2.

In the implementation of insert (or find) method, comparison between keys are required. However, for Coord which is not a built-in type, a required overloading comparison operator (e.g., the != operator) is not provided in the code.

3b.

In this recursive function, a parameter (as pointer) is needed to access subdomains of the current domain, while one more parameter is also needed for storing current path. In this way, complete paths could be obtained.

4a. O(N3).

There are 3 layers of for-loops. Each loop runs for N times. Moreover, each statement inside these loops, if not an internal loop, has O(1) time complexity. In sum, the overall time complexity should be O(NxNxN) = O(N3).

4b. O(N3).

For large N, we could fix the iteration times of the loops that do not run a fixed number of iterations with the maximum possible iteration times. In this approximate way, the second-layer for loop (with j) runs for N-1 times at most, which make this problem reduces to 4a with time complexity of O(N3).

5. O(N2).

The for-loop statement runs N times since there are N elements in smaller. For each iteration, the 3-parameter get method has O(N/2) time complexity since it traverses the doubly-linked list from one end to reach the destination node, which is of O(N/2) in average. Then the 2-parameter get method and one of the insert and erase methods that would be called (decided by the if-condition) have same O(N) time complexity since the common dominating work is to call find method (which traverses the linked list). Because these statements are executed in sequence, the iteration takes O(N) and the whole for-loop takes O(N2). Additionally, all other statements before/after the for-loop (e.g., definition of res that calls copy constructor who traverses the linked list, swap res and result exchanging head pointers and sizes of the two lists) take O(N) at most. So the overall time complexity should be O(N2) + O(N) = O(N2).