2023

CS5002 Final Project Report

OPTIMIZING PROJECT MANAGEMENT USING GRAPH THEORY

QIUYING ZHUO ZHUO.QI@NORTHEASTERN.EDU
ZHIWEI ZHOU ZHOU.ZHIWE@NORTHEASTERN.EDU

Abstract

This project utilizes graph theory to analyze a Directed Acyclic Graph (DAG)-type project and identify its critical path, minimum completion time, and tasks that can be delayed without affecting the overall project timeline. The project's final step generates a summarized message that provides users with a clear and concise overview of the critical path and its total duration. The critical path represents the sequence of tasks that determines the minimum completion time for the project, and any inaccurately timed task on this path can result in a delay in project completion. Tasks not on the critical path have some flexibility in their scheduling and can be delayed without affecting the overall project timeline. The insights from this analysis can help users optimize their time-sensitive DAG-type projects, allocate resources more efficiently, manage risks better, and ultimately achieve more successful project outcomes.

The project's limitations include its focus on acyclic task dependencies, while some project management scenarios can have cyclic dependencies that are not represented. Future research could extend the model to incorporate cyclic dependencies, allowing for a more comprehensive analysis of different project management scenarios. Additionally, the project's code implementation could be optimized by refining the approach to find the longest path and exploring more efficient algorithms, enhancing the scalability and performance of this script. The project also highlights the importance of efficient collaboration, where tools such as Miro Board, GitHub workflow, Word Online, GitHub, Jupyter Notebook were utilized.

Overall, the project aims to provide valuable insights and practical knowledge on managing tasks when faced with tight deadlines, unexpected tasks, or limited resources. It is an excellent learning experience in terms of technical knowledge and collaboration skills, which can be applied in future endeavors.

Table of Contents

Introduction: Context of the Chosen Topic

Rather than performing routine tasks, more and more jobs today involve completing a project within a specific timeframe. For example, in the architectural design and real estate development industries, professionals need to manage various projects such as building design, construction, and renovation, which require careful planning and coordination. Similarly, professionals in banking, law, and auditing often work on projects that involve multiple stakeholders, deadlines, and deliverables. Moreover, with the rise of technology and digital transformation, project-based work has become even more prevalent. For instance, software development professionals need to manage complex projects with tight deadlines, where a delay in one task can have a cascading effect on the entire project timeline.

With the increasing prevalence of project-based work across various industries, project management has become crucial for professionals to possess project management skills to deliver projects on time, within budget, and with the desired quality.

As a part of our coursework in the CS5002 Discrete Structure class, we have been introduced to graph theory, a topic with potential applications in project management. We have chosen to explore this subject further to understand how leveraging graph theory can improve the efficiency of our future projects.

The extensive use of Graph Theory in project management

Graph theory can be used to optimize project management in a variety of ways, including:

- ***Scheduling projects:*** Graph theory can be used to model the dependencies between tasks in a project, and to identify the critical path, which is the longest sequence of tasks that must be completed to complete the project. This information can be used to create a schedule for the project that minimizes the time required to complete it.
- ***Allocating resources to tasks:*** Graph theory can be used to model the availability of resources, such as people, equipment, and materials, and to assign resources to tasks in a way that minimizes costs and maximizes efficiency.
- ***Optimizing project budgets:*** Graph theory can be used to model the costs of different project options, and to identify the option that minimizes costs while meeting the project's objectives.

Personal investment of each group member in the topic

***Relevance to Qiuying*:** Prior to joining the CS-Align program, Qiuying worked as an Investment Banker specializing in The Telecommunications Media & Technology (TMT) sector. Her responsibilities primarily revolved around time-sensitive IPO (Initial Public Offerings) and M&A (Mergers & Acquisitions) projects, necessitating effective cross-team and cross-company coordination. Efficient work arrangements were essential to avoid missed deadlines or deal failures.

***Relevance to Zhiwei*:** Before enrolling in the CS-Align program, Zhiwei held positions as an architectural designer and a real estate development professional. Project management is a vital aspect of the architectural design industry, encompassing the management of diverse design phases, permit applications, construction schedules, and administrative tasks. Moreover, several precedents must be taken into account before initiating specific tasks within the overall development process.

The broader importance beyond personal experience

Project management's significance transcends individual experiences and is highly advantageous for other CS Align students. As project-based work gains prominence across various sectors, it becomes imperative for CS students to hone their project management skills to effectively oversee tasks, timelines, and resources. By employing graph theory to optimize project management, this project aims to deliver valuable insights and strategies that will enhance Qiuying, Zhiwei, and other CS Align students' ability to manage projects and multitask, regardless of their chosen industry.

Clearly Defined Research Question and Scope

While graph theory can be used to optimize project management in various ways, in this report we will focus our attention on specific project management scenarios that can be represented as **Directed Acyclic Graphs (DAGs)**, which is a commonly seen project type we had experienced our previous works. DAG projects involve tasks with dependencies, where tasks must be completed in a specific order without any cycles. Examples of such projects include construction projects, software development projects, and event planning. In contrast, other scenarios that do not fit the DAG model might involve cyclic dependencies or tasks that can be executed in any order.

For the purpose of demonstrating the application of graph theory in project management, we will use a hypothetical example that people can easily understand. This example involves organizing a promotional event with a set of interconnected tasks that must be completed within a specific time frame, and further details will be documented in the next chapter.

Thus, the clearly defined research question for this project is: **How can Graph Theory be applied to optimize DAG-type projects, especially those time-sensitive ones, in various industries?** Using the promotional event example, we aim to address the following questions:

1. What is the earliest completion time for the project, and how can it be found?
2. Which activities can be delayed, and by how long, without affecting the minimum completion time?

By examining these aspects of project management through the lens of graph theory, we hope to offer valuable insights and strategies for optimizing project management across various industries.

Approach and Theory Behind

Incorporating graph theory in project management

Our approach in understanding the application of graph theory in project management is to relate the theoretical concepts we've learned in class to the practical aspects of managing projects. We will start by discussing the critical path method, which has been extensively covered in the Harvard Business Review article, and further explore the connection between the critical path method and graph theory.

The critical path and its relation to graph theory

The **critical path method (CPM)** is a project management technique that helps identify the longest sequence of tasks that must be completed for a project to be finished on time. In a project graph, nodes represent tasks, and edges represent the dependencies between tasks. The critical path is the longest path in the graph, and its length determines the minimum completion time for the project.

The project graph below cited from the same HBR article provides a visual representation of the home construction process as a directed acyclic graph (DAG). This example demonstrates how a real-world project can be represented using graph theory concepts.

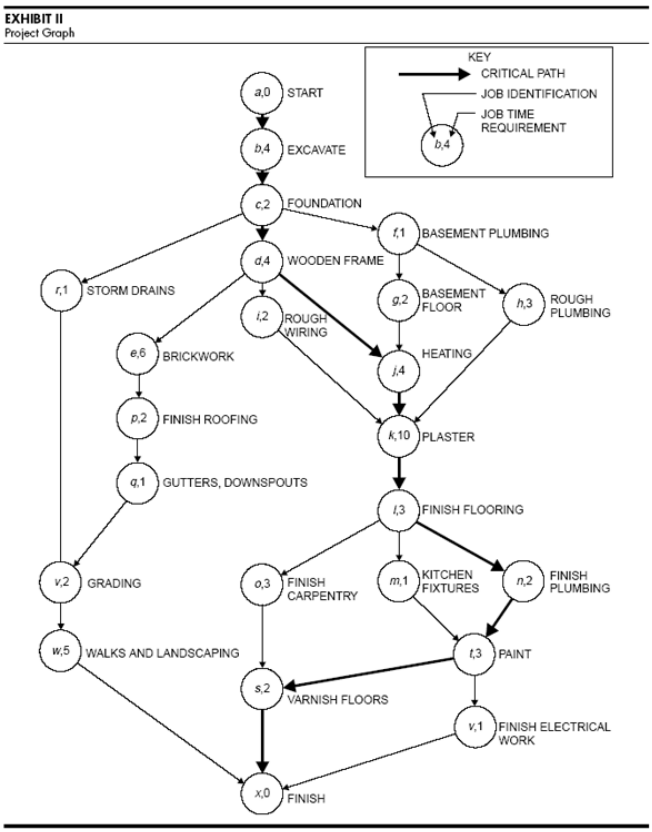

Figure 1 Home Construction Process as a DAG (from HBR)

Finding the longest path (the critical path): Explained

The relationship between graph theory and project management is evident when considering the critical path method and its application to DAGs. In our CS5002 Discrete Structure class, we've learned about finding simple paths. A simple path is a path in a graph with no repeated nodes or edges, meaning that it visits each node only once. This concept is crucial in project management, as it ensures that tasks are not duplicated or performed out of order. We've also learned **Dijkstra's algorithm** to find the shortest path.

In the context of this project, in theory, we could apply these concepts in reverse (by modifying Dijkstra's algorithm to prioritize longer paths over shorter ones) to find the critical path in a DAG. However, in practice, we find it challenging to do so in Python by leveraging existing functions for shortest paths in NetworkX (such as shortest_path, all_shortest_paths) and negate the weights (representing duration or days) to find the longest path from the starting to the end node. The reason is that our target is a graph with the weights attached to the nodes as labels, while a typical graph for Dijkstra's algorithm is directed with weights attaching to the edges, NOT the nodes. See below the example of a graph extracted from CS5002 HW10 Final Practice with weights attached to the edges instead of the nodes (for this project).

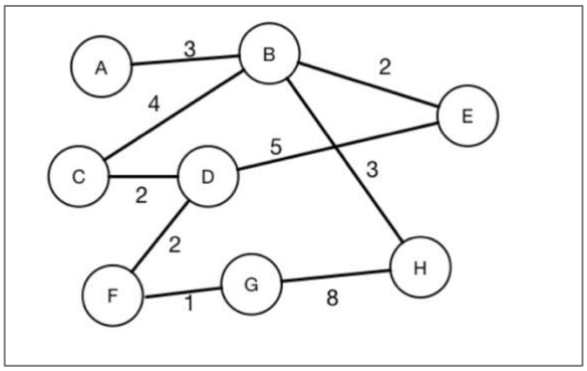

Figure 2 Weighted Graph (from CS5002 HW10 Final Practice)

As such, our workaround solution is to leverage the built-in dictionary in Python and the all_simple_paths function in NetworkX. First, we find all the simple paths. Then, we traverse each simple path. For each path, while visiting each node, we sum the required days stored in the dictionary as the id-duration pair (representing each task and its associated required days, respectively). As such, after looping through all simple paths with the designated start and end nodes, we can determine which path has the longest duration, being the critical path.

This approach allows us to incorporate the concepts of simple paths and longest paths learned in our CS5002 Discrete Structure class into project management practices. By identifying the critical path, we can better understand project dependencies, allocate resources more efficiently, and minimize the risk of delays.

Applying DAG Analysis to a Promotional Event: A Case Study

To fully understand and explore the case study presented in this section, we recommend reading this section together with the ***CS_5002_Project_Report_Code_Explanation_QZ_ZZ.ipynb*** file which includes the corresponding code and printout.

Step 0: Construction of the promotional event scenario

In this case study, we assume the role of an employee tasked by their manager to urgently prepare a promotional event for the company within a challenging 30-day timeframe. The manager expects us to focus on a range of tasks essential for the event's success, such as selecting products, designing advertisements, and distributing promotional materials (see below table). For the sake of this example, we will not concern ourselves with other events such as finding a suitable venue, setting the event date, or coordinating with attendees. However, upon estimating the required duration for each task, we discovered that the project would require a total of 39 days to complete, which exceeds the allocated timeframe of 30 days. At first sight, this seemed like a mission impossible!

Table 1 Promotional Event: To Do List

Item	Description	Duration
A	Plan the event (Event Coordinator)	5
B	Select products (Purchaser)	4
C	Design promotional materials (Graphic Designer)	8
D	Coordinate with suppliers (Purchaser)	5
E	Prepare email content (Marketing)	4
F	Print promotional materials (Printer)	4
G	Assemble email campaign (Marketing)	3
H	Distribute promotional materials (Retail Staff)	3
I	Launch email campaign (Marketing)	1
J	Execute promotional event (Retail Staff)	2
Total Duration (in days)		**39**

The primary challenge is identifying opportunities for parallelization and optimization, ensuring that the event can be prepared within the given time constraints. We approach this challenge tentatively, adopting a step-by-step method. First, we need to break down the tasks, assess their dependencies, and estimate their durations. Presented below is the list of tasks for the promotional event, including the item, description, duration, and preceding works. We use these assumptions to understand the scope of work and evaluate opportunities for optimization:

Table 2 Promotional Event: To Do List Revisited

Item	Description	Duration	Preceding Works
A	Plan the event (Event Coordinator)	5	
B	Select products (Purchaser)	4	
C	Design promotional materials (Graphic Designer)	8	A, B
D	Coordinate with suppliers (Purchaser)	5	C
E	Prepare email content (Marketing)	4	C
F	Print promotional materials (Printer)	4	D, E
G	Assemble email campaign (Marketing)	3	C
H	Distribute promotional materials (Retail Staff)	3	F, G
I	Launch email campaign (Marketing)	1	G
J	Execute promotional event (Retail Staff)	2	H, J

Step 1: Process input data

To enable our further analysis using Python, we must first process the input data in a structured format. Our choice to use the CSV file format is rooted in its simplicity, ease of processing, and potential compatibility with Google Forms and other data processing tools.

The first step involves reading and processing the data from the CSV file. We initialize dictionaries and an **edge_list** to store the processed data. To simplify our graph structure and analysis, we introduce virtual start and end nodes, which serve as placeholders for the beginning and end of the project timeline. These virtual nodes act as the common starting and ending points for all tasks in the project. By adding a virtual start node, we can easily connect it to all tasks that do not have any prerequisites, effectively establishing a single entry-point for the project. Similarly, the virtual end node is connected to all tasks that do not have any successors, providing a single exit point for the project. By incorporating virtual start and end nodes in our Python-based critical path analysis, we can more accurately represent the project structure and streamline the process of finding the critical path. See below figures for illustration.

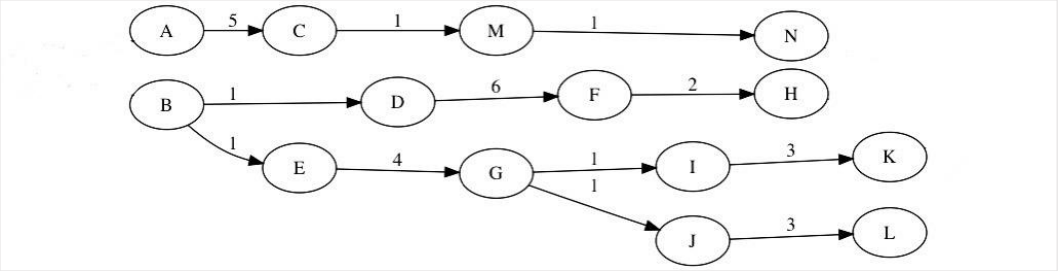

Figure 3 Without adding virtual points, a project may have multiple start and end points

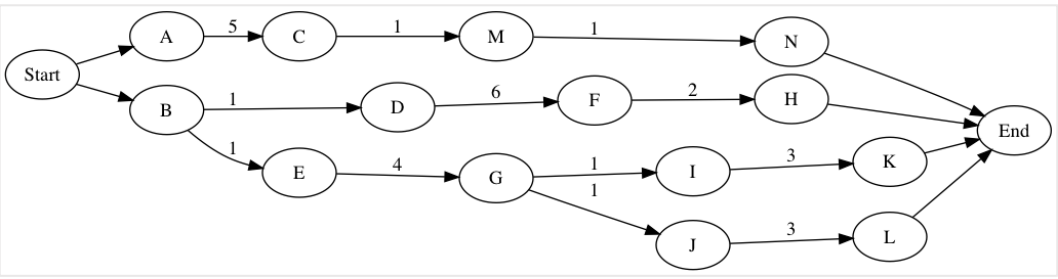

Figure 4 Adding virtual-start/virtual-in and virtual-end/virtual-out points helps to wrap up the project with unique start and end points, allowing for further exploration of the longest path (These two figures are extracted and adapted from [What is the 'critical path' when drawing an activity-on-node network diagram that doesn't converge?](#)*)*

Next, we iterate through each row in the CSV file, extracting essential information such as the node label, description, duration, and preceding works. For each task, we update the **id_to_name_dict** with the task's node label and duration, **id_duration_dict** with the node label and its duration, and **id_description_dict** with the node label and description.

We then create directed edges between tasks based on their dependencies, updating the **edge_list** accordingly. If a task has no preceding tasks, we add an edge from the virtual start node to the task. For tasks with preceding nodes, we add edges connecting the task to its dependencies. Finally, we add edges from tasks without successors to the virtual end node.

Once the input data is processed, we have the necessary information stored in dictionaries and the **edge_list**, which allows us to further analyze and optimize the project using graph theory.

Step 2: Represent the graph using NetworkX

In this step, we represent the project's task dependency graph using the NetworkX library recommended by our Professor. NetworkX can work with complex graphs and networks, making it an ideal choice for our project. The primary objective of this step is to create a directed graph object (**DiGraph**) and add the edges from the previously created **edge_list**.

To begin, we instantiate a new directed graph object G and add edges to it using the **add_edges_from** method. This function takes the **edge_list** as input and adds each edge to the graph. Since we are working with a DAG, it is essential to ensure that our graph G is a valid DAG. We use the **is_directed_acyclic_graph** method to check if the graph is a valid DAG. This function returns a Boolean value, which is True if the graph is a valid DAG and False otherwise.

Finally, we print the graph representation information to verify that our graph has been correctly constructed. The output shows that our graph G is a DiGraph with 12 nodes and 15 edges and confirms that it is a valid directed acyclic graph. By representing the graph using NetworkX, we can now leverage the library's extensive functionalities to analyze the graph and find the critical path.

Step 3: Find the critical path / the longest path

In this step, we aim to find the critical path, which is the longest path in our DAG. As a recap, the critical path represents the sequence of tasks that take the longest time to complete, determining the minimum project duration. To accomplish this, we implement a brute-force approach using the NetworkX library.

First, we define the **find_longest_path** function that takes a graph, a start node, and an end node as its arguments. Within this function, we initialize the **longest_path** as an empty list and the **longest_length** as negative infinity. We then iterate through all simple paths between the start and end nodes, using the **all_simple_paths** function. This function returns a generator of all simple paths in the graph, allowing us to efficiently process each path. The time complexity of this function is O(|V|+|E|) for each path generated, where |V| is the number of nodes and |E| is the number of edges in the graph.

For each simple path, we calculate its length by summing the durations of its nodes using the **id_duration_dict**. If the length is greater than the current **longest_length**, we update the **longest_length** and the **longest_path** accordingly. Once all simple paths have been processed, the function returns the **longest_path** and its **longest_length**.

Next, we call the **find_longest_path** function with our graph **G**, **start_node**, and **end_node** to determine the longest path and its length. We then update our data structures by removing the virtual nodes and edges associated with them. This removal is essential to prepare the list for formatting and plotting using matplotlib, as we want the final chart to represent the actual project tasks without the virtual points.

Finally, we print the critical path information, including the longest path, its length, and its edges. By finding the critical path in our DAG, we can effectively identify the sequence of tasks that determine the minimum project duration and optimize our project management process accordingly.

Step 4: Format and plot

Next, we focus on visualizing the DAG representing our project tasks and the critical path. We use the Matplotlib library to create a visually appealing and informative plot.

First, we create a new figure using **figure** function in the library. Next, we generate the node positions for our graph using the **shell_layout** function. We chose the shell layout after experimenting with several layout options, such as spring and spectral, and found that it provided the best visualization for our purposes. This function returns a dictionary of node positions, which we then modify by inverting the y-coordinates to achieve our desired layout.

We then define the node and edge colors for our plot. Nodes belonging to the critical path are colored red, while other nodes are colored steelblue. Similarly, edges belonging to the critical path are colored red, and all other edges are colored grey.

With the layout and styling defined, we proceed to draw the DAG using the **draw** function. We pass the necessary parameters, such as the graph **G**, node positions **pos**, node and edge colors, as well as other visual properties like label colors and node size. After drawing the graph, we save the resulting plot as a PNG file using the **savefig** function. We then display the plot using **show** function (see below) and clear the current figure with **clf** to prepare for any subsequent plots.

Lastly, we print information about the generated plot, including the file name and storage location. By visualizing our project's DAG and the critical path, we can better understand the project structure and identify areas for potential optimization.

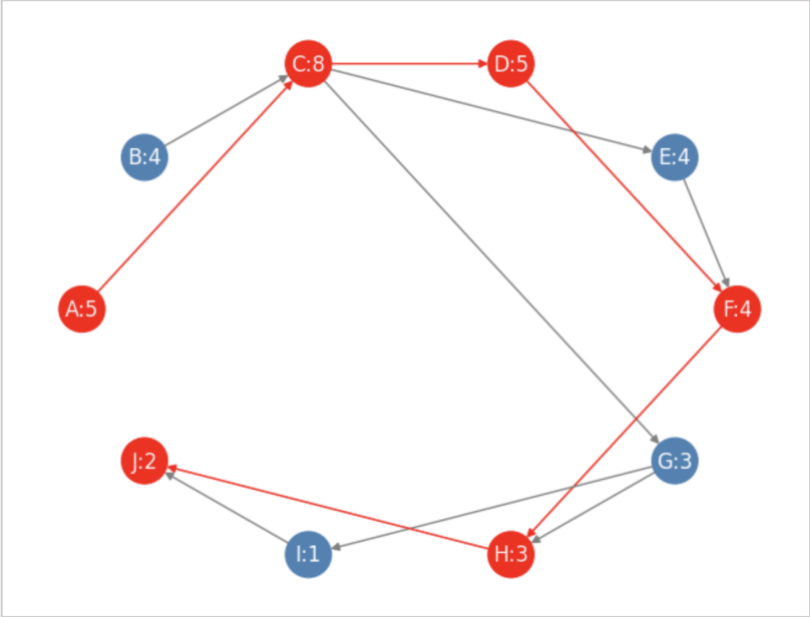

Figure 5 Critical Path for the Promotional Event

Step 5: Generate summarized message

In the fifth and final step of our project, we generate a summarized message to provide a clear and concise overview of the critical path and its total duration. This summary helps stakeholders understand the project timeline and the essential tasks that determine its overall duration.

We create a function called **generate_summary** that returns the completed summary string, which is printed as the final output of our program. Below is an example of the summary output. By providing a clear and concise summary of the critical path, we offer valuable information to project stakeholders that can be used to optimize project management and resource allocation.

```
SUMMARY

The critical path consists of the following tasks:
1. A: Plan the event (Event Coordinator) (5 days)
2. C: Design promotional materials (Graphic Designer) (8 days)
3. D: Coordinate with suppliers (Purchaser) (5 days)
4. F: Print promotional materials (Printer) (4 days)
5. H: Distribute promotional materials (Retail Staff) (3 days)
6. J: Execute promotional event (Retail Staff) (2 days)

The total duration of the critical path is 27 days.

REFERENCE

The tasks NOT on the critical path are:
B: Select products (Purchaser) (4 days)
E: Prepare email content (Marketing) (4 days)
G: Assemble email campaign (Marketing) (3 days)
I: Launch email campaign (Marketing) (1 days)
```

Figure 6 Summary Output for the Promotional Event

Analysis: Interpretation of the Output

The output of our script, as shown in the previous section, provides valuable insights into the critical path and the minimum completion time of the project. In the context of the given promotional event scenario, this information is crucial for meeting the challenging 30-day deadline set by the manager.

The critical path represents the sequence of tasks that determines the minimum completion time for the project. In this case, the total duration of the critical path is 27 days, which means that the promotional event project cannot be completed in less than 27 days. This result is promising, as it falls within the 30-day timeframe.

An extended interpretation of the critical path implies that if any task is inaccurately timed and taking longer than expected and use the up the 3-day allowance, the overall project may not be completed on time. Furthermore, if there's a need to shorten the total project time, we could start by focusing on the tasks on the critical path, as these are the primary determinants of the overall project duration.

To address the second part of our research question, which involves identifying tasks that can be delayed without affecting the minimum completion time, we can compare the tasks on the critical path with those not on the path. Tasks not on the critical path have some flexibility in their scheduling. For example, task B (select products) can be delayed by 1 day, task E (prepare email content) can be delayed by 1 days, task G (assemble email campaign) can be delayed by 6 days, and task I (launch email campaign) has some flexibility too: depending on when task G is finished, I can be delayed for a minimum of 2 days.

Understanding the flexibility of non-critical tasks also allows the stakeholders to accommodate any last-minute changes or unforeseen events without jeopardizing the project timeline. For instance, resources can be reallocated from non-critical tasks to critical ones in case of unexpected delays or issues.

Together, these insights can be used by the professionals to make informed decisions on resource allocation and task prioritization.

Conclusion: Reflections and Future Directions

By applying graph theory to our promotional event example, we have successfully identified the critical path and its minimum completion time, as well as tasks that can be delayed without affecting the overall project timeline. This information can be utilized by stakeholders across various industries to optimize their time-sensitive DAG-type projects, leading to more efficient resource allocation, better risk management, and ultimately, more successful project outcomes.

Throughout our project, efficient collaboration was essential. We used Miro Board during brainstorming to refine our focus and select a topic. Code collaboration proved challenging as it was our first time coding together. We initially tried VSCode Live Share, but its limitations led us to adopt GitHub workflow with our preferred IDEs and GitHub Desktop for easier progress tracking and code review. For the project report, we used Word Online for text editing and plan to integrate text and code in Jupyter Notebook for a dynamic presentation.

Weaknesses, limitations, and future study directions

This project deals with DAG-type projects which are acyclic. However, not all project management scenarios can be represented as DAGs. There can be situations where the task dependencies in project management form cycles, which are not covered in the current study. For example, in agile or iterative software development methodologies, tasks can overlap or be revisited, creating cyclic dependencies (Requirement Analysis → Design → Coding → Testing → Deployment → (Feedback) → Requirement Analysis). Future research could focus on extending the current model to incorporate cyclic dependencies, allowing for a more comprehensive analysis of different project management scenarios.

Meanwhile, even when dealing with tasks on the critical path, we might need to consider more factors than just estimating a duration. This would at least include early start time, early end time, late start time, and late end time, as indicated in this HBR article. This would allow the responsible parties to understand the total slack (maximum time that could be delayed without delaying the whole project) for unexpected events. Further research could explore the integration of these additional factors into the model, enhancing its applicability and utility in real-world project management contexts.

In terms of code implementation, there are some limitations due to our limited knowledge in graph theory, algorithms, and programming. For example, we leveraged all_simple_paths method from NetworkX and traversed each path to find the longest path. This is a simple but brute-force approach that is not the most efficient from a Big-O perspective. In the future, we could refine this. Also, the overall script may not have been properly packaged per the object-oriented programming guidelines, which is something we wish to improve in the near future after taking CS5004 OOP classes. Future work could focus on optimizing the code and exploring more efficient algorithms, ultimately improving the scalability and performance of our program.

NetworkX is a powerful Python toolset for data visualization, but our team lacks advanced knowledge and skills in using the module. Our current graph could be improved with more comprehensive labels and distributed flows (we only used shell layout and spring layout for the graph). Although we only printed step descriptions on the terminal, we plan to explore more features and make further improvements in the future. The graph maker will not be limited to NetworkX. Thanks to the great open-source community, there are many tools to try.

Key takeaways: Reflections for Zhiwei

This project taught me that it is not enough to simply learn textbook knowledge, but it is equally important to apply it to real-world problems, which is the essence of computer science. Graph theory was a completely new concept to me, but through this project, I was able to gain a thorough understanding of its basics and how to apply it to practical problems. The effort we put into the project paid off, and we were able to produce a comprehensive project report.

Additionally, I gained valuable experience using new tools such as NetworkX, matplotlib, and Jupyter Notebook. Although it was the first time working with these tools, I believe that they will be useful in my future studies as my programming and algorithmic knowledge continues to grow.

Collaboration played a crucial role in the success of our project. My teammate Qiuying and I come from different academic backgrounds, which we saw as a strength that allowed us to apply mathematical theories to different areas and explore new possibilities. Throughout our collaboration, we adjusted the tools and scope of the project according to our needs and effectively assigned tasks based on our respective knowledge backgrounds, which improved our efficiency. Overall, our collaboration was smooth, and we had a lot of fun working together.

In conclusion, I am grateful for the challenge and joy that this project brought to me. It helped me realize the charm of applying knowledge to solve real-world problems, and I am confident that I will be able to thoroughly analyze problems and confidently apply my knowledge to solve them in my future studies and work.

Key takeaways: Reflections for Qiuying

In this project, I gained valuable knowledge and experience in Graph Theory, which was a completely new concept to me. I now have a clearer understanding of how to structure a proper graph, as demonstrated in our promotional event example. This knowledge will be helpful for my future studies at Northeastern.

I also learned about new tools such as NetworkX, matplotlib, and Jupyter Notebook. Although my proficiency with these tools is still in its early stages, I believe they will remain valuable as my programming and algorithm knowledge grow through my future studies.

Collaboration was a key highlight. I had a smooth experience working with my team member Zhiwei, who was always open to discussion and trying new things. Our brainstorming sessions were productive, and our openness to new tools contributed to the project's efficient completion. Together, we navigated potential topics before settling on the promotional event scenario. We sought resources to learn graph drawing and formatting. Throughout the coding process, we engaged in back-and-forth discussions and reworks to refine our code. Zhiwei was highly receptive to new tools, and we quickly decided to use Jupyter Notebook to showcase our code. I credit my partner for the project report's timely completion and the enjoyable collaboration process.

The project provided practical insights on the critical path and tasks that can be delayed, which is crucial for managing tasks when faced with tight deadlines, unexpected tasks, or limited resources. This valuable experience will undoubtedly serve me well in my future endeavors at Northeastern and beyond.

In conclusion, this project was a great learning experience, both in terms of technical knowledge and collaboration skills. I am grateful for the opportunity to work with Zhiwei and to gain practical insights into project management.

References

- The ABCs of the Critical Path Method from Harvard Business Review
- Building DAGs / Directed Acyclic Graphs with Python from MungingData
- What is the "critical path" when drawing an activity-on-node network diagram that doesn't converge
- Notebook 2.2- Weighted and directed graphs
- HiLite.me to insert code snippet in the word document
- Customizing NetworkX Graphs by Aren Carpenter

Appendix

Supplement files for the report

Please refer to the attached files:

- ***CS_5002_Project_Report_Code_QZ_ZZ.py*** for the Python code
- ***CS_5002_Project_Report_Code_Explanation_QZ_ZZ.ipynb*** for the Jupyter Notebook code with more detailed explanations

Note that to run the code, please ensure that the "event_planning.csv" file is in the same directory as the script. If you want to use a different file, please change the file name at the top of the script accordingly.

Real-life examples from our work experience

To demonstrate the real-life applicability of this project's analysis and to relate it to our previous work experience, we have attached two real-life examples in the appendix for reference. These examples will help illustrate how our approach can be applied to optimize workflows and resource allocation across different industries and scenarios.

Ex. 1: Hong Kong IPO Project

This example showcases an IPO project's schedule, demonstrating the application of our approach in optimizing workflows and resource allocation for large projects with multiple steps.

Table 3 Typical Schedule for IPO in Hong Kong Main Board

Item	Description	Duration	Preceding Works	Party
A	Preparation of historical financial statements	15		Auditors
B	Preparation of financial projections	15		Auditors
C	Internal legal due diligence	15		Company's Legal Counsel
D	External legal due diligence	25	C	External Lawyers
E	Internal operational due diligence	15		Company's Management
F	External operational due diligence	25	E	Industry Advisor
G	Industry analysis and market research	35		Industry Advisor
H	Preparation of business plan	18	B, G	Company's Management
I	Drafting of the prospectus	35	D, F, H	Sponsor & External Lawyers
J	Review of financial statements and prospectus	15	A, I	Auditors
K	Finalizing the prospectus	5	J, I	Sponsor & External Lawyers
L	Filing the prospectus with the HK Stock Exchange	1	K	Sponsor
S	Q&A with Hong Kong Stock Exchange	12	L	All parties
T	Q&A with SFC (Securities and Futures Commission)	12	L	All parties
M	Marketing materials preparation	7	H, I	Company's Management
N	Roadshow presentation preparation	7	M	Sponsor & Co's Management
O	Marketing and roadshow	14	L, S, T	Sponsor & Co's Management
P	Book-building and pricing	7	O	Sponsor
Q	Allotment and allocation of shares	3	P	Sponsor
R	Listing and commencement of trading	1	Q	Sponsor

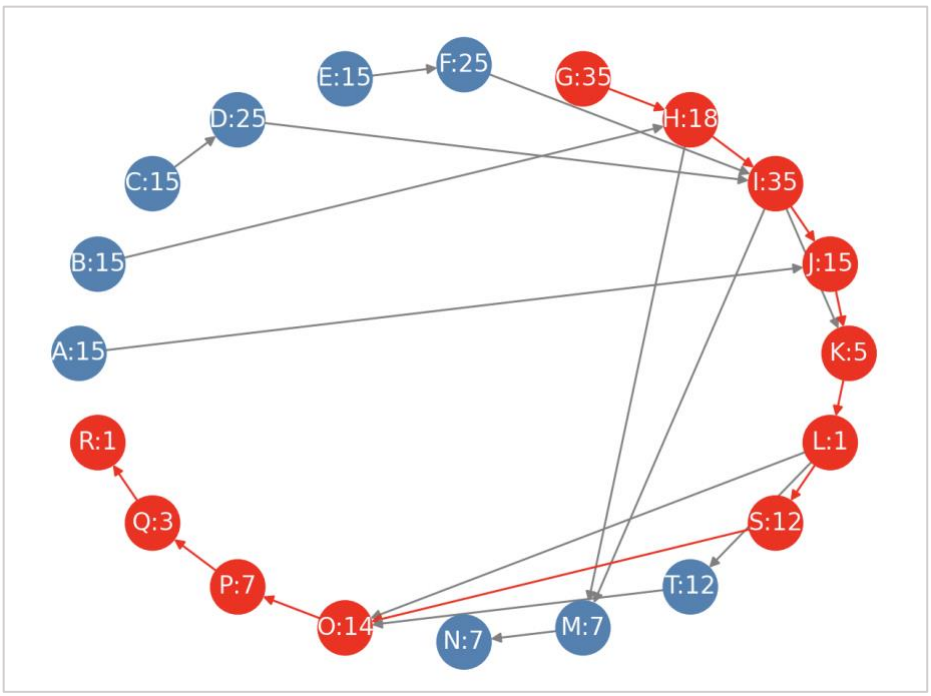

Figure 7 Critical Path for the IPO Project

```
SUMMARY

The critical path consists of the following tasks:
1. G: Industry analysis and market research (35 days)
2. H: Preparation of business plan (18 days)
3. I: Drafting of the prospectus (35 days)
4. J: Review of financial statements and prospectus (15 days)
5. K: Finalizing the prospectus (5 days)
6. L: Filing the prospectus with the Hong Kong Stock Exchange (1 days)
7. S: Q&A with Hong Kong Stock Exchange (12 days)
8. O: Marketing and roadshow (14 days)
9. P: Book-building and pricing (7 days)
10. Q: Allotment and allocation of shares (3 days)
11. R: Listing and commencement of trading (1 days)

The total duration of the critical path is 146 days.

REFERENCE

The tasks NOT on the critical path are:
A: Preparation of historical financial statements (15 days)
B: Preparation of financial projections (15 days)
C: Internal legal due diligence (15 days)
D: External legal due diligence (25 days)
E: Internal operational due diligence (15 days)
F: External operational due diligence (25 days)
T: Q&A with SFC (Securities and Futures Commission) (12 days)
M: Marketing materials preparation (7 days)
N: Roadshow presentation preparation (7 days)
```

Figure 8 Summary Output for the IPO Workstream

Ex. 2: Architectural Design Project

For example, an architectural firm is designing a new office building for a client. The project will involve multiple stages, including:

Table 4 Major Phases for the Architecture Project (Example of an Office Building)

1	Survey
2	Pre-Design
3	Sub-contractors onboarding
4	Schematic Design
5	LEED application
6	Master Plan Review with City
7	Design Development
8	Construction Documents
9	Building Permit
10	Value Engineering
11	Bidding Negotiation
12	Construction Administration

Each of these tasks will take a certain amount of time and may have dependencies on other tasks. For example, the schematic design cannot begin until the concept design is complete, and the permit applications cannot be submitted until the construction drawings are finalized.

These dependencies can be represented as a DAG, with each task represented as a node and the dependencies represented as directed edges. The resulting DAG might look something like this:

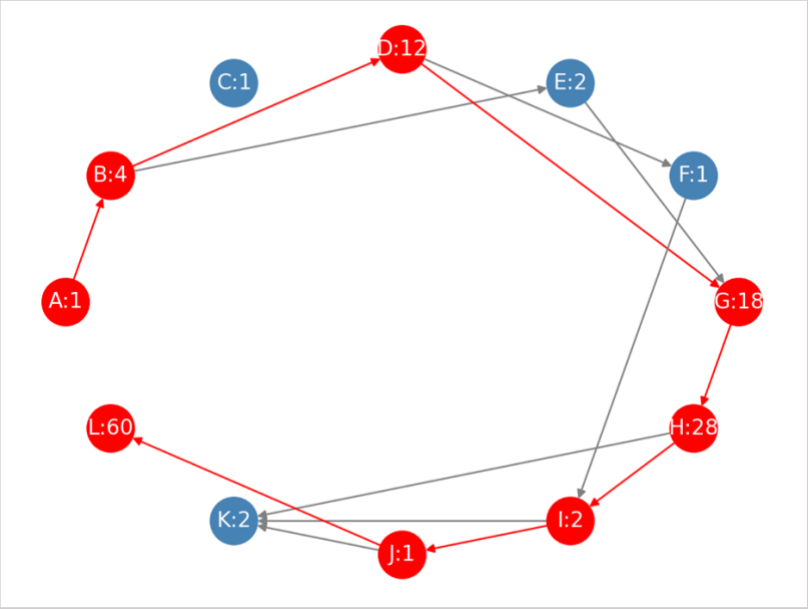

Figure 9 Critical Path for the Architecture Project

In this DAG, each node represents a task, and the directed edges represent the dependencies between tasks. For example, the Concept Design task depends on the Site Survey task, and the Schematic Design task depends on the Concept Designs.

By using a DAG to represent the dependencies between tasks, the architectural firm can better manage the project by identifying the critical path and minimizing the time required to complete the project. They can also use the DAG to assign resources to tasks in a way that maximizes efficiency and minimizes costs.

```
SUMMARY

The critical path consists of the following tasks:
1. A: Survey (1 weeks)
2. B: Pre Design (4 weeks)
3. D: Schematic Design (12 weeks)
4. G: Design Development (18 weeks)
5. H: Construction Documents (28 weeks)
6. I: Building Permit (2 weeks)
7. J: Value Engineering (1 weeks)
8. L: Construction Administration (60 weeks)

The total duration of the critical path is 126 weeks.

REFERENCE

The tasks NOT on the critical path are:
C: Subconstrators onboarding (1 weeks)
E: LEED application (2 weeks)
F: Master Plan Review with City (1 weeks)
K: Bidding Negotiation (2 weeks)
```

Figure 10 Summary Output for the Architecture Project