### **Pianificazione Dettagliata: Integrazione Algoritmi di Ricerca MIU con SQLite**

**Obiettivo Generale:** Implementare il processo completo di esecuzione di una ricerca MIU (derivazione di stringhe) utilizzando gli algoritmi BFS/DFS, persistendo tutti gli stati intermedi e i risultati finali nel database SQLite, garantendo la consistenza dei dati tramite transazioni.

#### Fase 1: Preparazione e Inizializzazione della Ricerca

Questa fase si occupa di impostare il contesto per una singola esecuzione di ricerca.

1. **Selezione Stringhe Iniziale e Target:**
   * **Azione:** Implementare una funzione che selezioni **due stringhe MIU casuali e distinte** dalla tabella MIU\_States.
   * **Dettaglio:** Queste due stringhe saranno rispettivamente la InitialString e la TargetString per la ricerca che verrà eseguita. Assicurarsi che le stringhe siano effettivamente diverse per avere una ricerca significativa.
2. **Registrazione della Nuova Ricerca in MIU\_Searches:**
   * **Azione:** Inserire una nuova riga nella tabella MIU\_Searches.
   * **Dettaglio:**
     + InitialString: La prima stringa selezionata al punto 1.
     + TargetString: La seconda stringa selezionata al punto 1.
     + Success: Inizialmente impostato a NULL o FALSE (verrà aggiornato alla fine).
     + FlightTime: Inizialmente impostato a 0 o NULL (verrà aggiornato alla fine).
   * **Output:** Recuperare l'ID autogenerato di questa nuova riga. Questo ID sarà cruciale per collegare tutti gli stati intermedi.
3. **Inizializzazione dello Stato Iniziale in MIU\_States:**
   * **Azione:** Inserire la InitialString come primo stato della ricerca nella tabella MIU\_States.
   * **Dettaglio:**
     + SearchID: L'ID ottenuto dal punto 2.
     + CurrentString: La InitialString della ricerca.
     + RuleSequence: Stringa vuota o NULL (non ci sono regole applicate per arrivare allo stato iniziale).
     + ParentStateID: NULL (è il punto di partenza).
     + AppliedRuleID: NULL (nessuna regola applicata).
     + Steps: 0 (zero passi dalla stringa iniziale).
     + IsTarget: TRUE se InitialString è uguale a TargetString, altrimenti FALSE.
     + CreationTIC: Timestamp corrente.

#### Fase 2: Modifica e Struttura degli Algoritmi di Ricerca (BFS / DFS)

Questa fase si concentra sull'adattamento degli algoritmi esistenti per generare e comunicare i dati degli stati.

1. **Modifica delle Firme delle Funzioni di Ricerca:**
   * **Azione:** Aggiornare le firme delle funzioni BFS e DFS (scegliendone una per l'implementazione iniziale, ad esempio BFS).
   * **Dettaglio:** La funzione dovrà accettare i seguenti parametri:
     + initialString: La stringa di partenza.
     + targetString: La stringa obiettivo.
     + searchId: L'ID della ricerca corrente (ottenuto dalla Fase 1, punto 2).
     + maxDepth: Il parametro ProfonditaMassimaRicerca dalla configurazione.
     + onStateGeneratedCallback: Una funzione di callback (o un'azione delegata) che verrà invocata ogni volta che un nuovo stato viene generato durante l'esplorazione.
2. **Generazione dei Dati dello Stato all'interno degli Algoritmi:**
   * **Azione:** All'interno della logica di BFS/DFS, ogni volta che una nuova stringa MIU viene derivata applicando una regola a uno stato genitore, raccogliere i seguenti dati:
     + CurrentString: La nuova stringa generata.
     + RuleSequence: La sequenza di regole applicate per arrivare a CurrentString (ottenuta estendendo la RuleSequence del ParentState).
     + ParentStateID: L'ID dello stato genitore in MIU\_States da cui questa CurrentString è stata derivata. Questo richiederà che gli stati in memoria (coda/stack) contengano anche il loro ID nel DB.
     + AppliedRuleID: L'ID della regola specifica che è stata applicata per generare CurrentString.
     + Steps: Steps del genitore + 1.
     + IsTarget: TRUE se CurrentString è uguale a targetString, altrimenti FALSE.
     + CreationTIC: Timestamp corrente.
3. **Invocazione della Callback:**
   * **Azione:** Subito dopo aver generato i dati di un nuovo stato (punto 2), invocare la onStateGeneratedCallback passando tutti i dati raccolti come parametri.
   * **Dettaglio:** La callback non dovrebbe bloccare l'algoritmo di ricerca. La sua responsabilità è solo di ricevere i dati e avviare la persistenza.

#### Fase 3: Implementazione della Callback per la Persistenza e Gestione Transazionale

Questa fase è critica per la persistenza dei dati e la garanzia di consistenza.

1. **Creazione della Funzione di Callback (onStateGeneratedCallback):**
   * **Azione:** Implementare la funzione che verrà passata agli algoritmi di ricerca.
   * **Dettaglio:** Questa funzione riceverà i dati di un singolo stato generato (come specificato in Fase 2, punto 2).
   * **Responsabilità:** Inserire una nuova riga nella tabella MIU\_States con i dati forniti, assicurandosi che l'operazione avvenga **all'interno della transazione database attiva**.
2. **Gestione della Transazione Database:**
   * **Azione:** Implementare un blocco di codice che gestisca l'intera transazione.
   * **Dettaglio:**
     + **Inizio Transazione:** Subito dopo aver registrato la ricerca in MIU\_Searches (Fase 1, punto 2) e prima di inizializzare lo stato iniziale in MIU\_States (Fase 1, punto 3), **aprire una transazione database**.
     + **Operazioni all'interno della Transazione:** Tutte le operazioni di inserimento in MIU\_States (sia lo stato iniziale che quelli generati dalla callback) e l'aggiornamento finale di MIU\_Searches (Fase 4) devono avvenire all'interno di questa transazione.
     + **Commit:** Se tutte le operazioni hanno successo e la ricerca termina regolarmente, eseguire il COMMIT della transazione.
     + **Rollback:** Implementare un blocco try-catch (o equivalente) attorno all'intera operazione di ricerca. In caso di qualsiasi eccezione o errore, eseguire il ROLLBACK della transazione per annullare tutte le modifiche e mantenere l'integrità del database.

#### Fase 4: Esecuzione Completa della Ricerca e Aggiornamento Finale

Questa fase orchestra l'intero processo e finalizza la registrazione della ricerca.

1. **Misurazione del Tempo di Volo (FlightTime):**
   * **Azione:** Registrare il timestamp di inizio prima di avviare l'algoritmo di ricerca.
   * **Azione:** Registrare il timestamp di fine dopo che l'algoritmo ha completato la sua esecuzione.
   * **Calcolo:** Calcolare la differenza per ottenere il FlightTime.
2. **Avvio dell'Algoritmo di Ricerca:**
   * **Azione:** Chiamare la funzione dell'algoritmo di ricerca scelto (es. BFS) passando i parametri richiesti (initialString, targetString, searchId, maxDepth, onStateGeneratedCallback).
   * **Output dell'Algoritmo:** L'algoritmo dovrà restituire un booleano (o un risultato più complesso) che indichi se la TargetString è stata trovata.
3. **Aggiornamento Finale di MIU\_Searches:**
   * **Azione:** Aggiornare la riga della ricerca in MIU\_Searches (identificata dal SearchID).
   * **Dettaglio:**
     + Success: Impostare a TRUE o FALSE in base all'esito restituito dall'algoritmo di ricerca.
     + FlightTime: Impostare con il tempo calcolato al punto 1.
   * **Importante:** Questa operazione deve avvenire **all'interno della stessa transazione** avviata in Fase 3, punto 2.

#### Considerazioni Aggiuntive e Dettagli Implementativi:

* **Regole MIU:** Assicurarsi che le regole MIU siano accessibili e applicabili dagli algoritmi di ricerca. Presumo che tu abbia già una tabella RegoleMIU e una logica per applicarle.
* **Gestione ParentStateID:** Quando un nuovo stato viene generato, l'algoritmo deve sapere quale ID (dal database) ha il suo stato genitore. Questo significa che la struttura dati utilizzata dall'algoritmo per tenere traccia degli stati (coda per BFS, stack per DFS) dovrà contenere non solo la stringa, ma anche il suo ID nel database e la RuleSequence fino a quel punto.
* **Ottimizzazione Inserimenti:** Per un numero molto elevato di stati, potresti considerare l'uso di inserimenti batch (se il tuo driver SQLite lo supporta efficientemente) all'interno della transazione, ma un singolo INSERT per ogni stato generato è un buon punto di partenza.
* **Gestione della Profondità:** L'algoritmo deve rispettare il parametro maxDepth per evitare proliferazioni eccessive di stringhe e ricerche infinite.

Questa pianificazione dovrebbe fornirti una roadmap chiara e dettagliata per i prossimi passi. Ogni punto è pensato per essere un'azione concreta. Se durante l'implementazione sorgono dubbi su un passaggio specifico, non esitare a chiedere!