Part 1

My team has performed a critical path analysis using linear programming to plan a project. I have mapped out 16 tasks, identified their immediate predecessors, and assigned roles with their respective hourly rates(Appedex_4). You've calculated the duration of each task in the best-case(Appedex_3), expected-case(Appedex_2) , and worst-case (Appedex_1)scenarios.

Areas of uncertainty include:

Task duration may vary due to unforeseen challenges.

Resource availability could fluctuate, impacting the project timeline.

Cost estimates could change if the project scope alters or if unforeseen expenses arise.

Directed graph diagram (Appedex_5)

Part 2

The objective function to minimize the project's total duration directly minimizes the project's total cost. The model focused on minimizing the maximum end time among all tasks (which represents the total project duration

Cost Calculation

Total Project Cost: If the total project duration in hours is T, then the total project cost C can be expressed as:

$C=T\times\$50$

Part 3

Code for worse-case:

Appedex_6

Output:

Appedex_9

Code for Expected-case:

Appedex_7

Output:

Appedex_10

Code for Best-case:

Appedex_8

Output:

Appedex_11

Part 4

Using linear programming, the project plan was solved for each time estimate scenario. I used minimum-time objective. The solution provided the start and end times for each task, helping to identify the critical path—the sequence of tasks that directly affects the project's completion time.

For the best-case scenario, the project completes in 283 hours, the expected-case scenario in 461 hours, and the worst-case scenario in 693 hours.

The critical path for worst case scenario is:

Critical Path time:

Task A starts at time 0: Task A ("Describe product") can start immediately at the beginning of the project.

Task B starts at time 0: Task B ("Develop marketing strategy") can also start immediately as it has no dependencies.

Task D starts at time 0: Task D ("Develop product prototype") can start immediately as well as it has no dependencies.

H ends at 693.0 hours in duration: The project is complete when Task H ("Write client proposal") finishes, which, in the worst-case scenario, happens 693 days after the project starts.

The critical path for expected case scenario is:

Critical Path time:

The start times for tasks A, B, and D are the same as before, at time 0, meaning these tasks can begin right away with no dependencies.

Task H ends at 461.0 hours, which is the expected total duration of the project.

The critical path is start with Task H, then trace backward through its dependencies (tasks F and G in this case), then continue tracing back through each task's dependencies until you reach tasks with no predecessors.

The critical path for best case scenario is:

Critical Path time:

Just like the previous scenarios, tasks A, B, and D start immediately at time 0.

Task H, which is presumably the last task in the project sequence, ends in 283 hours. This marks the completion of the entire project.

The critical path in this scenario would be traced by looking at Task H and then going back through the sequence of dependencies that lead up to it. You look for the longest sequence of tasks (the path with the longest duration) leading to the completion of Task H without any delays or slack time. In other words, the critical path is the sequence of tasks where if any task is delayed, the entire project end date is pushed back.

Gantt charts for the best-case(Appedex_12), expected(Appedex_13), and worst-case (Appedex_14)solutions.

Part 5

For the prospective client:

Our project planning has meticulously laid out the roadmap for developing your product. With our expertise in linear programming, we've identified a completion window ranging from 283 to 693 hours, depending on various factors.

Our best-case estimate, assuming optimal conditions and resource availability, would see the project wrapped up in just under 1 month. However, we recommend planning for the expected-case scenario of approximately 3 months to allow for typical project uncertainties.

Considering our skilled team of independent contractors, the cost of the project would be calculated based on the expected hours worked, multiplied by the respective hourly rates. For the best-case scenario, this would translate to a total labor cost of approximately $283 x $50 =$14,150. This figure serves as a baseline, excluding software licenses and hosting services.

To expedite the delivery, we could integrate more independent contractors, reducing the critical path by allowing parallel task execution. This flexibility could potentially bring the product prototype to you sooner, depending on the specific tasks that could be accelerated.

In summary, we offer a robust plan with clear milestones and a detailed cost breakdown, ensuring transparency and efficiency from conception to delivery.

Appendix

Appedex_1

taskID	task	predecessorTaskIDs	worstCaseHours	projectManager	frontendDeveloper	backendDeveloper	dataScientist	dataEngineer
A	Describe product		9	9	0	0	0	0
B	Develop marketing strategy		9	9	0	0	0	0
C	Design brochure	A	60	10	20	20	0	10
D	Develop product prototype		120	12	36	36	0	36
D1	Requirements analysis	A	60	5	20	0	10	25
D2	Software design	D1	60	5	25	5	0	25
D3	System design	D1	60	5	25	12.5	0	17.5
D4	Coding	D2, D3	120	7.2	48	36	0	28.8
D5	Write documentation	D4	120	12	36	36	0	36
D6	Unit testing	D4	120	9.6	0	48	14.4	48
D7	System testing	D6	120	12	0	48	24	36
D8	Package deliverables	D5, D7	120	12	0	48	0	60
E	Survey potential market	B, C	60	15	0	0	45	0
F	Develop pricing plan	D8, E	60	15	0	0	45	0
G	Develop implementation plan	A, D8	60	20	0	0	40	0
H	Write client proposal	F, G	24	9	0	0	15	0

Appedex_2

taskID	task	predecessorTaskIDs	expectedHours	projectManager	frontendDeveloper	backendDeveloper	dataScientist	dataEngineer
A	Describe product		6	6	0	0	0	0
B	Develop marketing strategy		6	6	0	0	0	0
C	Design brochure	A	40	6.7	13.3	13.3	0	6.7
D	Develop product prototype		80	8	24	24	0	24
D1	Requirements analysis	A	40	3.3	13.3	0	6.7	16.7
D2	Software design	D1	40	3.3	16.7	3.3	0	16.7
D3	System design	D1	40	3.3	16.7	8.3	0	11.7
D4	Coding	D2, D3	80	4.8	32	24	0	19.2
D5	Write documentation	D4	80	8	24	24	0	24
D6	Unit testing	D4	80	6.4	0	32	9.6	32
D7	System testing	D6	80	8	0	32	16	24
D8	Package deliverables	D5, D7	80	8	0	32	0	40
E	Survey potential market	B, C	40	10	0	0	30	0
F	Develop pricing plan	D8, E	40	10	0	0	30	0
G	Develop implementation plan	A, D8	40	13.3	0	0	26.7	0
H	Write client proposal	F, G	15	5.6	0	0	9.4	0

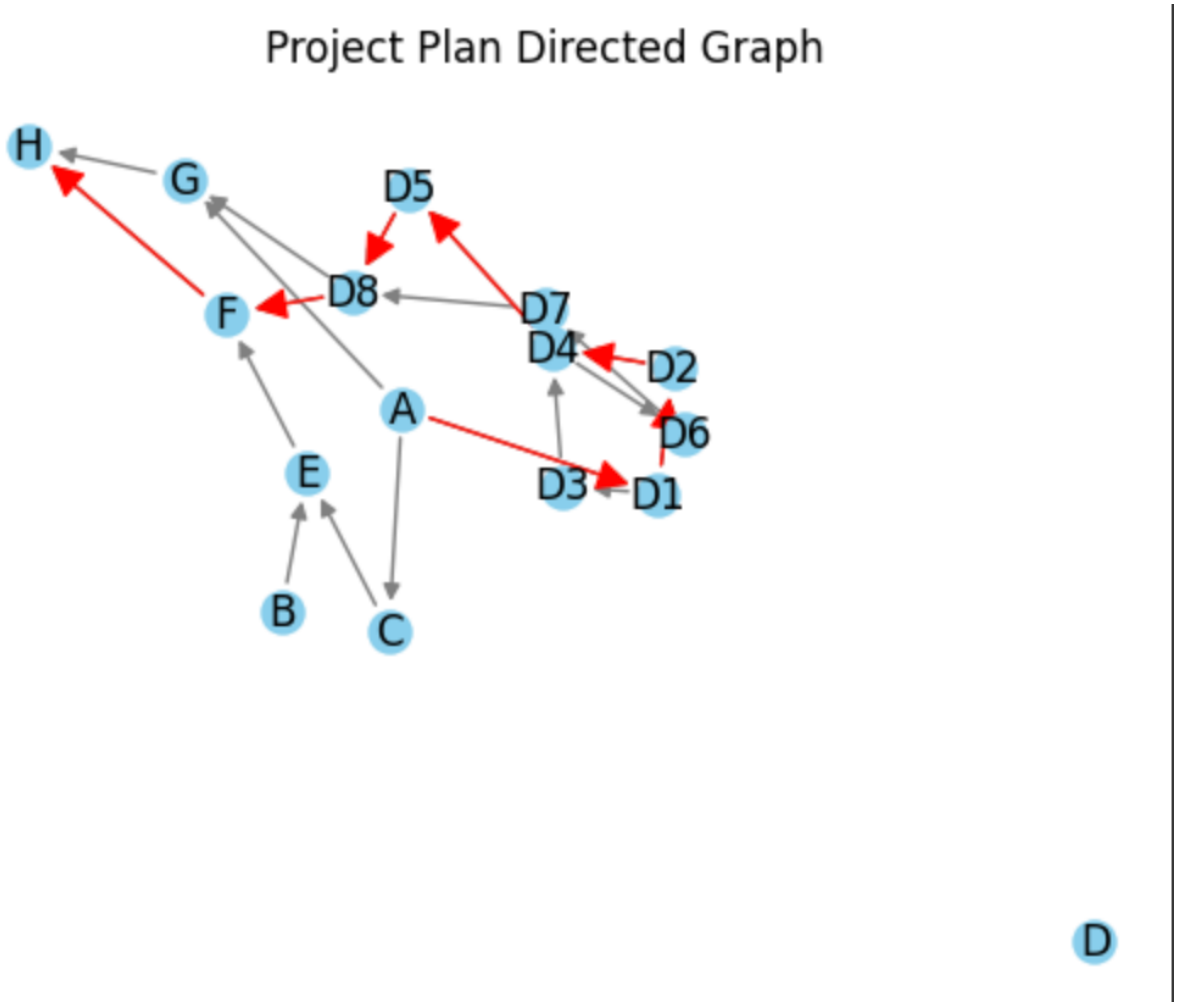
Appedex_3

taskID	task	predecessorTaskIDs	bestCaseHours	projectManager	frontendDeveloper	backendDeveloper	dataScientist	dataEngineer
A	Describe product		3	3	0	0	0	0
B	Develop marketing strategy		3	3	0	0	0	0
C	Design brochure	A	24	4	8	8	0	4
D	Develop product prototype		50	5	15	15	0	15
D1	Requirements analysis	A	24	2	8	0	4	10
D2	Software design	D1	24	2	10	2	0	10
D3	System design	D1	24	2	10	5	0	7
D4	Coding	D2, D3	50	3	20	15	0	12
D5	Write documentation	D4	50	5	15	15	0	15
D6	Unit testing	D4	50	4	0	20	6	20
D7	System testing	D6	50	5	0	20	10	15
D8	Package deliverables	D5, D7	50	5	0	20	0	25
E	Survey potential market	B, C	24	6	0	0	18	0
F	Develop pricing plan	D8, E	24	6	0	0	18	0
G	Develop implementation plan	A, D8	24	8	0	0	16	0
H	Write client proposal	F, G	8	3	0	0	5	0

Appedex_4

projectManager	$50/ hour
frontendDeveloper	$50/ hour
backendDeveloper	$50/ hour
dataScientist	$50/ hour
dataEngineer	$50/ hour

Appedex_5



Appendix_6

```
# worst_case_critical_path.py
from pulp import *

task_durations = {
   'A': 9, 'B': 9, 'C': 60, 'D': 120, 'D1': 60, 'D2': 60, 'D3': 60,
   'D4': 120, 'D5': 120, 'D6': 120, 'D7': 120, 'D8': 120, 'E': 60,
   'F': 60, 'G': 60, 'H': 24
}

# Create a list of the task_durations
task_durations_list = list(task_durations.keys())

# Update the precedences based on the new information provided.
precedences = {
   'A': [], 'B': [], 'C': ['A'], 'D': [], 'D1': ['A'], 'D2': ['D1'],
   'D3': ['D1'], 'D4': ['D2', 'D3'], 'D5': ['D4'], 'D6': ['D4'],
   'D7': ['D6'], 'D8': ['D5', 'D7'], 'E': ['B', 'C'], 'F': ['D8', 'E'],
   'G': ['A', 'D8'], 'H': ['F', 'G']
}

# Create the LP problem
prob = LpProblem("Critical_Path", LpMinimize)

# Decision variables for the start times
```

```
start_times = {task_durations: LpVariable(f"start_{task_durations}", 0, None) for
task_durations in task_durations_list}

end_times = {task_durations: LpVariable(f"end_{task_durations}", 0, None) for
task_durations in task_durations_list}



for task in task_durations_list:

   prob += end_times[task] == start_times[task] + task_durations[task], f"{task}_duration"

   for predecessor in precedences[task]:

      prob += start_times[task] >= end_times[predecessor],
f"{task}_predecessor_{predecessor}"


# Set the objective function

prob += lpSum([end_times[task] for task in task_durations_list]), "minimize_end_times"


# Solve the LP problem

status = prob.solve()


# Print the results

print("Critical Path time:")

for task in task_durations_list:

   if value(start_times[task]) == 0:

      print(f"{task} starts at time 0")

   if value(end_times[task]) == max([value(end_times[task]) for task in task_durations_list]):

      print(f"{task} ends at {value(end_times[task])} hours in duration")


# Print solution
```

```
print("\nSolution variable values:")
for var in prob.variables():
    if var.name != "_dummy":
        print(var.name, "=", var.varValue)
```

Appendix_7

```
# Expected_case_critical_path.py
from pulp import *



task_durations = {
   'A': 6, 'B': 6, 'C': 40, 'D': 80, 'D1': 40, 'D2': 40, 'D3': 40,
   'D4': 80, 'D5': 80, 'D6': 80, 'D7': 80, 'D8': 80, 'E': 40,
   'F': 40, 'G': 40, 'H': 15
}

# Update the precedences based on the new information provided.
precedences = {
   'A': [], 'B': [], 'C': ['A'], 'D': [], 'D1': ['A'], 'D2': ['D1'],
   'D3': ['D1'], 'D4': ['D2', 'D3'], 'D5': ['D4'], 'D6': ['D4'],
   'D7': ['D6'], 'D8': ['D5', 'D7'], 'E': ['B', 'C'], 'F': ['D8', 'E'],
   'G': ['A', 'D8'], 'H': ['F', 'G']
}

# Create the LP problem
prob = LpProblem("Critical_Path", LpMinimize)

# Decision variables for the start times
start_times = {task_durations: LpVariable(f"start_{task_durations}", 0, None) for
task_durations in task_durations_list}
end_times = {task_durations: LpVariable(f"end_{task_durations}", 0, None) for
task_durations in task_durations_list}
```

```
for task in task_durations_list:
    prob += end_times[task] == start_times[task] + task_durations[task], f"{task}_duration"
    for predecessor in precedences[task]:
        prob += start_times[task] >= end_times[predecessor],
f"{task}_predecessor_{predecessor}"

# Set the objective function
prob += lpSum([end_times[task] for task in task_durations_list]), "minimize_end_times"

# Solve the LP problem
status = prob.solve()

# Print the results
print("Critical Path time:")
for task in task_durations_list:
    if value(start_times[task]) == 0:
        print(f"{task} starts at time 0")
    if value(end_times[task]) == max([value(end_times[task]) for task in task_durations_list]):
        print(f"{task} ends at {value(end_times[task])} hours in duration")
# Print solution
print("\nSolution variable values:")
for var in prob.variables():
    if var.name != "_dummy":
        print(var.name, "=", var.varValue)
```

Appendix_8

```
# Best_case_critical_path.py
from pulp import *



task_durations = {
   'A': 3, 'B': 3, 'C': 24, 'D': 50, 'D1': 24, 'D2': 24, 'D3': 24,
   'D4': 50, 'D5': 50, 'D6': 50, 'D7': 50, 'D8': 50, 'E': 24,
   'F': 24, 'G': 24, 'H': 8
}

# Update the precedences based on the new information provided.
precedences = {
   'A': [], 'B': [], 'C': ['A'], 'D': [], 'D1': ['A'], 'D2': ['D1'],
   'D3': ['D1'], 'D4': ['D2', 'D3'], 'D5': ['D4'], 'D6': ['D4'],
   'D7': ['D6'], 'D8': ['D5', 'D7'], 'E': ['B', 'C'], 'F': ['D8', 'E'],
   'G': ['A', 'D8'], 'H': ['F', 'G']
}

# Create the LP problem
prob = LpProblem("Critical_Path", LpMinimize)

# Decision variables for the start times
start_times = {task_durations: LpVariable(f"start_{task_durations}", 0, None) for
task_durations in task_durations_list}
end_times = {task_durations: LpVariable(f"end_{task_durations}", 0, None) for
task_durations in task_durations_list}
```

```
for task in task_durations_list:
    prob += end_times[task] == start_times[task] + task_durations[task], f"{task}_duration"
    for predecessor in precedences[task]:
        prob += start_times[task] >= end_times[predecessor],
f"{task}_predecessor_{predecessor}"

# Set the objective function
prob += lpSum([end_times[task] for task in task_durations_list]), "minimize_end_times"

# Solve the LP problem
status = prob.solve()

# Print the results
print("Critical Path time:")
for task in task_durations_list:
    if value(start_times[task]) == 0:
        print(f"{task} starts at time 0")
    if value(end_times[task]) == max([value(end_times[task]) for task in task_durations_list]):
        print(f"{task} ends at {value(end_times[task])} hours in duration")

# Print solution
print("\nSolution variable values:")
for var in prob.variables():
    if var.name != "_dummy":
        print(var.name, "=", var.varValue)
```

appendix_9

Critical Path time:

A starts at time 0

B starts at time 0

D starts at time 0

H ends at 693.0 hours in duration

Solution variable values:

end_A = 9.0

end_B = 9.0

end_C = 69.0

end_D = 120.0

end_D1 = 69.0

end_D2 = 129.0

end_D3 = 129.0

end_D4 = 249.0

end_D5 = 369.0

end_D6 = 369.0

end_D7 = 489.0

end_D8 = 609.0

end_E = 129.0

end_F = 669.0

end_G = 669.0

end_H = 693.0

start_A = 0.0

start_B = 0.0

start_C = 9.0

start_D = 0.0

start_D1 = 9.0

start_D2 = 69.0

start_D3 = 69.0

start_D4 = 129.0

start_D5 = 249.0

start_D6 = 249.0

start_D7 = 369.0

start_D8 = 489.0

start_E = 69.0

start_F = 609.0

start_G = 609.0

start_H = 669.0

Appedix_10

Critical Path time:

A starts at time 0

B starts at time 0

D starts at time 0

H ends at 461.0 hours in duration

Solution variable values:

end_A = 6.0

end_B = 6.0

end_C = 46.0

end_D = 80.0

end_D1 = 46.0

end_D2 = 86.0

end_D3 = 86.0

end_D4 = 166.0

end_D5 = 246.0

end_D6 = 246.0

end_D7 = 326.0

end_D8 = 406.0

end_E = 86.0

end_F = 446.0

end_G = 446.0

end_H = 461.0

start_A = 0.0

start_B = 0.0

start_C = 6.0

start_D = 0.0

start_D1 = 6.0

start_D2 = 46.0

start_D3 = 46.0

start_D4 = 86.0

start_D5 = 166.0

start_D6 = 166.0

start_D7 = 246.0

start_D8 = 326.0

start_E = 46.0

start_F = 406.0

start_G = 406.0

start_H = 446.0

Appedix_11

Critical Path time:

A starts at time 0

B starts at time 0

D starts at time 0

H ends at 283.0 hours in duration

Solution variable values:

end_A = 3.0

end_B = 3.0

end_C = 27.0

end_D = 50.0

end_D1 = 27.0

end_D2 = 51.0

end_D3 = 51.0

end_D4 = 101.0

end_D5 = 151.0

end_D6 = 151.0

end_D7 = 201.0

end_D8 = 251.0

end_E = 51.0

end_F = 275.0

end_G = 275.0

end_H = 283.0

start_A = 0.0

start_B = 0.0

start_C = 3.0

start_D = 0.0

start_D1 = 3.0

start_D2 = 27.0

start_D3 = 27.0

start_D4 = 51.0

start_D5 = 101.0

start_D6 = 101.0

start_D7 = 151.0

start_D8 = 201.0

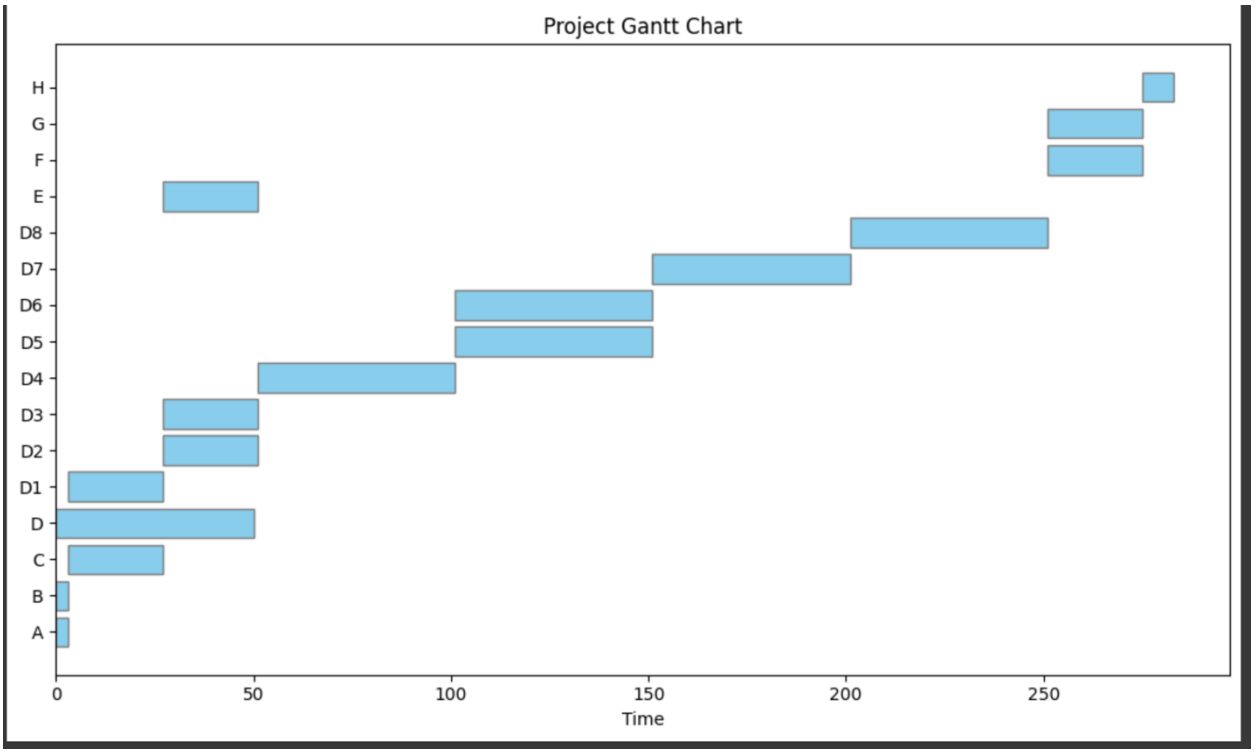
start_E = 27.0

start_F = 251.0

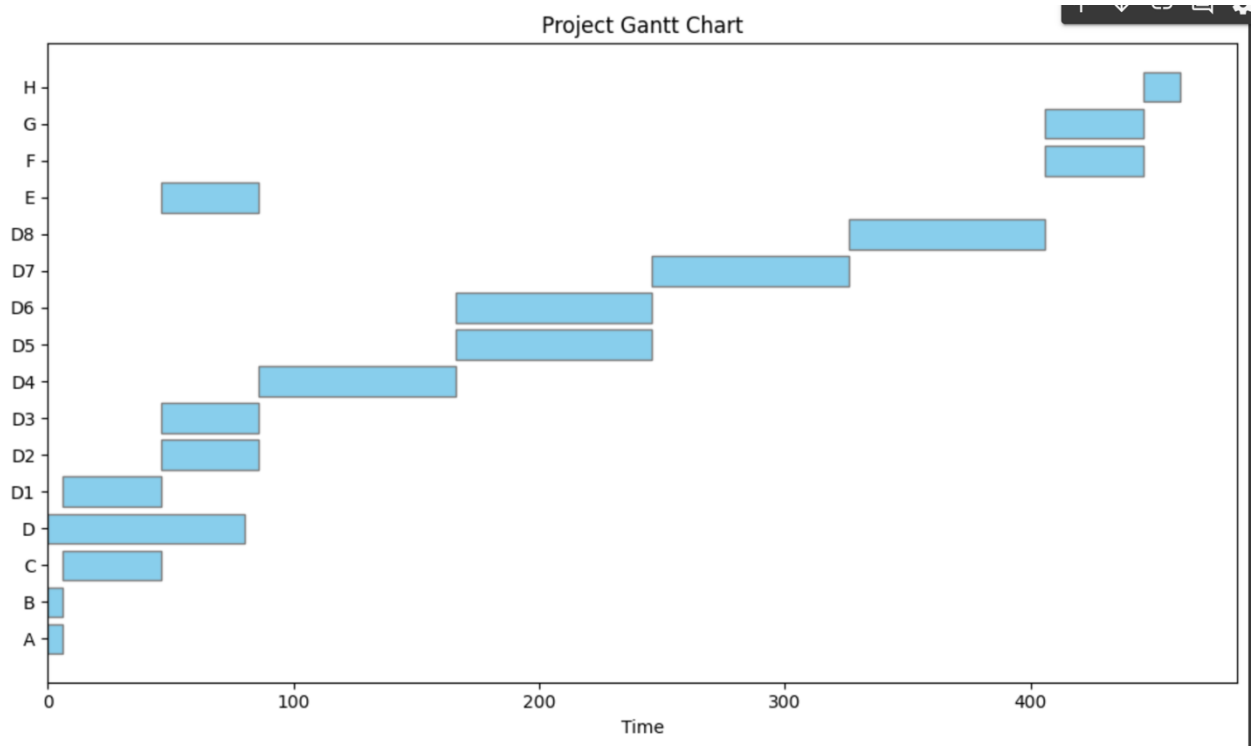
start_G = 251.0

start_H = 275.0

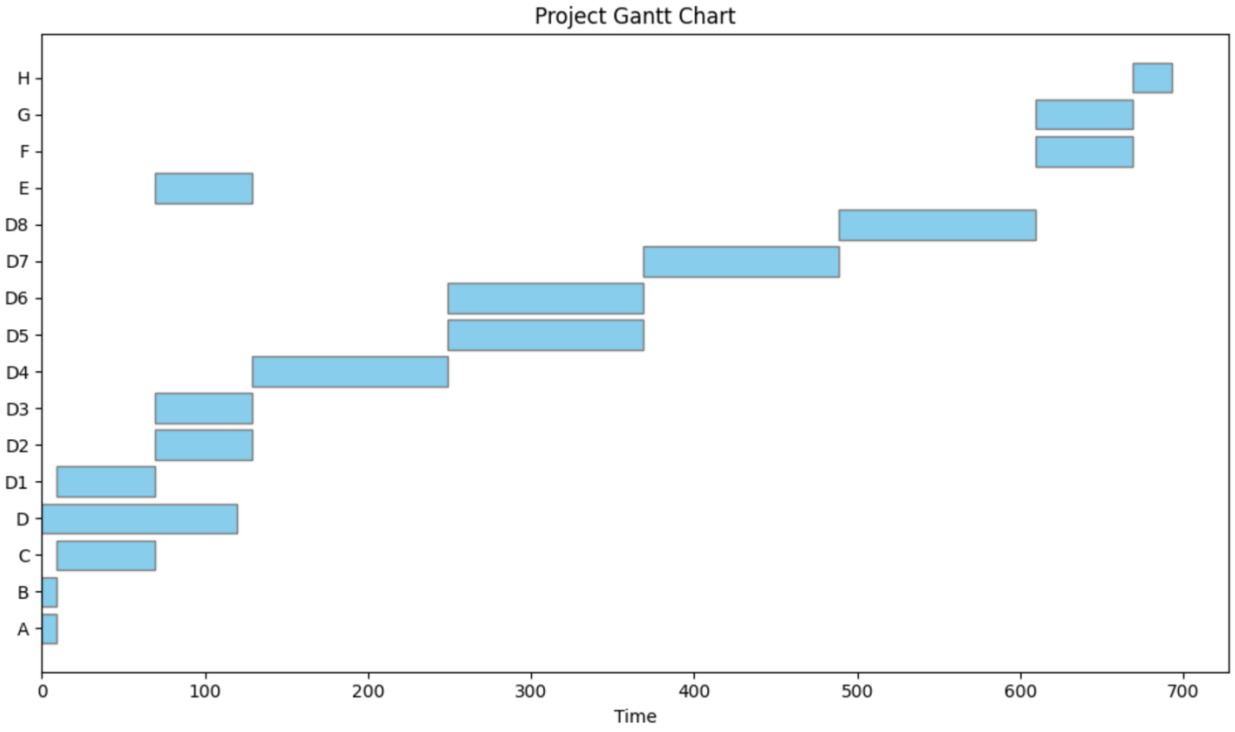
Appedix_12



Appedix_13



Appedix_14


Project Gantt Chart
H
G
F
E
D8
D7
D6
D5
D4
D3
D2
D1
D
C
B
A
0
100
200
300
400
500
600
700
Time