21. True or false? The memory overhead associated with the Binary Tree Sort is less than that of the memory overhead associated with the Bubble Sort.

This is generally false. The Bubble sort has a very low memory overhead.

22. The integers 65, 80, 70, 18, 86, 6, and 37 are stored sequentially in an array with 65 stored in element 0 and 37 stored in element 6. Draw the binary tree represented by the array.

See Below. The first tree shown in number 26 is this one.

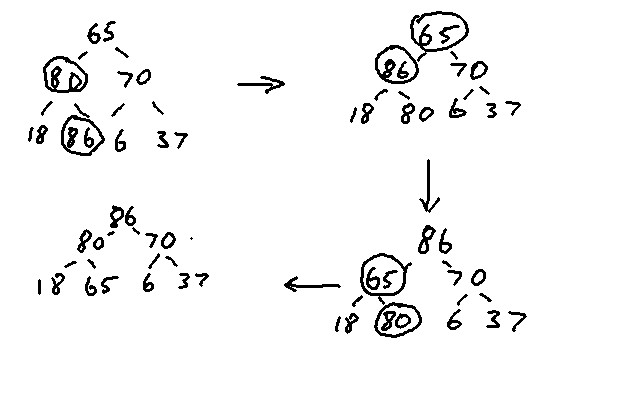
23. Define the term “heap.” In a heap any given "parent" node is >= both of the "child" nodes.

24. Is the tree discussed in **Exercise 22** a heap? If not, identify the contents of the array that is preventing it from being a heap. No. The requirement is that each subtree of each non-leaf node has to be a heap. This is true for NONE of the parent nodes.

25. How does the Reheap Downward algorithm get its name? It rebuilds the heap starting at the "top" (= root).

26. The integers 65, 80, 70, 18, 86, 6, and 37 are stored sequentially in an array with 65 stored in element 0, and 37 stored in element 6. The Reheap Downward algorithm is used to arrange the integers into a heap. Trace the execution of the algorithm by drawing a sequence of trees similar to the ones depicted in Figure 8.11. In your figures, shade the elements being compared.

The actual question is about the "Build Heap" part of the algorithm.



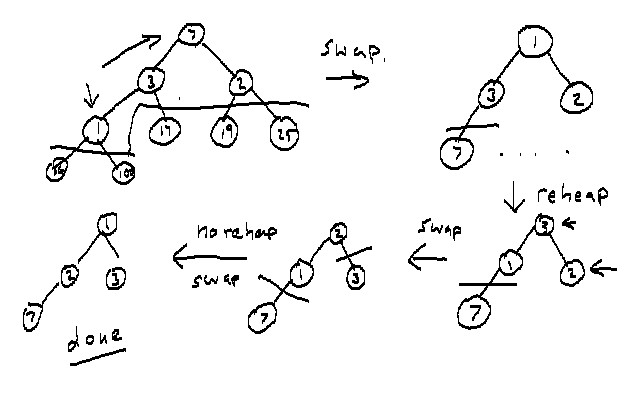
27. Show the changes to the array discussed in the previous exercise in a table similar to the one shown in Figure 8.12.

|  |  |  |  |
| --- | --- | --- | --- |
| 65 | 65 | 86 | 86 |
| 80 | 86 | 65 | 80 |
| 70 | 70 | 70 | 70 |
| 18 | 18 | 18 | 18 |
| 86 | 80 | 80 | 65 |
| 6 | 6 | 6 | 6 |
| 37 | 37 | 37 | 37 |

28. Draw the tree representation of the array shown in column t of Figure 8.14.

See Below. The first tree from 29 is this one.

29. Show the changes made to the tree drawn in the previous exercise in order to sort the remaining four integers (7, 3, 2, and 1) using the Heap Sort.



30. Show the changes to the array of integers shown in column s of Figure 8.14 that reflect the changes to the trees drawn in the previous exercise. Produce a table similar to the one shown in Figure 8.14.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 7 | 1 | 3 | 2 | 1 |
| 3 | 3 | 3 | 1 | 1 | 2 |
| 7 | 2 | 2 | 2 | 3 | 3 |
| 1 | 1 | 7 | 7 | 7 | 7 |
| 17 | 17 | 17 | 17 | 17 | 17 |
| 19 | 19 | 19 | 19 | 19 | 19 |
| 25 | 25 | 25 | 25 | 25 | 25 |
| 36 | 36 | 36 | 36 | 36 | 36 |
| 100 | 100 | 100 | 100 | 100 | 100 |

31. A 16 element array stores the integers that are to be sorted using the merge sort.

a. Using the formulas presented in Table 8.6, how many comparisons would be made to sort the 16 items.

SE = (0.75)n log2(n) = (0.75)\*16\*4 = 48. It takes an average of 48 compares to sort 16 items.

d. Using the formulas presented in Table 8.6, how many swaps would be made to sort the16 items.

Nswaps = 2.67\*48 = 128.

32. What characteristic of a data set makes the Quicksort slow?

When the dataset is already sorted the speed goes like n2. The reason is, there are no swaps but every element gets compared at each level of the recursion.

33. A 16 element array stores the integers is to be sorted using the Quicksort.

a. Using the formulas presented in Table 8.6, how many comparisons would be made to sort the 16 items.

SE = 1.45(16)log2(16) = (1.45)(64) =92.8 (No fractional compares, this is an average.)

b. Using the formulas presented in Table 8.6, how many swaps would be made to sort the16 items.

Nswaps = (0.255)(92.8) = 23.7

This takes more compares but fewer swaps than the MergeSort.