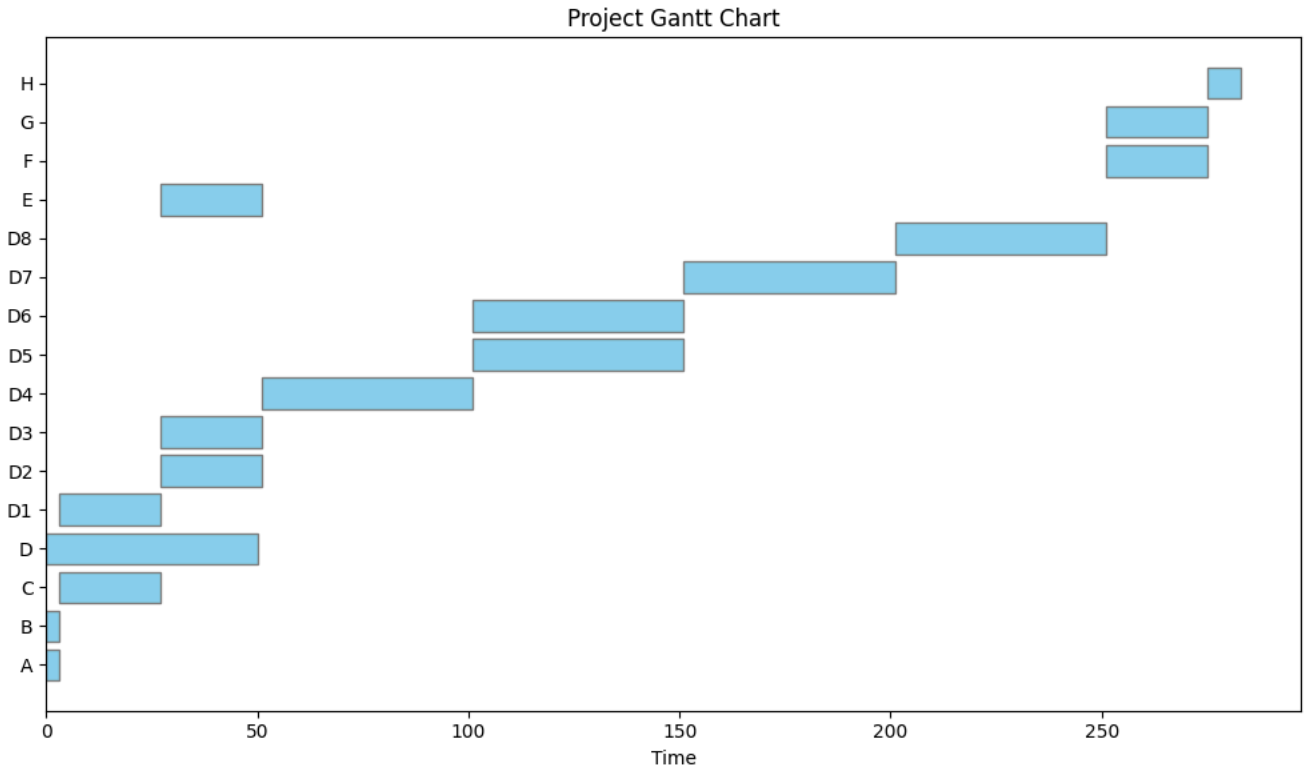
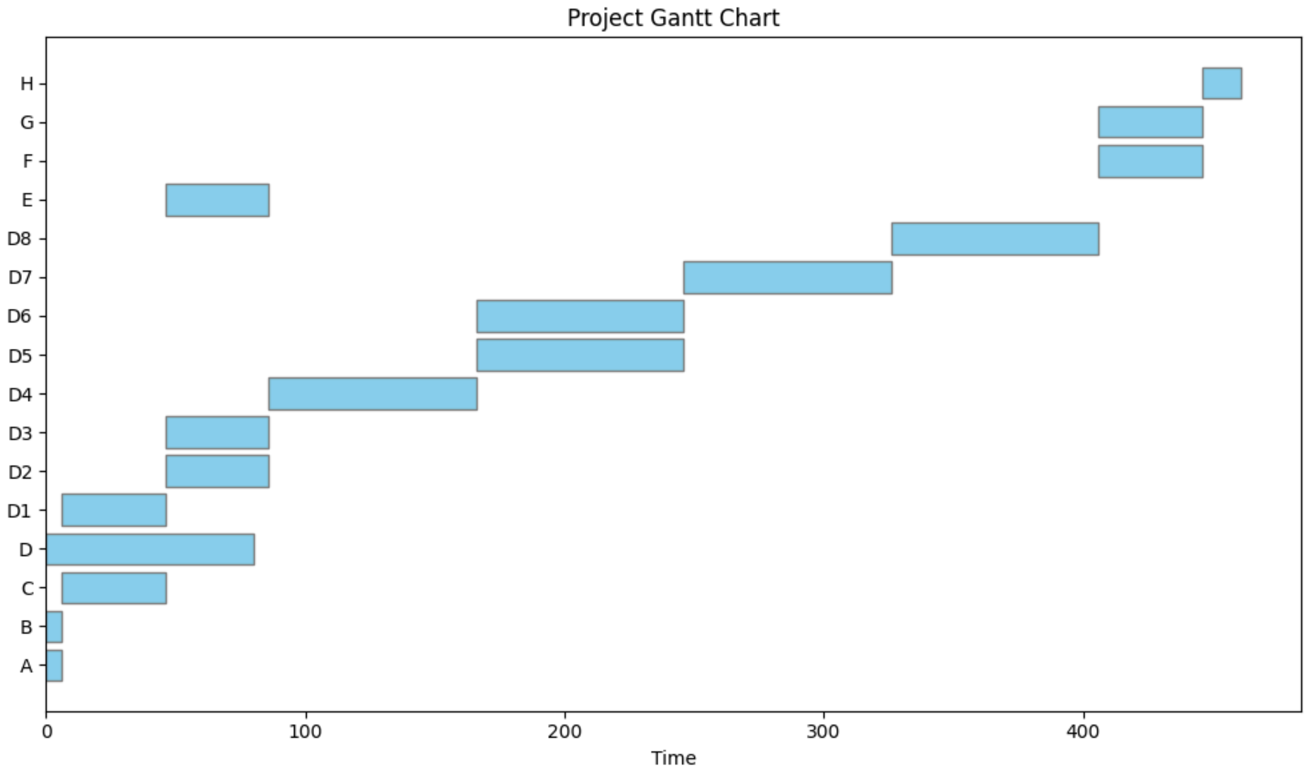
```
#best case gantt charts
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import datetime, timedelta
from pulp import *

# Given durations and precedences
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)

# Create variables for the start times
start_times = LpVariable.dicts("Start", task_durations.keys(), 0)
end_times = LpVariable.dicts("End", task_durations.keys(), 0)

# Add the constraints for task durations and precedences
for task, duration in task_durations.items():
    prob += end_times[task] == start_times[task] + duration
    for predecessor in precedences.get(task, []):
        prob += start_times[task] >= end_times[predecessor]

# Set the objective function to minimize the latest end time
prob += lpSum([end_times[task] for task in task_durations.keys()])

# Solve the problem
prob.solve()

# Now we have start and end times defined, let's proceed to create the Gantt chart
task_times = {task: (value(start_times[task]), value(end_times[task])) for task in task_durations}

# Create the plot figure
fig, ax = plt.subplots(figsize=(10, 6))

# Create a bar for each task
for task, (start, end) in task_times.items():
    ax.barh(task, end - start, left=start, color='skyblue', edgecolor='grey')

# Set the labels and titles
ax.set_xlabel('Time')
ax.set_title('Project Gantt Chart')
plt.tight_layout()
plt.show()
```

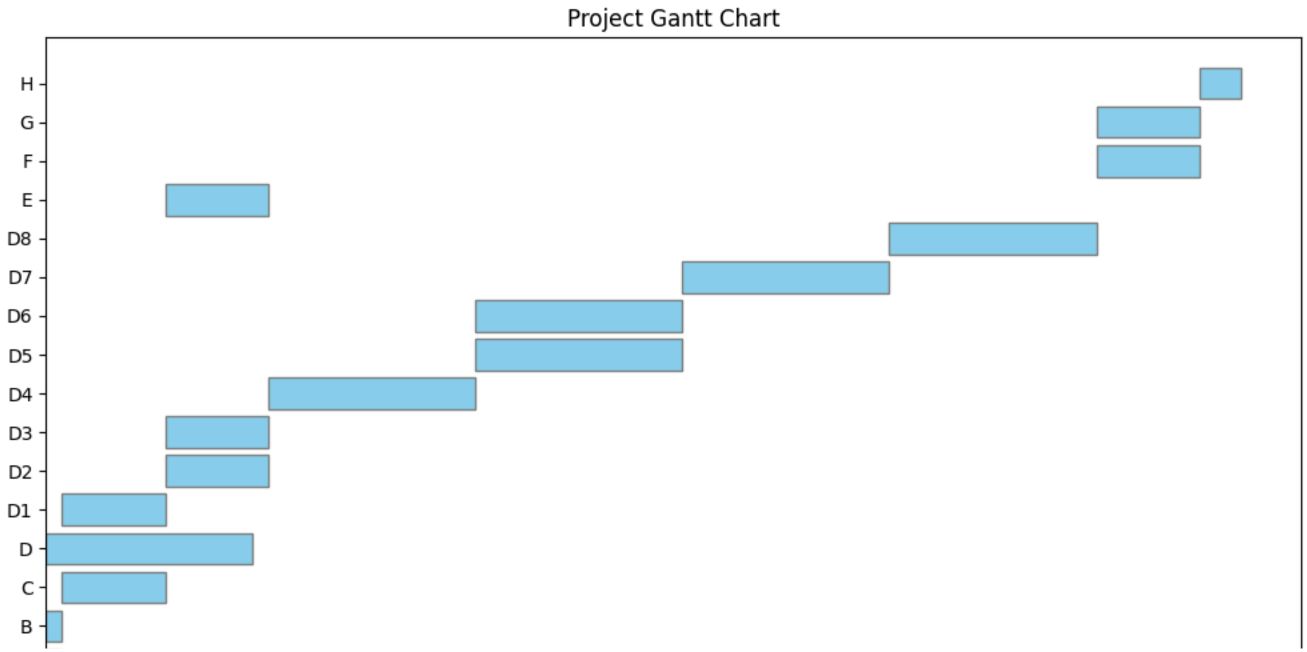

Project Gantt Chart
H
G
F
E
D8
D7
D6
D5
D4
D3
D2
D1
D
C
B
A
0
50
100
150
200
250
Time

```
# Expected_case gantt charts
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import datetime, timedelta
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)

# Create variables for the start times
start_times = LpVariable.dicts("Start", task_durations.keys(), 0)
end_times = LpVariable.dicts("End", task_durations.keys(), 0)

# Add the constraints for task durations and precedences
for task, duration in task_durations.items():
    prob += end_times[task] == start_times[task] + duration
    for predecessor in precedences.get(task, []):
        prob += start_times[task] >= end_times[predecessor]

# Set the objective function to minimize the latest end time
prob += lpSum([end_times[task] for task in task_durations.keys()])

# Solve the problem
prob.solve()

# Now we have start and end times defined, let's proceed to create the Gantt chart
task_times = {task: (value(start_times[task]), value(end_times[task])) for task in task_durations}

# Create the plot figure
fig, ax = plt.subplots(figsize=(10, 6))

# Create a bar for each task
for task, (start, end) in task_times.items():
    ax.barh(task, end - start, left=start, color='skyblue', edgecolor='grey')

# Set the labels and titles
ax.set_xlabel('Time')
ax.set_title('Project Gantt Chart')
plt.tight_layout()
plt.show()
```



Project Gantt Chart

```
# Worst case gantt charts
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import datetime, timedelta
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
}

# 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)

# Create variables for the start times
start_times = LpVariable.dicts("Start", task_durations.keys(), 0)
start_times = LpVariable.dicts("Start", task_durations.keys(), 0)
end_times = LpVariable.dicts("End", task_durations.keys(), 0)

# Add the constraints for task durations and precedences
for task, duration in task_durations.items():
    prob += end_times[task] == start_times[task] + duration
    for predecessor in precedences.get(task, []):
        prob += start_times[task] >= end_times[predecessor]

# Set the objective function to minimize the latest end time
prob += lpSum([end_times[task] for task in task_durations.keys()])

# Solve the problem
prob.solve()

# Now we have start and end times defined, let's proceed to create the Gantt chart
task_times = {task: (value(start_times[task]), value(end_times[task])) for task in task_durations}

# Create the plot figure
fig, ax = plt.subplots(figsize=(10, 6))

# Create a bar for each task
for task, (start, end) in task_times.items():
    ax.barh(task, end - start, left=start, color='skyblue', edgecolor='grey')

# Set the labels and titles
ax.set_xlabel('Time')
ax.set_title('Project Gantt Chart')
plt.tight_layout()
plt.show()
```


Project Gantt Chart
H
G
F
E
D8
D7
D6
D5
D4
D3
D2
D1
D
C
B