***Karatsuba Multiplication:***

Base is 2 for binary, n is # of bits/digits

***Recursive Relations:***

Tree:

Example of Recurrence:

Depth:

***Master Theorem:***

Examples:

We could have used any x > 0, in order to upper bound T(n) with since:

for any x > 0

--

--

T(n/2) + 1 = O(logn)

2T(n/2) + log n = O(n)

2T(n/2) + n = O(nlogn)

2T(n/2) + n log n = O(nlog2n)

2T(n/2) + n2 = O(n^2)

***Quicksort:***

Leftmost digit is the pivot

[23, 17, 14, 6, 13, 10, 5, **18**]

23 > **18**

Swap 5 with 23, then 23 with 18

[5, 17, 14, 6, 13, 10, **18**, *23*]

Worst-case:

Random average-case:

Split:

1. Pick right-most pivot
2. Compare all numbers to the right with pivot
3. If larger add to right, if smaller add to left

***Mergesort:***

1. Split into single numbers
2. Merge based on lowest number:
   1. Compare lowest number of pair to lowest of other pair

[13, 10], [18,5]

[10 > 5, 10 < 18, 13 < 18] : 4 compares

Worst-case:

Average-case:

***Heapsort:***

Build-max-heap:

[23, 17, 14, 6, 13, 10, 5, 18]

1. Place numbers in heap arbitrarily, no order at all.
2. Start at bottom most leaf node “i”.
3. Max-heapify, compare two leaf nodes for larger, then compare larger to parent node. If larger you swap them.

Worst-case:

***Binary Search:***

Worst-case:

Average-case:

Suppose n = 3.

\* : number

\_ : gap between numbers

[2, 1, 2] = [\*, \*, \*]

[2, 2, 2, 1, 2, 2, 2] = [\_, \*, \_, \*, \_, \*, \_]

***Logarithmic Properties:***

***Horner’s Method Of Polynomial Evaluation:***

3 multiplications, 3 additions

***Stirling’s Formula:***

***Important Summations:***

***The Selection Algorithm:***

***Big Oh Notation:***

Examples: