Why Merge Operation in Merge Sort is O(n)?

Microsoft Azure
0100
11011
Scott Guthrie reveals new technology to build business apps, fast.
AdChoices
Save the date
Microsoft

For merge-sort divide and conquer operations, how much time is required in bottom up merging phase? My instructor says that it is be linear, hence it will be `O(n)`. But I didn't get it. How will it be linear?

How will merging operation be linear `O(n)`?

mergesort

edited Aug 14 '12 at 7:40
Nishant
25.4k ● 4 ● 51 ● 74

asked Aug 14 '12 at 7:03
Pankaj_C
1 ● 3

The arrays are sorted. Take a look at algolist.net/Algorithms/Merge/Sorted_arrays – irrelephant Aug 14 '12 at 7:05

1 Answer

Merge operation of two arrays, is scanning arrays and picking the lowest/highest of two.

so you have

```
a: [1, 3, 6, 7]
b: [4, 5, 7, 8]
```

you compare like this (sort of pseudo code)

```
indexA = 0;
indexB = 0;
auxilaryArray = [];
indexAux = 0;

while true
   if indexA > len(a)-1 or indexb > len(b) -1
       break
   # you are cherry picking one from the two array which is lesser
   # until any one of these array exausts
   if(a[indexA] > b[indexB])
       auxilaryArray[indexAux++] = b[indexB++]
   else
       auxilaryArray[indexAux++] = a[indexA++]

#append the remaining array
while indexA < len(a)
    auxilaryArray[indexAux++] = a[indexA++]

#appends the remaining array
while indexB < len(b)
    auxilaryArray[indexAux++] = b[indexB++]
```

you see if array `a[k]`, and `b[m]` the *sum of iterations by the three loops* will be `k+m`.

In case

```
a: [1, 3, 6, 7]
b: [4, 5, 7, 8]
```

Here is the first loop for this:

```
(0, 0), (1, 0), (2, 0), (2, 1), (2, 2), (3, 2) # 6 iterations; array a exhausted
```

The second loop does not run since `a` is already scanned.

The third loop appends

```
b[2], b[3]   # 2 iterations; b exhaused
```

You see `8 = 4 + 4` loops running? What's the order `O(n)`.

In Mergesort the divide operation is logarithmic, `ln n` -- the merge part is linear. Since you divide and merge back the order becomes multiplicative so Mergesort is `O(nln(n))`.

Unlike Bubble, Selection, Insertion sort where you scan left to right (O(n)) and then fit the right candidate by consecutive swaps (bubble), or by scanning the minimum in rest of the the unsorted array (selection), or by inserting at right place in sorted part of the array (insertion) -- these operations are O(n)... so the overall order of these algos becomes O(n^2)

edited Aug 14 '12 at 7:34

answered Aug 14 '12 at 7:20
Nishant
25.4k ● 4 ● 51 ● 74

it's really useful...... thanks. – Pankaj_C Aug 14 '12 at 8:56

@Pankaj_C If this answer solved your problem, please mark it as answered by clicking on the up-tick next to this answer. See here how. If you liked the answer, you may also up-vote by clicking on up arrow next to the answer. – Nishant Aug 14 '12 at 17:28