Data Structures and Algorithms (CSL2020)

INTELLIGENT TASK MANAGER

Mentor TA: Dixit Dutt Bohra
Shubham Kumar

Team members: Chinmay Vashishth (B22BB016)
Shikhar Dave (B22CH032)
Shivam Khanchandani (B22BB038)

PROBLEM STATEMENT

IMPORTANCE:

- An intelligent task manager plays a critical role in organizing individuals' workflows, optimizing productivity, and fostering effective time management.

CHALLENGES:

- The task manager must seamlessly integrate task prioritization, scheduling, and collaboration, while remaining user-friendly and adaptable to diverse user preferences and work styles.

REQUIREMENT OF DATA-DRIVEN SOLUTION:

- The task manager's success hinges on its ability to continuously evolve and improve based on real-world usage data and user feedback.
- Therefore, a data-driven approach is required to not only address the inherent complexities of task management but also to deliver personalized and efficient solutions that meet the evolving needs of users.

CURRENT STATUS

- In professional settings, task managers help individuals and teams manage work-related tasks, projects, deadlines, and meetings.
- In the corporate sector, task managers are essential tools for project managers and teams working on projects of all sizes.

LIMITATIONS OF CURRENT-DAY TASK MANAGERS:

- Task managers may not take into account all the factors affecting the priority of a task due to lack of context awareness.
- Task managers struggle to handle complex task dependencies, such as tasks that are contingent on the completion of others or tasks with multiple subtasks.
- Users may have to manually input tasks and set deadlines, reducing time savings due to automation.

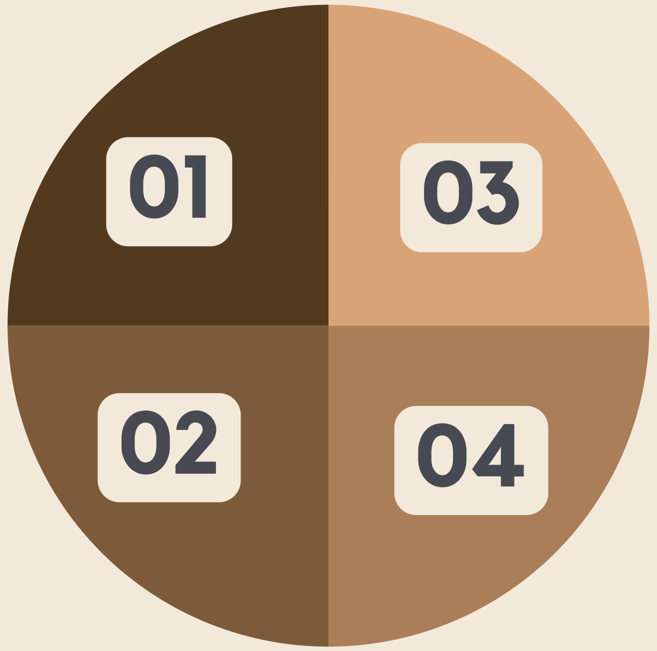
DATA STRUCTURES

01 GRAPH

Task dependencies.

02 LIST

Store task dependencies and priorities.



03 PRIORITY QUEUE

Task scheduling

04 B-TREE

Database management system

Algorithms- Topological sort for dependencies of graph , use of ML to determine the weights for priortization

CODE

We implemented a task management system in C++.It includes classes for tasks and task manager as well as functionality to add, delete and execute tasks.

a) Task class:

- Attributes include name, description, deadline, completion status and priority. This class represents individual tasks.
- Calculates priority based on deadline and completion factors with appropriate weights to each factor.
- Allows setting and getting task attributes.
- All the methods of this class have a time complexity of O(1).

b) Task Manager class:

- Manages tasks, including adding, deleting and executing them.

CODE

- Utilizes priority queue to prioritize tasks based on their priority.
- Implements a dependency graph to handle task dependencies.
- Provides methods for adding dependencies between tasks and marking tasks as completed.
- Supports user interaction for operations such as adding, deleting and viewing tasks.
- The time complexities of the methods of this class range from O(log n) to O(n log n).

c) **Dependency handling:**

- Supports task dependencies, allowing tasks to have prerequisites before they can be executed.
- Uses directed graph data structure to represent these task dependencies, ensuring tasks are executed in the correct order.

CODE

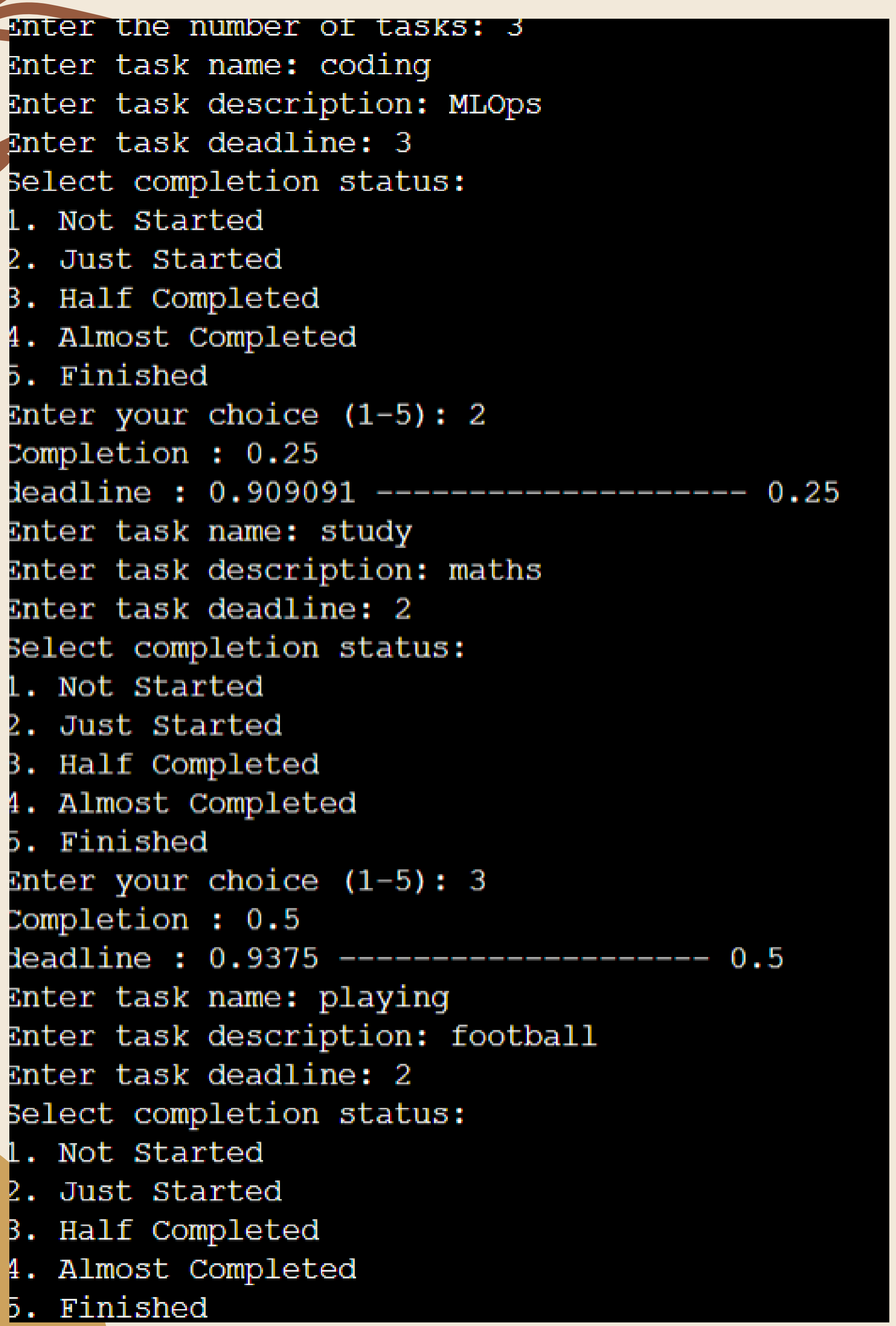
d) **User interaction:**

- Allows users to interactively create tasks, specify their attributes and add dependencies between tasks.
- Provides a command-line interface for users to perform various task management operations.

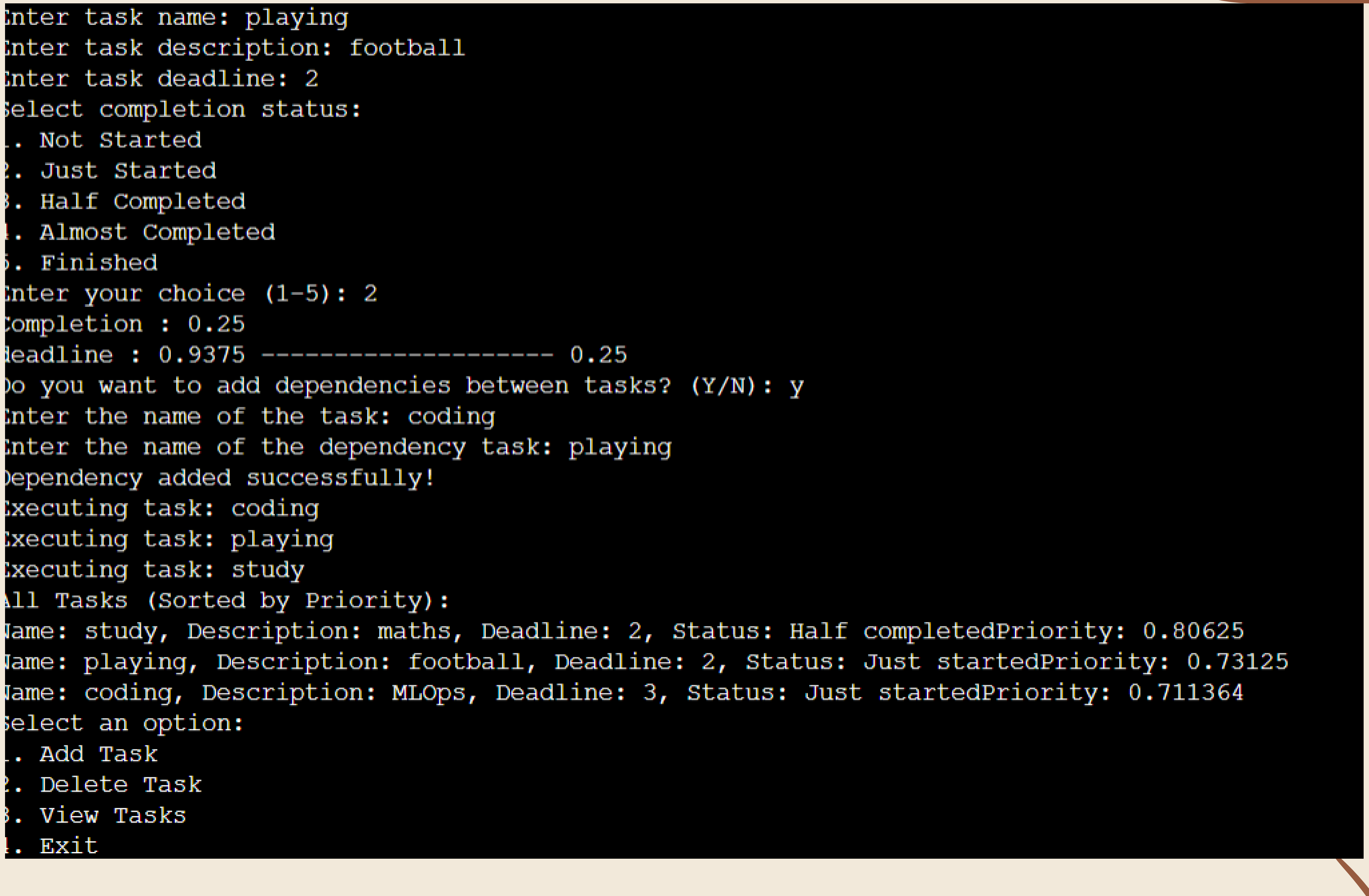
e) **Execution:**

- Executes tasks based on their priority, considering any dependencies they may have.
- Displays tasks sorted by priority, along with their details such as name, description, deadline, status and priority.

RESULTS



```
Enter the number of tasks: 3
Enter task name: coding
Enter task description: MLOps
Enter task deadline: 3
Select completion status:
1. Not Started
2. Just Started
3. Half Completed
4. Almost Completed
5. Finished
Enter your choice (1-5): 2
Completion : 0.25
deadline : 0.909091 -------------------- 0.25
Enter task name: study
Enter task description: maths
Enter task deadline: 2
Select completion status:
1. Not Started
2. Just Started
3. Half Completed
4. Almost Completed
5. Finished
Enter your choice (1-5): 3
Completion : 0.5
deadline : 0.9375 -------------------- 0.5
Enter task name: playing
Enter task description: football
Enter task deadline: 2
Select completion status:
1. Not Started
2. Just Started
3. Half Completed
4. Almost Completed
5. Finished
```



```
nter task name: playing
nter task description: football
nter task deadline: 2
elect completion status:
. Not Started
. Just Started
. Half Completed
. Almost Completed
. Finished
nter your choice (1-5): 2
ompletion : 0.25
eadline : 0.9375 -------------------- 0.25
o you want to add dependencies between tasks? (Y/N): y
nter the name of the task: coding
nter the name of the dependency task: playing
ependency added successfully!
xecuting task: coding
xecuting task: playing
xecuting task: study
ll Tasks (Sorted by Priority):
ame: study, Description: maths, Deadline: 2, Status: Half completedPriority: 0.80625
ame: playing, Description: football, Deadline: 2, Status: Just startedPriority: 0.73125
ame: coding, Description: MLOps, Deadline: 3, Status: Just startedPriority: 0.711364
elect an option:
. Add Task
. Delete Task
. View Tasks
. Exit
```

COST-BENEFIT TRADE-OFF

BENEFITS:

- Manageability of complex tasks: The code structure handles dependencies between tasks, which is beneficial to complex workflows.
- Prioritization: The priority queue ensures tasks are addressed based on urgency, potentially leading to improved efficiency.

COSTS:

- Not as efficient for large datasets: Adding, deleting and displaying a large number of tasks might become slow for some 'Task Manager' class methods.

FUTURE PROSPECTS

- Using a more efficient data structure for the dependency graph, e.g., Adjacency list for sparse graphs)
- Integrating a Database Management System which will help to manage large datasets and eliminate the need to manually enter the data.
- Making a website for the code using web assembly and emscripten.

NOTEWORTHY ACHIEVEMENTS:

- The code implementation utilizes a dependency graph to manage relationships between tasks.
- We defined an appropriate formula for task priority based on deadline and completion status by analysing various research papers.

SUMMARY

- This project is a task manager with features to prioritize tasks, manage dependencies between tasks, and interact with the user.
- It allows users to create tasks, set deadlines, mark tasks complete, and specify dependencies.
- The program prioritizes tasks based on deadline and completion status, and uses a dependency graph to ensure tasks are completed in the correct order.
- While there isn't ground-breaking innovation, the code showcases good practices and core functionalities found in modern task management applications.

REFERENCES

RESEARCH PAPERS: Submitted in the github repo

-

INDIVIDUAL CONTRIBUTIONS:

- Chinmay Vashishth: Research, Bug fixing, Code structure and Code analysis.
- Shikar Dave: Research, Bug Fixing , Coding of the project , DBMS integration try
- Shivam Khanchandani: Research, Bug fixing, User input and Presentation

GITHUB REPO LINK:

https://github.com/shikhar5647/Intelligent_task_manager

THANK YOU