Network Models
Assignment 2

Aaliya Merchant
MSDS 460: Decision Analytics
October 20th, 2024

1 Introduction

The purpose of this project is to design and develop a consumer-focused recommendation system for over 100 restaurants in Marlborough, Massachusetts, utilizing Yelp reviews as a primary source of data. This system will be dynamic, with daily updates from the Yelp GraphQL API to reflect changes in customer feedback and restaurant performance. The primary goal is to provide restaurant owners with a valuable tool to help customers discover dining options tailored to their specific preferences, thereby enhancing customer satisfaction, loyalty, and engagement.

As Marlborough continues to establish itself as a hub for culinary experiences, the competitive nature of the industry necessitates a system that offers personalized recommendations. This not only ensures that local businesses can better meet the demands of their customers but also helps restaurants stand out in an increasingly digital marketplace. This recommendation system could be instrumental in ensuring that small, independently owned restaurants have a competitive edge over larger chains that often have more extensive marketing resources. Success depends on careful planning, task scheduling, and resource allocation, which are detailed throughout this report.

2 Methods

A detailed project plan has been developed, consisting of sixteen distinct tasks, each with assigned team members and completion estimates. The project team includes roles such as Project Manager, Frontend Developer, Backend Developer, Data Scientist, and Data Engineer. Each task is defined in terms of best-case, expected, and worst-case time estimates to account for uncertainties. These estimates were compiled using expert judgment and historical data from similar projects. The project's resource allocation and task dependencies are represented in an Excel spreadsheet, which captures the following data:

- Task ID and Description
- Predecessor Tasks
- Estimated Completion Times (Best-case, Expected, Worst-case)
- Hourly Costs per Role
- Personnel Responsible

Uncertainties in the project arise primarily in the software development and testing phases, where tasks such as *Coding* and *Unit Testing* may encounter unforeseen technical complexities or integration challenges, potentially prolonging completion times. Additionally, external factors like changes to the Yelp API structure or fluctuations in data availability could further impact the project timeline. To mitigate these risks, worst-case time estimates have been incorporated to account for potential delays. Despite these uncertainties, tasks like *Design Brochure* and *Survey Potential Market* can be completed in parallel, helping to optimize the project's overall duration and minimize bottlenecks.

The formula used for calculating the expected time (T_e) for each task is given by:

$$T_e = \frac{T_{best} + 4 \times T_{expected} + T_{worst}}{6}$$

For instance, in the case of *Software Design (Task D2)*, the best-case estimate was 4 hours, the expected case was 25 hours, and the worst-case was 49 hours. Using the formula, the expected time was calculated as follows:

$$T_e = \frac{4 + 4 \times 25 + 49}{6} = 25.67 \text{ hours.}$$

This method allows for factoring in uncertainties and risks associated with project components, such as delays in design or testing.

2.1 Directed Graph Diagram

The directed graph diagram illustrating tasks and dependencies is included in the project documentation. This diagram visually represents the flow of tasks, highlighting critical paths and tasks that can be completed in parallel, thus optimizing resource allocation and scheduling. Figures 4, 5, and 6 show the directed graph representations for the best-case, expected-case, and worst-case scenarios, respectively. Each figure provides insights into task interdependencies and the overall project timeline.

3 Model Specification

3.1 Decision Variables

Let x_i represent the number of hours allocated to task i.

The objective is to minimize the total cost associated with the project. Since all contributors charge the same hourly rate, the objective function can be defined as:

$$\text{Minimize } Z = \sum_{i=1}^{n} x_i$$

where n is the total number of tasks.

The constraints for the LP model are defined as follows:

Each task must be allocated between the best-case and worst-case hours:

$$T_{best_i} \leq x_i \leq T_{worst_i}, \quad \forall i \in \{A, B, C, D1, D2, D3, D4, D5, D6, D7, D8, E, F, G, H\}$$

Dependent tasks must not start until their predecessors have been completed:

$$x_j \geq x_i \text{ for all dependencies}$$

Combining the objective function and constraints, the linear programming model is specified as follows:

$$\text{Minimize } Z = \sum_{i=1}^{n} x_i$$

Subject to:

1. $T_{best_i} \leq x_i \leq T_{worst_i}, \quad \forall i \in \{A, B, C, D1, D2, D3, D4, D5, D6, D7, D8, E, F, G, H\}$
2. $x_j \geq x_i$ for all dependencies

The time estimates obtained from the previous analysis using PERT directly inform the constraints of this model, ensuring that task allocations are realistic and account for potential uncertainties.

4 Programming

4.1 Implementation Overview

The linear programming model for the project plan was implemented using Python's PuLP library, which provides a flexible and intuitive way to define and solve linear optimization problems. The primary objective of this implementation is to minimize the total time required to complete the project while accounting for task dependencies and durations based on best-case, expected, and worst-case estimates. The complete code implementation and output listings for all scenarios can be found in Appendix 8.4.

The core of the program lies in the `solve_project_plan()` function, which takes the task durations and a scenario name (such as best-case, expected, or worst-case) as input. This function defines the linear programming problem, adds the necessary constraints, and solves the model. Below is an overview of the key components of the code:

1. **Decision Variables**: The decision variables x_i represent the time allocated to each task, defined using `LpVariable.dicts()`.

2. **Objective Function**: The objective is to minimize the total project time, formulated as the sum of all task times:

$$\text{Minimize } Z = \sum_{i=1}^{n} x_i$$

3. **Constraints**: Task dependencies are modeled as precedence constraints. For instance, task C cannot start until task A is completed:

$$x_C \geq x_A + \text{duration}(A)$$

Similar constraints are added for all dependent tasks, ensuring proper sequencing.

4. **Solution and Output**: After defining the decision variables, objective function, and constraints, the problem is solved using the `solve()` method. The solution provides the optimal task durations, which are printed to the console along with the total project time.

The full source code and the outputs for all three scenarios (best-case, expected, and worst-case) have been uploaded to a dedicated GitHub repository. The repository can be accessed at the following URL:
`https://github.com/username/project-plan-optimization`.

4.2 Results

The optimization results provide insights into the total project completion time under different scenarios. Below are the summarized results, including both the optimized total duration and the critical path:

- **Best Case:**
  - Optimized Total Project Completion Time: 123.0 hours
  - Critical Path: ['A', 'D1', 'D3', 'D4', 'D6', 'D7', 'D8', 'G', 'H']
- **Expected Case:**
  - Optimized Total Project Completion Time: 169.0 hours
  - Critical Path:['A', 'D1', 'D2', 'D3', 'D4', 'D6', 'D7', 'D8', 'G', 'H']
- **Worst Case:**
  - Optimized Total Project Completion Time: 760.0 hours
  - Critical Path: ['A', 'D1', 'D2', 'D3', 'D4', 'D6', 'D7', 'D8', 'F', 'G', 'H']

The optimization process aims to minimize the project duration by adjusting task dependencies and durations. While the optimized durations represent the most efficient timelines, the critical path remains crucial for understanding potential delays in project completion.

5 Solution

This section delves into the project plan's optimization across three scenarios: best-case, expected-case, and worst-case. Each scenario provides insights into how various factors influence the overall project timeline while keeping in mind that our analysis assumes an idealized environment without resource constraints. In reality, this assumption often doesn't hold, as project teams face limitations such as finite work hours and varying availability.

5.1 Best Case Scenario

In the best-case scenario, we find that the total project completion time is 123.0 hours (see Section 8.4.1). This situation assumes everything goes smoothly, with tasks progressing without delays or hiccups. Tasks A and B are both set to zero hours, which implies they aren't critical to the project's timeline under these ideal conditions.

The critical path, comprising tasks `A`, `D1`, `D3`, `D4`, `D6`, `D7`, `D8`, `G`, and `H`, totals 133 hours. This difference between the optimized project duration (123.0 hours) and the critical path duration (133 hours) arises because the optimization reflects an ideal situation where certain tasks can run concurrently without affecting the project's overall timing. In contrast, the critical path identifies the longest sequence of dependent tasks that dictate the minimum time required to complete the project.

The Best-Case Gantt Chart, shown in Figure 1, illustrates an efficient workflow, demonstrating how tasks can be executed in parallel, thereby enhancing the likelihood of meeting the project deadline.

5.2 Expected Case Scenario

When we look at the expected-case scenario, the total project duration extends to 169.0 hours (see Section 8.4.2). This scenario considers typical challenges that could arise during the project, such as minor delays or resource constraints. Similar to the best-case scenario, tasks A and B remain zero hours, reinforcing their non-critical status.

The critical path in this case consists of tasks `A`, `D1`, `D2`, `D3`, `D4`, `D6`, `D7`, `D8`, `G`, and `H`, with a total duration of 179 hours. The increase from the best-case scenario highlights how the introduction of potential delays—particularly in task D5, which requires 99.0 hours—significantly impacts the project timeline. Here, the critical path duration

exceeds the optimized completion time, demonstrating how delays in task execution can cascade through the project.

The Expected-Case Gantt Chart, shown in Figure 2, reveals a more complex timeline, emphasizing the need for vigilant project management to address any potential bottlenecks and ensure timely progress.

5.3 Worst Case Scenario

In the worst-case scenario, we see the total project duration ballooning to 760.0 hours (see Section 8.4.3). This outcome is a reflection of severe delays and challenges that might arise, such as resource shortages or unexpected complexities in tasks.

Here, the critical path includes tasks `A`, `D1`, `D2`, `D3`, `D4`, `D6`, `D7`, `D8`, `F`, `G`, and `H`, culminating in a total duration of 939 hours. This considerable gap between the total project duration (760.0 hours) and the critical path duration (939 hours) emphasizes how delays in essential tasks can dramatically affect the overall timeline. For instance, tasks D5 (178.0 hours) and F (591.0 hours) contribute heavily to this increase, illustrating the cascading effect of one delay leading to another.

The Worst-Case Gantt Chart, shown in Figure 3, depicts a congested timeline, underscoring the challenges that arise when managing a project under adverse conditions. The results indicate a need for robust risk management strategies to navigate potential delays and meet project goals.

5.4 Comparative Analysis

When comparing the critical path across these scenarios, it becomes clear that variations in project completion times are significant. The optimized duration in the best-case scenario is 123.0 hours, while the expected-case scenario sees an increase to 169.0 hours. In contrast, the worst-case scenario escalates to 760.0 hours.

One important takeaway from this analysis is that critical path analysis, as applied here, assumes no resource constraints—a simplification that doesn't reflect most real-world projects. In practice, team members can only manage a limited number of hours each day, and not every member can work on every task simultaneously.

The variations between optimized and critical path durations highlight the importance of effective scheduling and resource allocation in project management. The optimized durations represent a best-case execution based on ideal conditions, while critical path durations reflect the longest necessary completion time determined by dependencies. Understanding this distinction is vital for project managers to identify which tasks require the most attention to ensure timely delivery and to develop contingency plans for potential delays.

In conclusion, this analysis emphasizes the need for ongoing monitoring of task progress and a proactive approach to risk management. By anticipating delays and adapting to changing project conditions, we can better navigate the complexities of project execution and work towards successful outcomes.

6 Overview

6.1 Project Overview for Prospective Client

The project focuses on developing a consumer-centered recommendation system for over 100 restaurants in Marlborough, Massachusetts. This system will harness Yelp reviews to provide real-time updates using the Yelp GraphQL API, ensuring that restaurant recommendations reflect the latest customer feedback. The frontend will be built with Alpine.js and Tailwind, offering a user-friendly interface, while the backend will use a Go web server integrated with a database such as PostgreSQL, EdgeDB, or PocketBase. Hosting the system on a major cloud platform will ensure scalability and reliability, addressing the client's need for a robust, future-proof solution.

6.2 Project Costs

The estimated total project cost is $43,470, which includes expected working hours and the hourly rates of independent contractors, as detailed in Table 2. This estimate covers all roles, including project management, frontend and backend development, data engineering, and data science. However, it is important to note that this figure does not include costs related to software licensing or cloud hosting, both of which may vary depending on the specific tools and services chosen by the client. Additionally, unforeseen challenges during development could affect costs, particularly in areas such as system integration or API adjustments if Yelp updates its structure.

6.3 Product Delivery Timeline

Under the expected scenario, the project will take approximately 169.0 hours to complete, leading to an estimated product prototype delivery time of **7 weeks**, assuming no significant delays and a standard work schedule.

- **Product Prototype Delivery (Expected Case)**: 7 weeks

6.4 Accelerated Delivery with Additional Contractors

To expedite the project, the client may choose to add more independent contractors, specifically one additional backend developer and one frontend developer. Doing so would allow the workload to be more evenly distributed, reducing bottlenecks in key areas. This strategy could shorten the delivery timeline by up to 30%, resulting in an estimated product prototype delivery in **5 weeks**.

- **Accelerated Product Prototype Delivery**: 5 weeks

While this accelerated timeline would speed up development, it would also increase the overall project cost due to additional resource requirements. The client should weigh the trade-offs between cost and time to determine the best approach based on project priorities and budget constraints.

7 Conclusion

In conclusion, the development of the consumer-focused recommendation system for over 100 restaurants in Marlborough, Massachusetts, aligns closely with the project's original objectives: enhancing customer engagement and satisfaction through personalized recommendations. By leveraging Yelp reviews and updating data dynamically through the Yelp GraphQL API, the system provides restaurant owners with valuable insights into customer preferences, positioning local businesses for increased competitiveness in a crowded marketplace.

The project plan effectively balances the client's need for timely delivery and cost control. By applying critical path analysis and using best-case, expected, and worst-case time estimates, we have developed a structured approach to managing the complexities of software development. This approach ensures that the product can be delivered within the client's timeframe, with the potential to expedite delivery if additional resources are allocated.

The analysis of potential scenarios highlights the importance of proactive risk management. The range of project completion times—from 123.0 hours in the best-case scenario to 760.0 hours in the worst-case—illustrates the possible impact of unforeseen technical challenges. Implementing comprehensive risk mitigation strategies will be essential to meet the client's requirements for timely delivery and reliable performance.

Ultimately, the recommendation system aims to meet the client's requirements by offering a scalable, reliable product that delivers personalized dining experiences for customers while empowering restaurant owners with data-driven insights. Through ongoing evaluation and flexibility in execution, the project is well-positioned to achieve these goals, ensuring a high-quality solution that meets both functional and performance expectations.

8 Appendices

8.1 Tables

Task ID	Task	predecessorTaskIDs	bestCaseHours	expectedHours	worstCaseHours	projectManager	frontendDeveloper	backendDeveloper	dataScientist	dataEngineer
A	Describe product	-	6	6	6	6	-	-	-	-
B	Develop marketing strategy	-	15	15	44	15	-	-	-	-
C	Design brochure	A	12	12	12	4	12	-	-	-
D1	Requirements analysis	A	18	18	24	8	-	-	8	8
D2	Software design	D1	4	25	49	-	15	-	-	10
D3	System design	D1	25	25	49	-	-	25	-	-
D4	Coding	D2, D3	10	50	99	-	10	40	-	40
D5	Write documentation	D4	15	15	42	15	-	-	-	-
D6	Unit testing	D4	18	18	17	-	18	-	-	18
D7	System testing	D6	25	25	42	-	-	25	-	-
D8	Package deliverables	D5, D7	6	12	54	6	-	-	12	-
E	Survey potential market	B, C	10	15	59	10	-	-	15	-
F	Develop pricing plan	D8, E	10	10	69	10	-	-	-	-
G	Develop implementation plan	A, D8	15	15	69	15	-	-	-	15
H	Write client proposal	F, G	10	10	79	10	-	-	-	-

Table 1: Task List and Time (hours) Estimates with Role Allocations

8.2 Tables

Role	**Hourly Rate**	**Estimated Hours**	**Cost**
Project Manager	$120/hour	61 hours	$7,320
Frontend Developer	$100/hour	87 hours	$8,700
Backend Developer	$110/hour	118 hours	$12,980
Data Engineer	$115/hour	78 hours	$8,970
Data Scientist	$125/hour	48 hours	$6,000
Total Cost			**$43,470**

Table 2: Adjusted Estimated Project Cost Breakdown

8.2.1 Gantt Chart

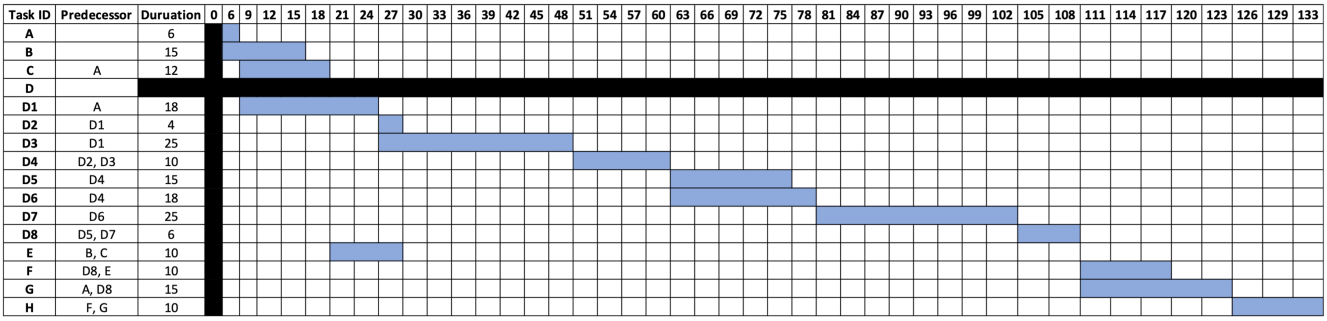

Figure 1: Best-Case Gantt Chart: Time Estimates (in hours) for the Project Plan

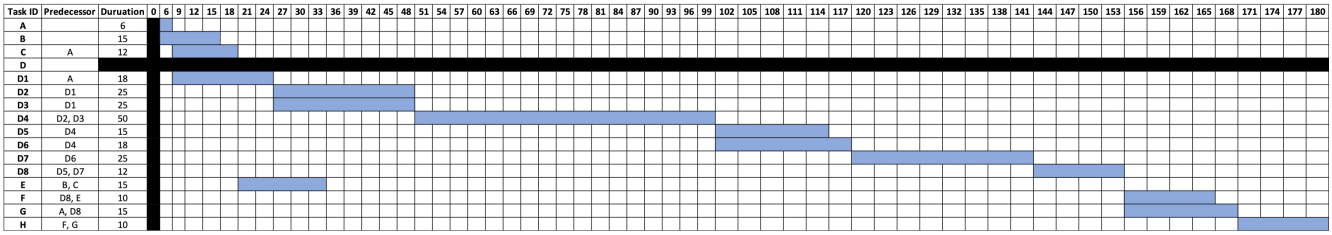

Figure 2: Expected-Case Gantt Chart: Time Estimates (in hours) for the Project Plan

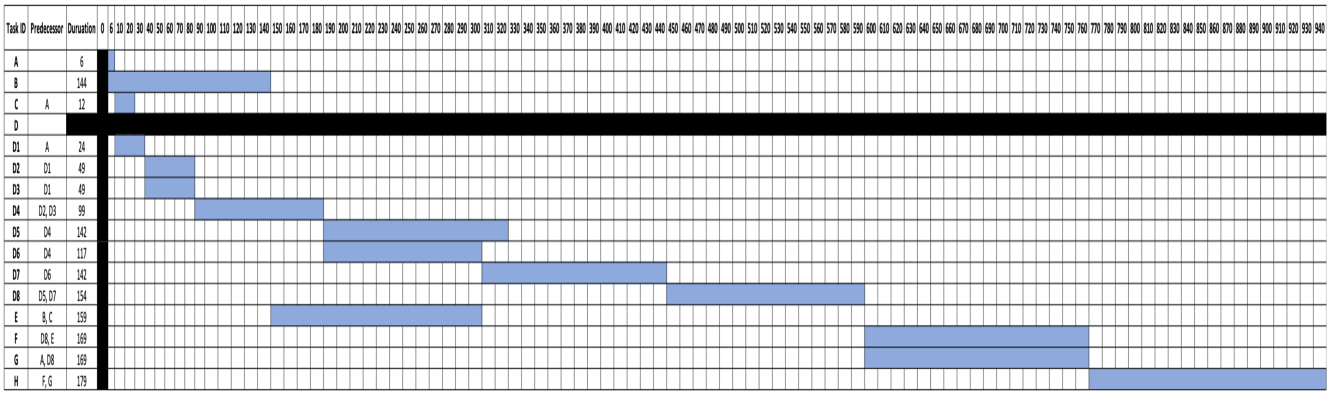

Figure 3: Worst-Case Gantt Chart: Time Estimates (in hours) for the Project Plan

8.3 Graphs

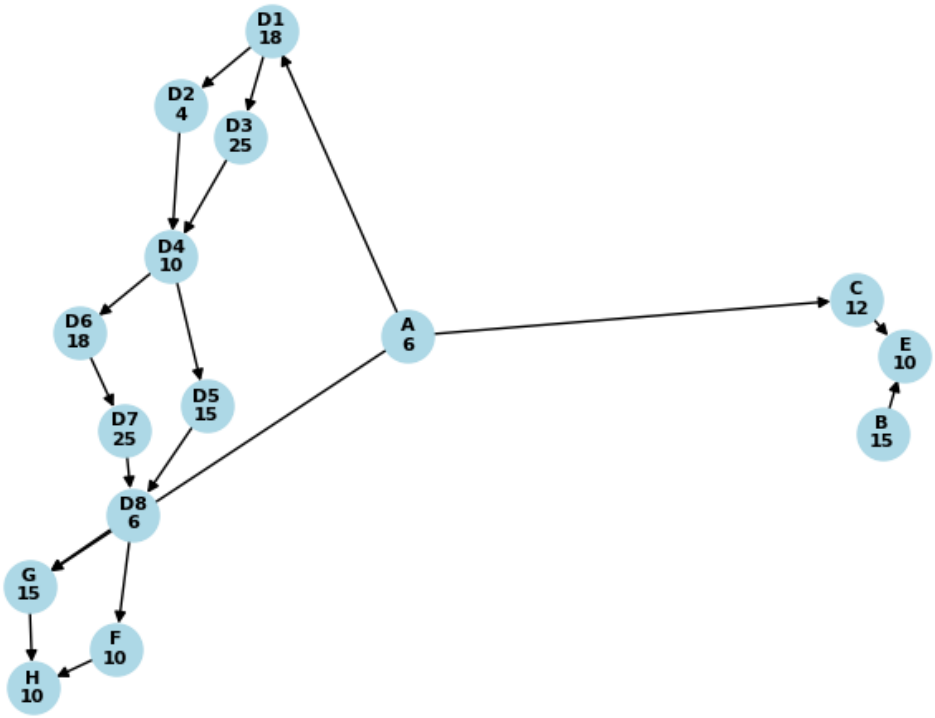

Figure 4: Best-Case Directed Graph Diagram

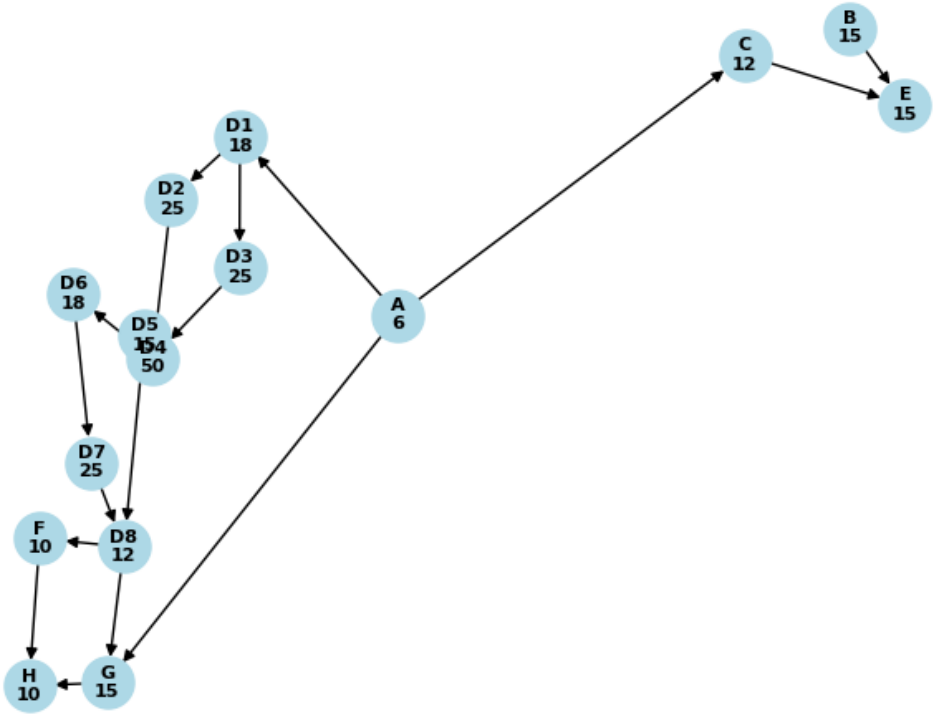

Figure 5: Expected-Case Directed Graph Diagram

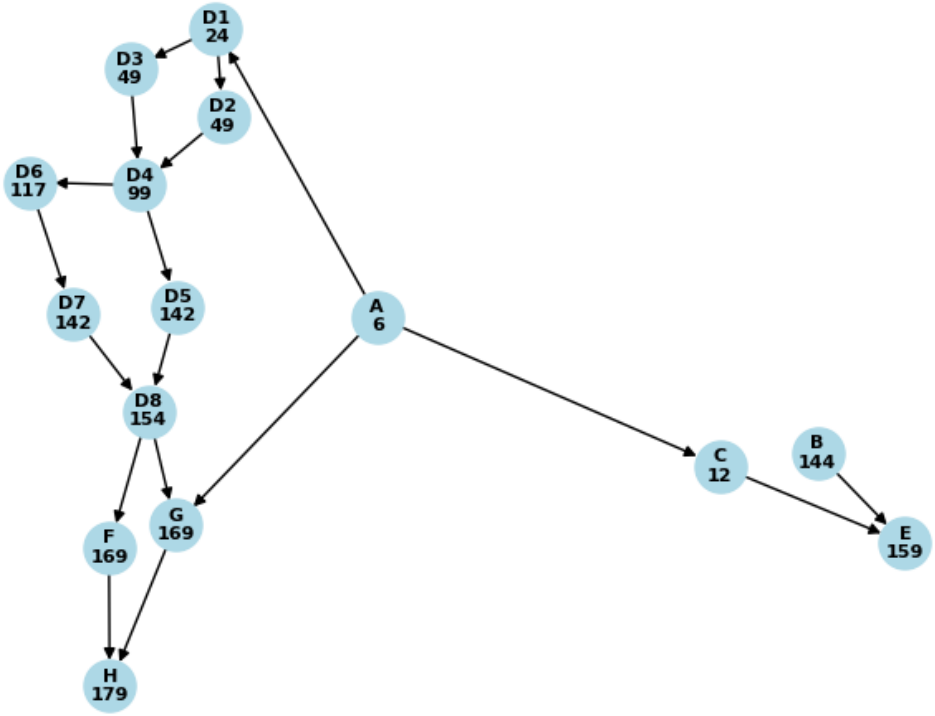

Figure 6: Worst-Case Directed Graph Diagram

8.4 Code

```
from pulp import LpVariable, LpProblem, LpMinimize, LpStatus, value, lpSum

def solve_project_plan(durations, scenario_name):
    # Create the LP problem
    lp_problem = LpProblem(f"Minimize_Project_Time_{scenario_name}", LpMinimize)

    # Define decision variables for task times
    time_vars = LpVariable.dicts("Task_Time", durations.keys(), lowBound=0)

    # Objective function: Minimize total time
    lp_problem += lpSum(time_vars[task] for task in durations), "Total_Time"

    # Adding constraints based on task dependencies
    lp_problem += time_vars['C'] >= time_vars['A'] + durations['A'], "C_A_Constraint"
    lp_problem += time_vars['D1'] >= time_vars['A'] + durations['A'], "D1_A_Constraint"
    lp_problem += time_vars['D2'] >= time_vars['D1'] + durations['D1'], "D2_D1_Constraint"
    lp_problem += time_vars['D3'] >= time_vars['D1'] + durations['D1'], "D3_D1_Constraint"

    # Constraints for D4
    lp_problem += time_vars['D4'] >= time_vars['D2'] + durations['D2'], "D4_D2_Constraint"
    lp_problem += time_vars['D4'] >= time_vars['D3'] + durations['D3'], "D4_D3_Constraint"

    lp_problem += time_vars['D5'] >= time_vars['D4'] + durations['D4'], "D5_D4_Constraint"
    lp_problem += time_vars['D6'] >= time_vars['D4'] + durations['D4'], "D6_D4_Constraint"
    lp_problem += time_vars['D7'] >= time_vars['D6'] + durations['D6'], "D7_D6_Constraint"

    # Constraints for D8
    lp_problem += time_vars['D8'] >= time_vars['D5'] + durations['D5'], "D8_D5_Constraint"
    lp_problem += time_vars['D8'] >= time_vars['D7'] + durations['D7'], "D8_D7_Constraint"

    # Constraints for E
    lp_problem += time_vars['E'] >= time_vars['B'] + durations['B'], "E_B_Constraint"
    lp_problem += time_vars['E'] >= time_vars['C'] + durations['C'], "E_C_Constraint"

    lp_problem += time_vars['F'] >= time_vars['D8'] + durations['D8'], "F_D8_Constraint"
    lp_problem += time_vars['G'] >= time_vars['A'] + durations['A'], "G_A_Constraint"
    lp_problem += time_vars['G'] >= time_vars['D8'] + durations['D8'], "G_D8_Constraint"
    lp_problem += time_vars['H'] >= time_vars['F'] + durations['F'], "H_F_Constraint"
    lp_problem += time_vars['H'] >= time_vars['G'] + durations['G'], "H_G_Constraint"

    # Solve the problem
    lp_problem.solve()
```

8.4.1 Best Case Output

```
Results for Best_Case:
Status: Optimal
Total Project Completion Time: 123.0 hours
Completion time for task A: 0.0 hours
Completion time for task B: 0.0 hours
Completion time for task C: 6.0 hours
Completion time for task D1: 6.0 hours
Completion time for task D2: 24.0 hours
Completion time for task D3: 24.0 hours
Completion time for task D4: 49.0 hours
```

```
Completion time for task D5: 59.0 hours
Completion time for task D6: 59.0 hours
Completion time for task D7: 77.0 hours
Completion time for task D8: 102.0 hours
Completion time for task E: 18.0 hours
Completion time for task F: 108.0 hours
Completion time for task G: 108.0 hours
Completion time for task H: 123.0 hours

Critical Path: ['A', 'D1', 'D3', 'D4', 'D6', 'D7', 'D8', 'G', 'H']
Total Project Duration on the Critical Path: Best Case is 133 hours
```

8.4.2 Expected Output

```
Results for Expected_Case:
Status: Optimal
Total Project Completion Time: 169.0 hours
Completion time for task A: 0.0 hours
Completion time for task B: 0.0 hours
Completion time for task C: 6.0 hours
Completion time for task D1: 6.0 hours
Completion time for task D2: 24.0 hours
Completion time for task D3: 24.0 hours
Completion time for task D4: 49.0 hours
Completion time for task D5: 99.0 hours
Completion time for task D6: 99.0 hours
Completion time for task D7: 117.0 hours
Completion time for task D8: 142.0 hours
Completion time for task E: 18.0 hours
Completion time for task F: 154.0 hours
Completion time for task G: 154.0 hours
Completion time for task H: 169.0 hours

Critical Path: ['A', 'D1', 'D2', 'D3', 'D4', 'D6', 'D7', 'D8', 'G', 'H']
Total Project Duration on the Critical Path: Expected Case is 179 hours
```

8.4.3 Worst Case Output

```
Results for Worst_Case:
Status: Optimal
Total Project Completion Time: 760.0 hours
Completion time for task A: 0.0 hours
Completion time for task B: 0.0 hours
Completion time for task C: 6.0 hours
Completion time for task D1: 6.0 hours
Completion time for task D2: 30.0 hours
Completion time for task D3: 30.0 hours
Completion time for task D4: 79.0 hours
Completion time for task D5: 178.0 hours
Completion time for task D6: 178.0 hours
Completion time for task D7: 295.0 hours
Completion time for task D8: 437.0 hours
Completion time for task E: 144.0 hours
Completion time for task F: 591.0 hours
Completion time for task G: 591.0 hours
Completion time for task H: 760.0 hours
```

```
Critical Path: ['A', 'D1', 'D2', 'D3', 'D4', 'D6', 'D7', 'D8', 'F', 'G', 'H']
Total Project Duration on the Critical Path: Worst Case is 939 hours
```