Easy Gantt (Version 2.0.2)

A web-based project scheduling tool designed and developed ***by Giampaolo Marucci***.

Main Features for Final Users

The application provides a comprehensive interface for project scheduling, visualization, and customization. The primary features available to users include:

- **Task Data Input and Management:**
    - **CSV/Tab Input:** Users can input task data directly into a text area using comma or tab-separated values. The data includes fields such as **Id, Name, Duration, Percent Complete, Predecessors**, and optional parent-child relationships.
    - **File Operations:** The tool allows users to **save and open projects as JSON files**, which preserve both task data and all UI settings. Additionally, users can **export the Gantt chart as an SVG** (Scalable Vector Graphics) file or save the task list as a **CSV**.
- **Advanced Scheduling & Calendar:**
    - **Project Dates:** Users can set a **Project Start Date** and a **Today Date** to track progress.
    - **Calendar Customization:** A "Calendar" popup allows users to define **non-working days**, specifically designating Saturdays and Sundays as non-working or adding custom **non-working date ranges**.
- **Visualization and Customization (Tabbed Interface):**
    - **Header and Chart Settings:** Users can customize date formats (e.g., DD MMM YYYY), font styles for headers, chart width, margins, and the appearance of the "Today" line.
    - **Bar and Task Styling:** Extensive options exist for changing the **height, spacing, colors, and border thickness** of task bars. Users can also toggle **detailed task information** (Actuals, Durations, Predecessors) to appear next to the bars.
    - **Filtering:** The application supports filtering the view to show only specific types of tasks, such as **Parents, Delays, Critical Tasks, Milestones**, or specifically selected task IDs.

The Critical Path Method (CPM) Algorithm

The core of the application is its implementation of the **Critical Path Method (CPM)**. This algorithm determines the project timeline and identifies which tasks are critical to the completion date.

1. Data Parsing and Topological Sorting

The process begins by converting the raw CSV input into task objects. Because tasks depend on one another, the algorithm performs a **topological sort** using **Kahn's algorithm**. This ensures that tasks are processed in an order where no task is handled before its predecessors.

2. Forward Pass (Earliest Dates)

The forward Pass of the algorithm calculates the **Earliest Start Date (ESD)** and **Earliest Finish Date (EFD)** for every task.

- It respects four dependency types: **Finish-to-Start (FS), Start-to-Start (SS), Finish-to-Finish (FF), and Start-to-Finish (SF)**.
- It incorporates **Lead/Lag** times and automatically skips **non-working days** defined in the calendar.

3. Backward Pass (Latest Dates)

The Backward Pass algorithm works in reverse, starting from the project end date to calculate the **Latest Start Date (LSD)** and **Latest Finish Date (LFD)**. This determines how much a task can be delayed without affecting the overall project deadline.

4. Float Calculation and Critical Path Identification

- **Total Float:** The algorithm calculates the "float" for each task by finding the difference between its Latest Start and Earliest Start dates.
- **Critical Tasks:** A task is marked as **critical** if its **Total Float is zero** and it resides on the longest path through the project. These tasks are visually highlighted (often in red) to show they cannot be delayed.

5. Parent Task Adaptation

Summary (Parent) tasks are automatically adjusted based on their children. The algorithm aggregates the **earliest start** and **latest finish** dates of all descendants to determine the parent's duration and weighted **percent complete**. The code uses a loop to iterate these passes multiple times to ensure parent and child dates are perfectly synchronized.

6. Progress and Delay Tracking

The algorithm calculates an **Ideal Percent Complete** based on the "Today Date" and the task's elapsed duration. If a task's actual percent complete is lower than this ideal value, it is flagged as a **delay**.