Interactive Activity Selection Algorithm Visualizer in C++: Report

Submitted by:
Mehedi Hasan Emon
ID:232-115-018
mehedihasanemon23mu@gmail.com

Submitted to:
Nasif Istiak Remon
Senior Lecturer
Metropolitan University

Date: April 11, 2025

- **Introduction:**

This report presents an interactive visualizer for the Activity Selection Problem, implemented in C++. The program employs a greedy approach to solve the problem and provides a step-by-step terminal-based visualization of the selection process. The visualizer is designed to help learners understand how greedy strategies work and how choices are made at each step of the algorithm.

- **Problem Description:**

The Activity Selection Problem involves selecting the maximum number of non-overlapping activities from a given set. Each activity has a start time and an end time. The goal is to choose the maximum number of compatible activities that do not overlap in time.

For example, given three activities with intervals (1, 3), (2, 4), and (3, 5), the maximum set of non-overlapping activities is (1, 3) and (3, 5).

This problem has practical applications in task scheduling, CPU job scheduling, and meeting room allocations.

- **3 Solution Approach:**

The program implements the greedy algorithm using the following steps:

Sorting: The list of activities is first sorted by their end times in ascending order.

Selection: Initialize the end time of the last selected activity to 0. Iterate over the sorted list.
Select an activity if its start time is greater than or equal to the end time of the last selected activity.
Visualization:
During iteration, the program clears the screen and displays the current activity being considered.
It prints the table of activities with a [Checking] mark for the current activity and highlights whether it was selected or not.
Code Structure and Design:

Input Handling: The user is prompted to enter the number of activities. Each activity's start and end time is entered manually by the user.

Sorting and Selection: The Activity struct stores start, end, and index.Activities are sorted using std::sort() with a custom comparator that sorts by end time.A boolean vector is used to keep track of selected activities.

*Visualization:*The screen is cleared using system("cls") for Windows.The current activity under consideration is printed with a delay (Sleep(1000)), allowing the user to observe the selection process.The activity table is updated after each step with clear indicators of the selected ones.

*Selection Logic:*If an activity's start time is greater than or equal to the end time of the last s elected activity, it is marked as selected, and the lastEnd variable is updated.

- **Time Complexity:**

Sorting: O(n log n), where n is the number of activities.
Selection: O(n), for iterating through the sorted list.
Overall: O(n log n)

The space complexity is O(n) due to the storage of activity information and selection status.

- **Additional Considerations:**

Platform Compatibility: The code is designed for Windows, using system("cls") and Sleep(). Compatibility for Unix systems can be added by replacing these functions with system("clear") and usleep().

Educational Value: The interactive display of activity selection decisions helps students visualize how greedy choices are made and how the algorithm avoids overlaps.

Potential Enhancements: Add support for colored terminal output to distinguish selected and non-selected activities.Include input validation and automatic generation of random activities for testing. Extend the visualizer to support other greedy problems like job sequencing or Huffman coding.

- **Conclusion:**

This project delivers a simple yet effective visualization of the Activity Selection Problem using C++. It not only demonstrates the greedy strategy but also engages users with real-time feedback and step-by-step updates.