Why in a heap implemented by array the index 0 is left unused?

I'm learning data structures and every source tells me not to use index 0 of the array while implementing heap, without giving any explanation why. I searched the web, searched StackExchange, and couldn't find an answer.

algorithm heap

edited Apr 6 '14 at 21:59

asked Apr 6 '14 at 21:41
Xiang Ji
382 ● 2 ● 13

1 I've never heard of not using index 0 in a heap. It slightly changes the arithmetic for calculating indices (left/right child, parent), but it's pretty insignificant. I've implemented heaps several times and never avoided using 0. – Emmet Apr 6 '14 at 22:20

3 Answers

There's no reason why a heap implemented in an array has to leave the item at index 0 unused. If you put the root at 0, then the item at `array[ix]` has its children at `array[ix*2+1]` and `array[ix*2+2]`. The node at `array[child]` has its parent at `array[(child-1)/2]`.

Let's see.

```
                root at 0    root at 1
Left child       ix*2 + 1     ix*2
Right child      ix*2 + 2     ix*2 + 1
Parent           (ix-1)/2     ix/2
```

So having the root at 0 rather than at 1 costs you an extra add to find the left child, and an extra subtraction to find the parent.

I can't see those few extra instructions making much of a difference in the run time.

edited Apr 8 '14 at 4:20

answered Apr 6 '14 at 22:25

Jim Mischel
75.3k ● 5 ● 60 ● 151

As observed by AnonJ, this is a question of taste rather than technical necessity. One nice thing about starting at 1 rather than 0 is that there's a bijection between binary strings x and the positive integers that maps a binary string x to the positive integer written 1x in binary. The string x gives the path from the root to the indexed node, where 0 means "take the left child", and 1 means "take the right child".

Another consideration is that the otherwise unused "zeroth" location can hold a sentinel with value minus infinity that, on architectures without branch prediction, may mean a non-negligible improvement in running time due to having only one test in the sift up loop rather than two.

edited Apr 19 '14 at 21:52

answered Apr 6 '14 at 22:23

David Eisenstat
19k ● 5 ● 11 ● 37

(While I was searching, I came up with an answer of my own but I don't know whether it's correct or not.)

If index `0` is used for the root node then subsequent calculations on its children cannot proceed, because we have `indexOfLeftChild = indexOfParent * 2` and `indexOfRightChild = indexOfParent * 2 + 1`. However `0 * 2 = 0` and `0 * 2 + 1 = 1`, which cannot represent the parent-children relationship we want. Therefore we have to start at `1` so that the tree, represented by array, complies with the mathematical properties we desire.

edited Apr 6 '14 at 22:03

Bolo
6,206 ● 2 ● 20 ● 48

answered Apr 6 '14 at 21:41

Xiang Ji
382 ● 2 ● 13

2 We don't **have to** start at 1, since nothing is forcing us to use those equations as is, but starting at 0 will add a few `-1`s and `+1`s to the equations. – Dukeling Apr 6 '14 at 21:44

1 @Dukeling OK, so the heap, as defined mathematically(conceptually), should have a root with an index "1" (the whole structure starts at 1). We might choose to implement this root with array[0], but if so we have to do some `+1`, `-1`, which will be a little annoying. So normally we start at array[1]. Am I right in this interpretation? – Xiang Ji Apr 6 '14 at 21:54

Yes, that sounds correct. – Dukeling Apr 6 '14 at 21:56