Array vs. Linked List

Khalid Hourani

February 14, 2017

An array is a collection of elements identified by their index. An array is stored in memory so that the index of each element of the array can be calculated by a simple mathematical formula. For example, an array of 5 32-bit integers beginning at memory address 5000 would have its i^{th} element's stored at memory region

$$5000 + 4i$$

An example of such an array would be [1,2,3,4,5].

On the other hand, a linked list is a collection of data structures containing an *object* and a *pointer*. Each element in a linked list is called a *node*, and each pointer refers to another node in the linked list. The first node in a linked list is the *head*, and the last node is the *tail*. The tail contains a null-pointer in order to show that it is the last node of a linked list.

A simple example of a linked list is one which represents a non-negative integer. Let the head represent the ones-digit of the number and the tail represent the leading digit.

```
typedef struct NumberNode
{
  int digit;
  struct NumberNode* next;
} NumberNode;
```

The number "12345", then, could be represented by the linked list:

```
Five.digit = 5;
Five.next = &Four;

Four.digit = 4;
Four.next = &Three;

Three.digit = 3;
Three.next = &Two;

Two.digit = 2;
Two.next = &One;

One.digit = 1;
One.next = NULL;
```

While an array needs its length declared and its elements must be stored contiguously in memory, a linked list can be arbitrarily long and its elements can be stored in virtually any memory region. Moreover, removing an element from or adding an element to an array is much harder than from a linked list, as, with an array, one is required to reallocate memory and adjust the array size. With a linked list, an element can be added or removed by changing pointers in the list.

However, a linked list necessarily requires sequentially traversing through its elements. An array, on the other hand, allows for random access. In the array example above, if one wants to find the 3rd element of the array, one simply needs to find the 32-bit integer stored at memory region

$$5000 + 4 * 3 = 5012$$

For the example above, if one wants to find the fourth digit from the right, one has to traverse through each node:

$$\text{Five} \Rightarrow \text{Four} \Rightarrow \text{Three} \Rightarrow \text{Two}$$

Finally, a linked list can be difficult to traverse backwards. While this can be solved with a doubly linked list (essentially including a previous and next pointer, instead of only a next pointer), this requires the use of additional memory.