CS2303 Project The Snake Game

Report

Lai Chun Wai (56211836)

Time complexity analysis:

(Snake Body Part)

The snake body part is a linked list implemetation which contain a linked list represent the Snake Body Segment. First, what I did is to update the linked list with a new node after the snake head eat a food. I created a new SnakeBodySegment object using the (prev\_head\_cell) and insert it to the tail of the linked list which is the head of the snake body segment. For insertion ,I made the current tail pointer point to the new node and make the new node the new tail. This is actually a insertion of linked list which may have time complexity of O(n) because I just insert it at the tail of the list for N nodes.

(Rendering Part)

The rendering part is actually a traversal of linked list, what I did is created a for loop and update the coordinate of every node in the body segment , printing every node in the loop .

Therefore the time complexity should be O(n) because there is only 1 loop to traverse the whole linked list with N nodes.

(Undo & Resume Part)

The Undo and Resume Part is actually storing the snake’s states in a stack using push function firstly ,which has time complextity of O(n) for pushing N states into the stack.Then whenever the snake is dead and backspace is pressed ,the top element of the stack will be assigned back to the snake and the top element will be poped out of the stack in order to restore the last state of the snake to revive using the pop function.The time complextity should also be O(n) of poping n elements.Then when player pressed enter, the snake.alive will be changed to true again to revive and continue the game which has O(1) time complexity.

(Collision Detection)

The collsion Dectection used a quadtree to find out that whether there is roadblocks in a block region. Firstly what I did is defining the case of no collision and collision occurs. When the root== null or root->count = 0, there is no collision, when the region is divided that right-left and celing-floor =1, there is collison. This two case has both O(1) time complexity.

After that , in order to dectect collision, I have to called the DetectCollision function recursively to check all the subtree of the quadtree, means every block in the game with dividing the map in to upper left, lower left, upper right, lower right. The time complexity of traverse and check the whole quadtree from its root should be O(logn).

Optional Part:

For the Auto Navigation of the snake game. Firstly , what we want is to find a path connect the snake and the food with no road blocks and snake body node in between, I think that can be implemented using a stack data structure. It is because it is similar to the solution of a maze. What we can do is to implement a stack , push every possible neighbour nodes from the snake head and breaks the walls, whenever we find a roadblock or bodysegments when visiting the next neighbour, we go back and pop out the top pushed element and try other neighbours again , eventually we can find a clear path to the food with every coordinates of the internal blocks, then we can insert them in to the snake body segment linked list. The snake can automatically find the foog in this ways, I think we will need to for loop to implement the above method, the first one is a loop for the neighbour visiting operations, we need to loop it until we reached the food, the second loop is for looping for every new generated food blocks in the game. Suppose we have N food nodes generated in the game and n neighbour blocks for visiting between the snake head and the food blocks. The time complexity of the operation should be O(n^2).