Implementazione del Progetto di Algoritmi e Strutture Dati

Filippo Raimondi

Politecnico di Milano
Anno Accademico 2022-2023

Descrizione Generale

Il programma gestisce una lista concatenata di stazioni di servizio lungo un'autostrada. Ogni stazione ha una distanza specifica dall'inizio dell'autostrada, un numero di auto e una serie di autonomie associate a ciascun veicolo. Il programma supporta operazioni come aggiungere e rimuovere stazioni, aggiungere e rimuovere veicoli, e pianificare percorsi tra le stazioni minimizzando il numero di tappe.

Strutture Dati Utilizzate

- **Lista concatenata (STAZIONE)**: Utilizzata per gestire le stazioni di servizio lungo l'autostrada. Ogni nodo della lista rappresenta una stazione, contenente la sua distanza, il numero di auto, un array per gestire le autonomie (implementato come una tabella hash), e un puntatore al nodo successivo.
- **Tabella Hash con Indirizzamento Aperto**: Utilizzata per gestire le autonomie dei veicoli all'interno di ogni stazione. Le collisioni sono gestite con l'indirizzamento aperto, dove in caso di collisione si prova il prossimo slot disponibile fino a trovare uno spazio libero.
- **Array per stazioni (ARRAY_STAZIONE)**: Utilizzato per convertire la lista concatenata in un array, rendendo possibile una ricerca binaria e l'esecuzione di algoritmi di grafi come Dijkstra.

Funzionalità Principali

- **Aggiungi Stazione (aggiungi_stazione)**:
  - Utilizza una lista concatenata ordinata per gestire le stazioni in ordine di distanza.
  - Implementa una tabella hash per gestire le autonomie dei veicoli usando il metodo dell'indirizzamento aperto per risolvere le collisioni.
- **Demolisci Stazione (demolisci_stazione)**:
  - Rimuove un nodo dalla lista concatenata, gestendo sia la rimozione della testa della lista che di nodi interni.
- **Aggiungi Auto (aggiungi_auto)**:
  - Aggiunge un'auto alla tabella hash di una stazione esistente, gestendo le collisioni con l'indirizzamento aperto.
- **Rottama Auto (rottama_auto)**:
  - Rimuove un'auto dalla tabella hash di una stazione esistente, marcando lo slot come cancellato (CANC) e aggiornando l'autonomia massima se necessario.
- **Pianifica Percorso (pianifica_percorso)**:
  - Utilizza l'algoritmo di Dijkstra per trovare il percorso con il minor numero di tappe da una stazione di partenza a una di arrivo.
  - La distanza tra due stazioni è calcolata in termini di tappe, considerando l'autonomia massima disponibile per ogni stazione come vincolo per i possibili spostamenti.

Algoritmi Utilizzati

- **Algoritmo di Dijkstra**: Utilizzato per determinare il percorso con il minor numero di tappe tra due stazioni. L'algoritmo è implementato attraverso l'uso di un array di distanze e tappe minime, aggiornato in modo iterativo per ogni nodo (stazione) non visitato. Ogni volta che viene trovata una stazione raggiungibile con un numero minore di tappe, la distanza e il percorso precedente vengono aggiornati.
- **Ricerca Binaria (ricerca_indice_stazione)**: Utilizzata per trovare la posizione di una stazione nell'array delle stazioni. Questo migliora l'efficienza rispetto a una ricerca lineare quando si lavora con dati ordinati.