CS 61B
Spring 2019

Extra Practice Worksheet
Discussion 5: ADTS

Worksheet 5 Solutions

1 ADT Basics

Consider the following scenarios and choose the appropriate ADT to tackle the problem.

At her EA internship, Jessica is responsible in building logging functionality for the company's latest puzzle game. In the game, characters can move up, down, left, or right at any given timestep and can die at any time.

a. Suppose Jessica wants to be able to keep track of all the character movements starting at time step 0. What ADT can she use to accomplish this?

Solution: Jessica should use a List. Because we are never concerned with removing elements, we don't require a stack or queue's pop properties.

b. Tess, Jessica's coworker, would like the game log to also keep track of whether or not a given character has made all possible movements (up, down, left, right). What ADT can be used to implement this?

Solution: Jessica should keep a Set of the 4 different moves. Whenever she encounters a character movement, remove that movement from the set if possible. Whenever q query is made to see if all possible movements have been done, check to see if the set is empty.

c. Jessica would also like to keep track of what movements lead to the characters death and record the timestamp of these movements. Assume that there is atleast one movement that never leads to the character's death. What ADT can be used to accomplish this?

Solution: Use a map to keep track of pairs of movements and timestamps.

2 This Question is Also About ADTS

Suppose we wanted to adjust the Stack class to also be able to support a min function which finds the minimum value in the stack. Briefly describe how we can implement this operation to take O(1) time.

Solution: We can approach this problem by using an additional stack to keep track of minimum values as each element is pushed onto the stack. Whenever a new element is added to our main stack, we compare it with the top value of the second stack and if our new element is smaller, we also push it on top of our second stack. Every time we pop an element off the main stack, we check to see if the element we had to pop off is equal to the top value of our second stack, in which case if it is, we pop it off the second stack as well. Whenever we have to access the minimum of the stack, we simply peek into our second stack.