**Advantages and Disadvantages**

**Vector**

* **Advantages**:
  + Easy to implement and straightforward for iterating through data.
  + Supports dynamic resizing.
  + Efficient for small datasets or sequential processing.
* **Disadvantages**:
  + Searching and validation require linear scans (O(n)).
  + Inefficient for large datasets or frequent lookups.

**Hash Table**

* **Advantages**:
  + Fast lookups, insertions, and deletions (O(1)) (on average).
  + Excellent choice for validating prerequisites quickly.
  + Well-suited for unordered data storage.
* **Disadvantages**:
  + Consumes more memory due to hashing overhead.
  + Hash collisions can degrade performance to O(n) in worst cases.
  + No inherent order of data.

**Binary Search Tree**

* **Advantages**:
  + Maintains data in sorted order, enabling efficient in-order traversal.
  + Logarithmic time complexity for insertion, deletion, and search (O(logn)) in balanced trees.
  + Combines the benefits of ordering and efficient lookups.
* **Disadvantages**:
  + Insertion and search can degrade to O(n) if the tree is unbalanced.
  + More complex implementation compared to vectors or hash tables.
  + Additional memory overhead for storing pointers.

**Runtime Analysis Table with Total Cost**

**Vector**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Open and read file** | 1 | n | n |
| **Parse line and create course object** | 1 | n | n |
| **Validate prerequisites** | 1 | {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>n</mi><mn>2</mn></msup></mstyle></math>","origin":"MathType for Microsoft Add-in"} | {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>n</mi><mn>2</mn></msup></mstyle></math>","origin":"MathType for Microsoft Add-in"} |
| **Store course in vector** | 1 | n | n |
| **Total Cost** | | | 3n+{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>n</mi><mn>2</mn></msup></mstyle></math>","origin":"MathType for Microsoft Add-in"} |
| **Runtime** | | | O({"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>n</mi><mn>2</mn></msup></mstyle></math>","origin":"MathType for Microsoft Add-in"}) |

**Memory**: Low memory usage. Scales linearly with the number of courses (O(n)).

**Hashmap**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Open and read file** | 1 | n | n |
| **Parse line and create course object** | 1 | n | n |
| **Validate prerequisites** | 1 | n | n |
| **Insert course into hash table** | 1 | n | n |
| **Total Cost** | | | 4n |
| **Runtime** | | | O(n) |

**Memory**: Higher memory usage due to hashing overhead and bucket storage.

**Binary Search Tree**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Open and read file** | 1 | n | n |
| **Parse line and create course object** | 1 | n | n |
| **Validate prerequisites** | {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>log</mi><mo>&#xA0;</mo><mi>n</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"} | n | n{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>log</mi><mo>&#xA0;</mo><mi>n</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"} |
| **Store course in binary search tree** | {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>log</mi><mo>&#xA0;</mo><mi>n</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"} | n | n{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>log</mi><mo>&#xA0;</mo><mi>n</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"} |
| **Total Cost** | | | 2n+2n{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>log</mi><mo>&#xA0;</mo><mi>n</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"} |
| **Runtime** | | | O(n{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>log</mi><mo>&#xA0;</mo><mi>n</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"}) |

**Memory**: Moderate memory usage due to pointer storage for tree nodes.

**Recommendation**

Based on the runtime analysis and evaluation of advantages and disadvantages:

* **Recommendation**: Use a **Hash Table** for storing and processing course data.
* **Justification**:

The hash table offers the fastest average-case performance (O(1)) for lookups, insertions, and deletions. This aligns well with the program's primary requirements, such as validating prerequisites quickly and retrieving course details efficiently. While its worst-case runtime is O(n), such scenarios are rare with a well-designed hash function. Additionally, its memory overhead is acceptable given its performance benefits.  
To address its lack of ordering, a secondary structure like a vector or binary search tree can handle tasks requiring sorted data. For scalability and efficiency, the hash table is the most practical choice.