**Comprehensive DSA Learning Path for .NET Developers (C# Focus)**

This path is designed to take you from basic to advanced Data Structures and Algorithms, with practical C# examples and .NET-specific implementations.

**Phase 1: Foundations (2-3 Weeks)**

**1. Basic Data Structures**

* **Arrays & Strings**
  + **Declaration, traversal, common operations**
  + **Example: Reverse an array, find duplicates**
* **Lists (List<T>)**
  + **Dynamic arrays, resizing, common methods (Add, Remove, Find)**
* **Stacks (Stack<T>) & Queues (Queue<T>)**
  + **LIFO vs FIFO, common use cases (e.g., undo operations, BFS/DFS)**
* **HashSets & Dictionaries (HashSet<T>, Dictionary<TKey, TValue>)**
  + **O(1) lookups, handling collisions, practical uses (counting frequencies)**

**2. Basic Algorithms**

* **Time & Space Complexity (Big-O)**
  + **Understanding O(1), O(n), O(n²), O(log n)**
* **Linear Search vs Binary Search**
* **Basic Sorting Algorithms**
  + **Bubble Sort, Selection Sort, Insertion Sort**
  + **.Sort() in C# (uses QuickSort/IntroSort**

**Phase 2: Intermediate (3-4 Weeks)**

**3. Advanced Data Structures**

* **Linked Lists (Singly, Doubly)**
  + **Implementing from scratch in C#**
* **Trees & Binary Trees**
  + **Terminology (root, leaf, height, depth)**
  + **Traversals (Inorder, Preorder, Postorder)**
* **Binary Search Trees (BST)**
  + **Insertion, deletion, searching (O(log n) avg.)**
  + **Balanced BSTs (AVL, Red-Black Trees - used in SortedDictionary<T>)**
* **Heaps & Priority Queues (PriorityQueue<T>)**
  + **Min-Heap vs Max-Heap**
  + **Used in Dijkstra’s, scheduling algorithms**

**4. Intermediate Algorithms**

* **Divide & Conquer**
  + **Merge Sort, QuickSort**
* **Recursion & Backtracking**
  + **Factorial, Fibonacci, Towers of Hanoi**
* **Sliding Window Technique**
  + **E.g., Maximum subarray, longest substring without repeating chars**
* **Two-Pointer Technique**
  + **E.g., Pair sum, palindrome check**

**Phase 3: Advanced (4-6 Weeks)**

**5. Graph Algorithms**

* **Graph Representations**
  + **Adjacency List (using Dictionary + List)**
* **Traversal Algorithms**
  + **BFS (Queue-based)**
  + **DFS (Stack/Recursion-based)**
* **Shortest Path Algorithms**
  + **Dijkstra’s (Greedy)**
  + **Bellman-Ford (Dynamic Programming)**
* **Cycle Detection (Union-Find)**
  + **Used in Kruskal’s MST algorithm**

**6. Dynamic Programming (DP)**

* **Memoization vs Tabulation**
* **Classic Problems**
  + **Fibonacci, Knapsack, Longest Common Subsequence (LCS)**

**7. Greedy Algorithms**

* **Coin Change Problem**
* **Interval Scheduling**
* **Huffman Coding (used in compression)**

**Phase 4: Mastery (Ongoing)**

**8. Advanced Topics**

* **Tries (Prefix Trees)**
  + **Used in autocomplete (e.g., RadixTree in .NET 7+)**
* **Segment Trees**
  + **Range queries (min/max/sum in subarrays)**
* **Advanced Graph Algos**
  + **Topological Sort (used in build systems)**
  + **Floyd-Warshall (all-pairs shortest path)**

**9. System Design & Optimization**

* **When to use which data structure?**
  + **List vs LinkedList vs HashSet**
  + **Dictionary vs SortedDictionary vs HashSet**
* **Real-world .NET optimizations**
  + **Span<T> for memory efficiency**
  + **ArrayPool<T> for reducing GC pressure**

**Recommended Practice Platforms**

| **Platform** | **Focus** | **C# Support?** |
| --- | --- | --- |
| **LeetCode** | **General DSA** | **✅ Yes** |
| **HackerRank** | **Coding challenges** | **✅ Yes** |
| **Codeforces** | **Competitive programming** | **✅ Yes** |
| **AlgoExpert** | **Structured problems** | **✅ Yes** |

**Next Steps**

1. **Start with Phase 1 (Arrays, Lists, Big-O).**
2. **Implement each structure from scratch (e.g., LinkedList, BinaryTree).**
3. **Solve 1-2 problems daily on LeetCode (Easy → Medium → Hard).**
4. **Revisit .NET collections (List, Dictionary) to see real-world implementations.**

**Would you like:**

* **A curated list of LeetCode problems for each phase?**
* **Detailed C# implementations of a specific data structure?**
* **A study schedule (weekly breakdown)?**