Vector Data Structure:

Runtime Complexity:

- Reading the file and creating course objects:
    - Opening the file: O(1) (1 time)
    - Looping through each line: O(n)
    - Splitting each line into tokens: O(n)
    - Creating Course object: O(n)
    - Adding Course object to vector: O(1) (amortized)
    - Total: O(n)

Memory Usage:

- The vector will store each course object, consuming memory proportional to the number of courses (O(n)).

Advantages:

- Allows for direct access to elements by index.
- Simple to implement and use.

Disadvantages:

- Insertion and deletion operations can be inefficient if resizing is required.
- Not optimal for searching (linear search).

Hash Table Data Structure:

Runtime Complexity:

- Reading the file and creating course objects:
    - Opening the file: O(1) (1 time)
    - Looping through each line: O(n)
    - Splitting each line into tokens: O(n)
    - Creating Course object: O(1)
    - Inserting course into hash table: O(1) (average case)
    - Total: O(n)

Memory Usage:

- The hash table will consume memory proportional to the number of courses (O(n)), plus additional overhead for storing the hash table itself.

Advantages:

- Offers constant-time average case for insertion, deletion, and search operations.
- Suitable for large datasets.
- Efficient for searching.

Disadvantages:

- May have collisionsleading to performance degradation.
- Not ordered.

Tree Data Structure:

Runtime Complexity:

- Reading the file and creating course objects:
    - Opening the file: O(1) (1 time)
    - Looping through each line: O(n)
    - Splitting each line into tokens: O(n)
    - Creating Course objec O(1)
    - Inserting course into tree: O(log n) (assuming balanced tree)
    - Total: O(n log n)

Memory Usage:

- The tree will consume memory proportional to the number of courses (O(n)), plus additional overhead for storing the tree structure.

Advantages:

- Provides efficient search, insertion, and deletion operations.
- Maintains order.
- Suitable for scenarios where data needs to be sorted.

Disadvantages:

- May require balancing to maintain optimal performance.
- May have higher memory overhead compared to hash tables.

Recommendation:

Based on the analysis of runtime complexity and memory usage, as well as considering the advantages and disadvantages of each data structure, the **hash table** would be the most suitable choice for this scenario.