**Phase 1: Research & Problem Definition**

**Goal:** Clearly define the scope and objective.

* Study what a **least common subsequence** is (vs LCS).
* Understand why this is useful in **DNA structure** (e.g., mutation detection, differences between two genomes).
* Prepare a **short literature review**—mention existing algorithms used in bioinformatics (BLAST, alignment tools, etc.).
* Decide on how you want to define “least common subsequence”:
  + Not in both strings?
  + Shortest substring in A not in B?
  + Subsequence vs substring?

**🔹 Phase 2: Algorithm Design & Selection**

**Goal:** Pick the 5 approaches and define how each will work.

Create a small section for each approach:

| **Approach** | **What it does** | **Where it fits** |
| --- | --- | --- |
| Dynamic Programming | Baseline for LCS + modified for least common | Accurate, scalable |
| Greedy | Approximate LCSq by choosing rare patterns | Fast, for huge inputs |
| Divide & Conquer | Space-optimized LCS (Hirschberg) + LCSq regions | Works for large DNA |
| Backtracking + Memo | All possible unique subsequences | Exhaustive, insightful |
| Trie / Hash | Fast pattern lookup for substrings | Speed + memory tradeoff |

* Design pseudo-code or flowcharts for each.
* Think about their **time-space tradeoffs** and what dataset sizes they can handle.

**🔹 Phase 3: DNA Data Handling**

**Goal:** Create tools/utilities for input & preprocessing.

* Decide whether you’ll use **manual input** or **FASTA format files**.
* Implement a helper module to:
  + Validate DNA sequences (A, C, G, T)
  + Clean, format, and maybe even generate random DNA for testing
* You can include a way to **load real DNA sequences** later for bonus points.

**🔹 Phase 4: Implementation (Modular)**

**Goal:** Keep code clean, separate logic per algorithm.

* Implement each algorithm in its own module/class/file.
* Have a **main controller program** that:
  + Accepts user input (sequences)
  + Lets user choose the algorithm
  + Displays output + time taken
* Add timing/benchmarking to compare performance.

**🔹 Phase 5: Testing & Evaluation**

**Goal:** Compare accuracy, performance, limitations.

* Test each approach on:
  + Small sequences (5-15 bases)
  + Medium sequences (100-200 bases)
  + Large sequences (1000+ if feasible)
* Record:
  + Time taken
  + Memory usage
  + Completeness (does it find the right LCSq?)
* Put results into **tables or graphs** for your report.

**🔹 Phase 6: Report & Presentation**

**Goal:** Document everything for submission.

Your report can follow this format:

1. **Title Page**
2. **Abstract**
3. **Introduction**
   * Problem background
   * DNA relevance
4. **Objective**
5. **Methodology**
   * Summary of 5 algorithms
   * Flowcharts/logic diagrams
6. **Implementation Details**
   * How input/output works
   * Any special data structures used
7. **Results & Evaluation**
   * Comparison table
   * Graphs (time vs sequence length)
8. **Conclusion**
   * Which approach worked best
   * Limitations and future scope
9. **References**

**✅ Final Tips:**

* Keep your code well-commented for easier debugging and report writing.
* Use simple DNA test cases to debug (like "ACGT" and "AGT").
* You don't have to finish all 5 methods 100% — even partial but well-explained ones are solid.