Project Specification Document

Project Title:

Cargo Management Optimization Using Greedy Knapsack Algorithm

Project Description:

This project is designed to optimize cargo selection by applying the greedy knapsack algorithm. The system loads cargo data from a CSV file, evaluates items based on a profit-to-weight ratio, and selects items that maximize value without exceeding the cargo's weight limit. This implementation demonstrates file handling, sorting algorithms, and data analysis in C.

Objectives

- Develop a C program to read cargo data from a CSV file.
- Calculate a profit-to-weight ratio for each item.
- Select the optimal combination of items based on a given cargo capacity.
- Practice efficient file handling, sorting, and mathematical evaluation in C.

Scope

The project processes up to 100 cargo items and performs the following tasks:

- Reads and parses item details from a CSV file.
- Sorts items by their profit-to-weight ratio.
- Selects items within a specified weight limit to maximize total profit.
- Displays selected items and overall metrics in a readable format.

System Overview

The application reads cargo item data from a CSV file, sorts items by their profit-to-weight ratio, and selects items based on user-defined weight limits. Key components include:

1. CSV Parsing: Reads and processes data from a CSV file.
2. Sorting: Uses `qsort` to arrange items by the calculated profit-to-weight ratio.

3. Selection: Implements a greedy algorithm to pick the most profitable items within the weight constraint.

Functional Requirements

Input Requirements:

- Reads data from a CSV file containing cargo information (ID, description, weight, and value).

- User inputs a maximum weight limit for cargo capacity.

Processing Requirements:

- Parses each line of the CSV to retrieve cargo details.

- Calculates the profit-to-weight ratio for each item.

- Sorts items by their profit-to-weight ratio in descending order.

- Selects items within the weight limit to maximize profit.

Output Requirements:

- Displays selected items with details such as ID, description, weight, and profit.

- Shows the total weight and profit of selected items.

Non-Functional Requirements

- Performance: Efficient sorting and selection for up to 100 items.

- Usability: Simple console input and output interface.

- Portability: Should compile and run in any standard C environment.

- Error Handling: Handles file access errors and malformed CSV data.

Technical Specifications

- Language: C

- Standard Library: stdio.h, stdlib.h, string.h

- File Handling: Reads data using `fopen` and parses lines with `fscanf`.

- Sorting Algorithm: Quick Sort (`qsort` function).

- Data Structures: `Item` struct to store cargo data, and an array for multiple items.

Design Details

Data Structures

Item Struct:

```
typedef struct {
    char id[20];
    char description[50];
    int weight;
    int value;
    double ratio;
} Item;
```

Functions

- load_data(): Loads cargo items from a CSV file.
- compare_items(): Custom comparison function for sorting by profit-to-weight ratio.
- greedy_knapsack(): Selects items based on the greedy algorithm and displays the results.

Assumptions

- The CSV file contains well-formed data, and each entry is unique.
- The CSV file is located in the same directory as the executable.

Limitations

- Processes a maximum of 100 items.
- Assumes all data entries are accurate; limited data validation is performed.

Testing and Validation

- Unit Testing: Test ratio calculation and sorting accuracy.
- Functional Testing: Verify file reading, sorting, and selection of items based on different cargo capacities.

- Edge Case Testing: Test with empty or partially filled CSV files, large values, and minimal weights.

Conclusion

The Cargo Management Optimization project aims to demonstrate efficient data handling and sorting through the application of the greedy knapsack algorithm. By using C programming fundamentals such as file handling, sorting, and data analysis, this project provides an optimized approach for selecting high-value cargo items within a weight limit, offering valuable insights into algorithmic design in a constrained environment.