1. a. When the dictionary list is kept in a Linked List

As each possibility of a word in a grid has to be checked, for each position in the grid, different combination of characters is formed from the position in the grid by appending the next character in a particular direction until we reach the boundary of the grid (or length of the longest word in the dictionary). This is performed in all 8 directions. Each of these combination of characters is then checked for their presence in the linked list as all the dictionary words are stored in the linked list one after another.

1. b. When the dictionary list is kept in an AVL tree

As each possibility of a word in a grid has to be checked, for each position in the grid, different combination of characters is formed from the position in the grid by appending the next character in a particular direction until we reach the boundary of the grid (or length of the longest word in the dictionary). This is performed in all 8 directions. Each of these combination of characters is then checked for their presence in the AVL tree as all the dictionary words are stored in the AVL tree in a balanced manner.

1. c. When the dictionary list is kept in a Hash Table

As each possibility of a word in a grid has to be checked, for each position in the grid, different combination of characters is formed from the position in the grid by appending the next character in a particular direction until we reach the boundary of the grid (or length of the longest word in the dictionary). This is performed in all 8 directions. Each of these combination of characters is then checked for their presence in the hash table as all the dictionary words are stored in the hash table and the collision resolution is done using separate chaining method. To check for their presence, each combination of characters obtained are sent to the hash function. Based on the position returned, the list at that position is checked for its presence.

2. a. Linked list

In the program, there are three for loops. One each for row, column and direction. These are nested and hence result in O(N^3). Within these, we use a contains method for linked list which takes O(N) time. Thus the entire operation results in O(N^4).

2. b. AVL tree

In the program, there are three for loops. One each for row, column and direction. These are nested and hence result in O(N^3). Within these, we use a contains method for AVL tree which takes O(log N) time. Thus the entire operation results in O(N^3 log N).

2. c. Hash table with separate chaining

In the program, there are three for loops. One each for row, column and direction. These are nested and hence result in O(N^3). Within these, we use a get method for hash table which takes O(1) time. Thus the entire operation results in O(N^3).

Result for a random grid of 8 rows and 9 columns

Elapsed time for linked list: 5324

Elapsed time for tree structure is: 60

Elapsed time for hash table is: 10

Result for a random grid of 10 rows and 10 columns

Elapsed time for linked list: 9872

Elapsed time for tree structure is: 83

Elapsed time for hash table is: 7