1. Define the term “sorting.”

To place a data set in numerical or alphabetical order.

2. Give two reasons for sorting a data set.

a. To produce sorted output.

b. So that non-hashed search algorithms are faster.

3. Give a condition under which it would not be advantageous to store a set of nodes in sorted order.

It makes sense to not use sort if you don't need a sorted list.

4. Give three factors to be considered in selecting a sorting algorithm for a particular application.

a. The size and character of the data set.

b. The needs of the application.

c. The skill of the programmer.

5. Define the term “sort effort.”

Sort Effort is the number of comparisons required to sort a data set of n items.

6. Give an expression for the minimum number of comparisons required to sort *n* items.

The minimum for any sort algorithm is: O(n log2 (n)).

7. My friend has told me that he has discovered an algorithm that performs 5,000,000 comparisons to sort 1,000,000 items, regardless of the character of the data set. Should I believe him, and why?

For 1,000,000 items nlogn is about 1.9X107 (19 million) which is more than she is claiming.

8. Calculate the time (in minutes and seconds) required to sort 10,000,000 items on a system that performs a comparison in two nanoseconds, assuming the sorting algorithm's sort effort is (don't consider swaps):

a. *n* 0.02 sec.

b. *n* log2(*n*) 0.465 sec.

c. *n* 2  20 sec

9. To reposition two nodes into sorted order, is it more desirable to perform a shallow or deep copy of the nodes? Why?

It's actually better to perform shallow copies. They are faster, in general, than deep copies.

10. Under what conditions is the Binary Tree Sort fast?

The Binary Tree sort is the fastest when the tree is balanced. This can usually occur if the entries are randomly presented.

11. Calculate the minimum and maximum number of comparisons required to sort 1,000,000 items using the Binary Tree Sort.

a. Minimum number of comparisons 1.99X107

b. Maximum number of comparisons 1012

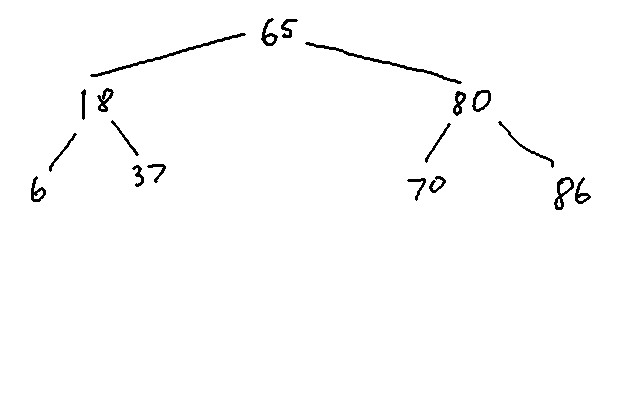
12. Give the times, (in minutes and seconds) to perform the sort described in the previous exercise on a machine that performs one comparison in one-half nanosecond (don't consider swaps).

a) 9.97X10-3 sec

b) 500 sec (8 min 20 sec) (The amount of time for light to get to Earth from the Sun)

13. The integers 65, 80, 70, 18, 86, 6, and 37 are to be sorted using the Binary Tree Sort. Assume the integers are processed by the algorithm in the order given.

a. Show the binary tree that results from sorting.

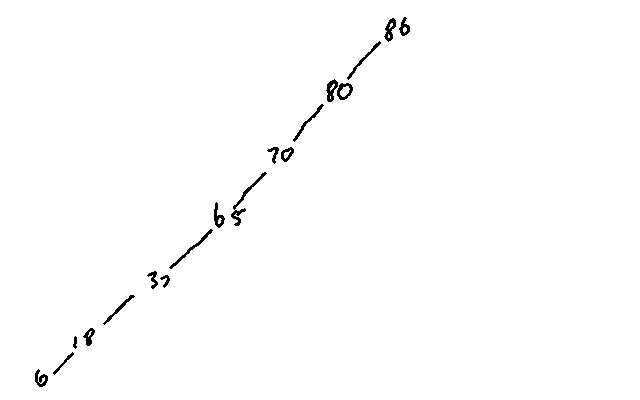


b. As the integers are placed in the tree, count the number of comparisons made. What is the total number of comparisons?

This tree had 11 total comparisons.

14. The integers 86, 80, 70, 65, 37, 18, and 6 are to be sorted using the Binary Tree Sort. Assume the integers are processed by the algorithm in the order given.

a. Show the binary tree that results from sorting.



b. As the integers are placed in the tree, count the number of comparisons made. What is the total number of comparisons?

This tree has 21 total comparisons. This is the largest possible number for a data set with seven elements.

15. Why is the variable flip included in the Bubble Sort algorithm?

Flip is a boolean variable. It is set false unless two nodes have been swapped in the current pass. It gets set true when a swap is made. If we get to the end of a pass and it's still false then there were no swaps in this pass and we are done.

16. 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 Bubble Sort is used to sort them. Trace the execution of the sort by constructing a table similar to the one presented in Figure 8.7. Shade the elements being compared.

Pass 1: Pass 2:

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

Pass 3: Pass 4:

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

There will be one more pass through the "while" loop. At the end of pass 4 the variable "flip" is true and the number of sorted elements is officially four. The remaining three are in order but the program doesn't know it. A fifth pass through the while loop will end with flip being false so that the program knows it is finished. The example in the book is similar, it requires a fifth pass for the program to know it is finished.

17. Count the number of comparisons made on each pass through the sort performed in the previous exercise and present the result as a tabulation of pass number vs. number of comparisons.

|  |  |
| --- | --- |
| Pass | Comparisons |
| 1 | 6 |
| 2 | 5 |
| 3 | 4 |
| 4 | 3 |
| 5 | 2 |

Total comparisons is 20.

18. If a Bubble Sort does not end early, how many comparisons are required to sort *n* items?

n(n-1)/2. A dataset which is in reverse sorted order will have the maximum.

19. What is the minimum number of comparisons necessary to sort *n* items using the Bubble Sort?

If they are already sorted then there are only n-1 compares.

20. Give an example of a 10-item data set that would be sorted quickly by the Bubble Sort.

1,2,3,4,5,6,7,8,9,10 (Although with only ten elements the difference in times is going to be in the nanosecond range.)