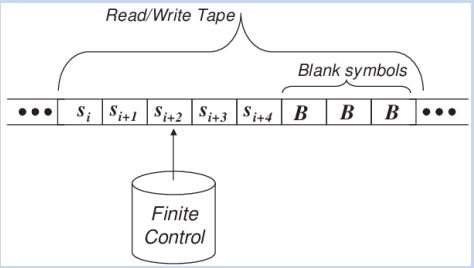
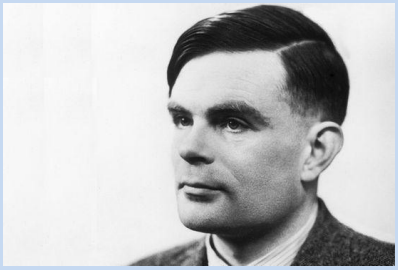
Chapter 3 Part 3 Supplement: Reason for Doubly Linked List

By now, you know the difference between arrays and linked lists and their relative advantages and disadvantages. You learned about two types of linked list: single-linked list and double-linked list. Here we will discuss why double linked lists are important.

Although a single-linked list allows us to handle a linear sequence of elements that can be of unknown length and dynamic[1] in our program, there are some limitations in the traversal of the list. Suppose we want to do read/remove/insert near at the end of the list. Then we have to traverse all the way from the head to our position of interest then do the desired operation. When the list is long then this traversal cost is a huge problem. Furthermore, we have to be very careful regarding where to stop the traversal. For example, if you want to insert at position n, then we have to stop traversal at position n – 1. If you want to read then we stop traversal at n. If we make one more wrong step then we cannot backtrack. You have to restart from the head.

This second problem of not being able to move back-and-forth during our traversal is a major concern. In later courses, you will learn that moving backward on a list can be considered as going back in time to investigate the context of future events. This is highly useful for complex language parsing, intelligent agent design, and statistical modeling.





Some of you know about the famous mathematician and computer scientist Alan Turing. He investigated and proved many important properties about computers. One of his most fascinating contributions is the Turing Machine that can simulate any computer. By doing so, the Turing machine allows us to investigate the fundamental limitations and capabilities of all computers ever being built. The machine is surprisingly simple. It has an infinite tape of symbols and a read-write head that can make one-step forward or backward on that tape in each step or change the content of the tape cell where the head is located at that moment. The read-write head, also called the finite control, keeps track of its current state and decides what to do based-on the symbol on the tape in its position and its state.

So what kind of list you need to implement this tape?

[1] means the sequence may change during program execution