No eligible fill-in tasks available, waiting…

  t=16h12: No eligible fill-in tasks available, waiting…

  t=16h13: No eligible fill-in tasks available, waiting…

  t=16h14: No eligible fill-in tasks available, waiting…

  t=16h15: No eligible fill-in tasks available, waiting…

  t=16h16: No eligible fill-in tasks available, waiting…

  t=16h17: No eligible fill-in tasks available, waiting…

  t=16h18: No eligible fill-in tasks available, waiting…

  t=16h19: No eligible fill-in tasks available, waiting…

  t=16h20: No eligible fill-in tasks available, waiting…

  t=16h21: No eligible fill-in tasks available, waiting…

  t=16h22: No eligible fill-in tasks available, waiting…
```

```
t=16h23: No eligible fill-in tasks available, waiting…
t=16h24: No eligible fill-in tasks available, waiting…
t=16h25: No eligible fill-in tasks available, waiting…
t=16h26: No eligible fill-in tasks available, waiting…
t=16h27: No eligible fill-in tasks available, waiting…
t=16h28: No eligible fill-in tasks available, waiting…
t=16h29: No eligible fill-in tasks available, waiting…
t=16h30: No eligible fill-in tasks available, waiting…
t=16h31: No eligible fill-in tasks available, waiting…
t=16h32: No eligible fill-in tasks available, waiting…
t=16h33: No eligible fill-in tasks available, waiting…
t=16h34: No eligible fill-in tasks available, waiting…
t=16h35: No eligible fill-in tasks available, waiting…
t=16h36: No eligible fill-in tasks available, waiting…
t=16h37: No eligible fill-in tasks available, waiting…
t=16h38: No eligible fill-in tasks available, waiting…
t=16h39: No eligible fill-in tasks available, waiting…
t=16h40: No eligible fill-in tasks available, waiting…
t=16h41: No eligible fill-in tasks available, waiting…
t=16h42: No eligible fill-in tasks available, waiting…
t=16h43: No eligible fill-in tasks available, waiting…
t=16h44: No eligible fill-in tasks available, waiting…
t=16h45: No eligible fill-in tasks available, waiting…
t=16h46: No eligible fill-in tasks available, waiting…
```

```
t=16h47: No eligible fill-in tasks available, waiting…
t=16h48: No eligible fill-in tasks available, waiting…
t=16h49: No eligible fill-in tasks available, waiting…
t=16h50: No eligible fill-in tasks available, waiting…
t=16h51: No eligible fill-in tasks available, waiting…
t=16h52: No eligible fill-in tasks available, waiting…
t=16h53: No eligible fill-in tasks available, waiting…
t=16h54: No eligible fill-in tasks available, waiting…
t=16h55: No eligible fill-in tasks available, waiting…
t=16h56: No eligible fill-in tasks available, waiting…
t=16h57: No eligible fill-in tasks available, waiting…
t=16h58: No eligible fill-in tasks available, waiting…
t=16h59: No eligible fill-in tasks available, waiting…
t=17h00: No eligible fill-in tasks available, waiting…
t=17h01: No eligible fill-in tasks available, waiting…
t=17h02: No eligible fill-in tasks available, waiting…
t=17h03: No eligible fill-in tasks available, waiting…
t=17h04: No eligible fill-in tasks available, waiting…
t=17h05: No eligible fill-in tasks available, waiting…
t=17h06: No eligible fill-in tasks available, waiting…
t=17h07: No eligible fill-in tasks available, waiting…
t=17h08: No eligible fill-in tasks available, waiting…
t=17h09: No eligible fill-in tasks available, waiting…
t=17h10: No eligible fill-in tasks available, waiting…
```

```
t=17h11: No eligible fill-in tasks available, waiting…
t=17h12: No eligible fill-in tasks available, waiting…
t=17h13: No eligible fill-in tasks available, waiting…
t=17h14: No eligible fill-in tasks available, waiting…
t=17h15: No eligible fill-in tasks available, waiting…
t=17h16: No eligible fill-in tasks available, waiting…
t=17h17: No eligible fill-in tasks available, waiting…
t=17h18: No eligible fill-in tasks available, waiting…
t=17h19: No eligible fill-in tasks available, waiting…
t=17h20: No eligible fill-in tasks available, waiting…
t=17h21: No eligible fill-in tasks available, waiting…
t=17h22: No eligible fill-in tasks available, waiting…
t=17h23: No eligible fill-in tasks available, waiting…
t=17h24: No eligible fill-in tasks available, waiting…
t=17h25: No eligible fill-in tasks available, waiting…
t=17h26: No eligible fill-in tasks available, waiting…
t=17h27: No eligible fill-in tasks available, waiting…
t=17h28: No eligible fill-in tasks available, waiting…
t=17h29: No eligible fill-in tasks available, waiting…
t=17h30: No eligible fill-in tasks available, waiting…
t=17h31: No eligible fill-in tasks available, waiting…
t=17h32: No eligible fill-in tasks available, waiting…
t=17h33: No eligible fill-in tasks available, waiting…
t=17h34: No eligible fill-in tasks available, waiting…
```

```
t=17h35: No eligible fill-in tasks available, waiting…
t=17h36: No eligible fill-in tasks available, waiting…
t=17h37: No eligible fill-in tasks available, waiting…
t=17h38: No eligible fill-in tasks available, waiting…
t=17h39: No eligible fill-in tasks available, waiting…
t=17h40: No eligible fill-in tasks available, waiting…
t=17h41: No eligible fill-in tasks available, waiting…
t=17h42: No eligible fill-in tasks available, waiting…
t=17h43: No eligible fill-in tasks available, waiting…
t=17h44: No eligible fill-in tasks available, waiting…
t=17h45: No eligible fill-in tasks available, waiting…
t=17h46: No eligible fill-in tasks available, waiting…
t=17h47: No eligible fill-in tasks available, waiting…
t=17h48: No eligible fill-in tasks available, waiting…
t=17h49: No eligible fill-in tasks available, waiting…
t=17h50: No eligible fill-in tasks available, waiting…
t=17h51: No eligible fill-in tasks available, waiting…
t=17h52: No eligible fill-in tasks available, waiting…
t=17h53: No eligible fill-in tasks available, waiting…
t=17h54: No eligible fill-in tasks available, waiting…
t=17h55: No eligible fill-in tasks available, waiting…
t=17h56: No eligible fill-in tasks available, waiting…
t=17h57: No eligible fill-in tasks available, waiting…
t=17h58: No eligible fill-in tasks available, waiting…
```

```
  t=17h59: No eligible fill-in tasks available, waiting…

 t=18h00: started 'CS111 Class' for 90 mins…
          t=19h30, task completed!

After adding tasks to the queue:
  t=19h30: No eligible fill-in tasks available, waiting…

  t=19h31: No eligible fill-in tasks available, waiting…

  t=19h32: No eligible fill-in tasks available, waiting…

  t=19h33: No eligible fill-in tasks available, waiting…

  t=19h34: No eligible fill-in tasks available, waiting…

  t=19h35: No eligible fill-in tasks available, waiting…

  t=19h36: No eligible fill-in tasks available, waiting…

  t=19h37: No eligible fill-in tasks available, waiting…

  t=19h38: No eligible fill-in tasks available, waiting…

  t=19h39: No eligible fill-in tasks available, waiting…

  t=19h40: No eligible fill-in tasks available, waiting…

  t=19h41: No eligible fill-in tasks available, waiting…

  t=19h42: No eligible fill-in tasks available, waiting…

  t=19h43: No eligible fill-in tasks available, waiting…

  t=19h44: No eligible fill-in tasks available, waiting…

  t=19h45: No eligible fill-in tasks available, waiting…

  t=19h46: No eligible fill-in tasks available, waiting…

  t=19h47: No eligible fill-in tasks available, waiting…

  t=19h48: No eligible fill-in tasks available, waiting…

  t=19h49: No eligible fill-in tasks available, waiting…

  t=19h50: No eligible fill-in tasks available, waiting…
```

```
 t=19h51: No eligible fill-in tasks available, waiting…

 t=19h52: No eligible fill-in tasks available, waiting…

 t=19h53: No eligible fill-in tasks available, waiting…

 t=19h54: No eligible fill-in tasks available, waiting…

 t=19h55: No eligible fill-in tasks available, waiting…

 t=19h56: No eligible fill-in tasks available, waiting…

 t=19h57: No eligible fill-in tasks available, waiting…

 t=19h58: No eligible fill-in tasks available, waiting…

 t=19h59: No eligible fill-in tasks available, waiting…

t=20h00: started 'Bake Cookies with Friends' for 120 mins…
         t=22h00, task completed!

 The day has ended! The following tasks have been delayed for tomorrow:

1) Task 7: Skill Builder for CS113
        Duration: 150 minutes
        Status: D
        Priority Score = 50

2) Task 9: Planning for the Rest of the Week
        Duration: 45 minutes
        Status: D
        Priority Score = 0


 Completed all planned tasks in 13h00min!

Final descriptions of executed tasks:
-------------------------------------
1. Get up and have breakfast
2. Meeting for AI Sustainability Lab
3. Daily Exercise
4. Do Pre-Class Work for CS111
5. Online Discussion with CCP Team
6. Revise last CS113 Content
7. CS111 Class
8. Bake Cookies with Friends
```