Test One
Ryan English
02/15/2021

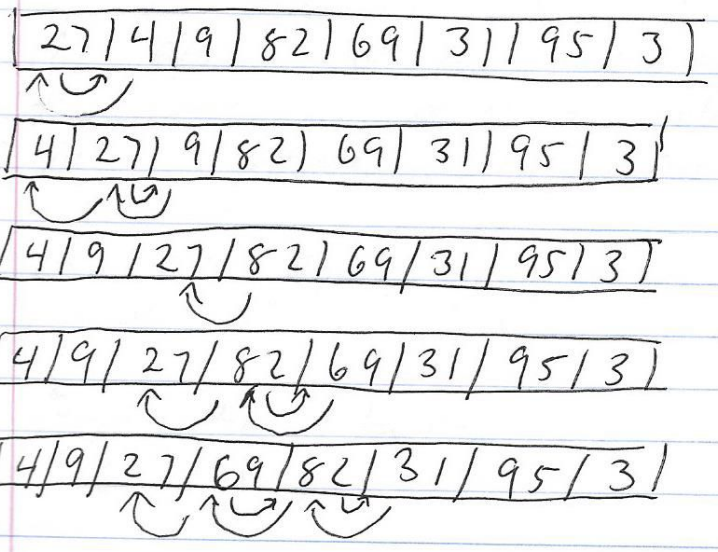
1. Describe Insertion sort

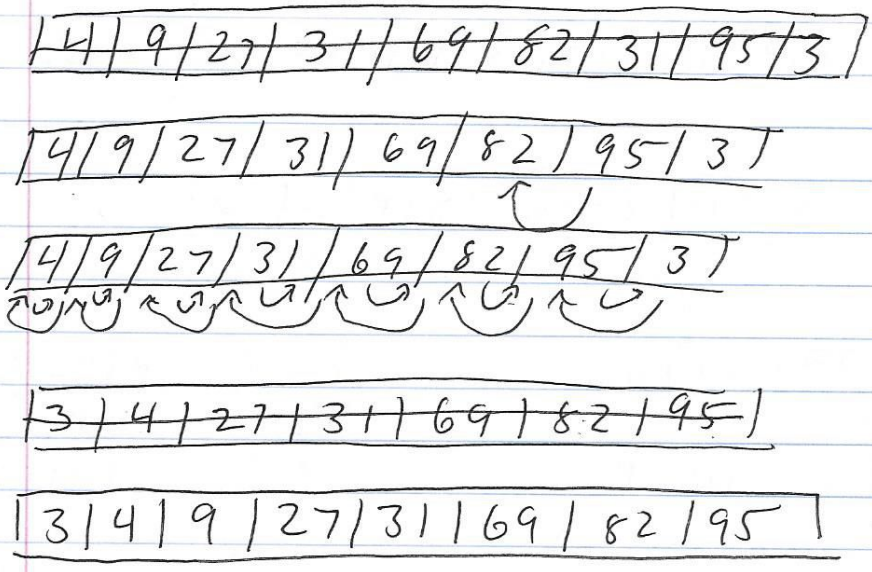
Insertion sort takes an array of elements of length n where $n > 0$ and sorts them where $a_0 < a_1 < a_2 < \ldots < a_n$ by starting at position 1 (i) and comparing backwards each element. If the element is greater than ($a_i < a_{i-j}$) swap the two elements, only stopping where the element is greater ($a_i > a_{i-j}$); thus creating a sorted array.

Recurrence Eq
$T(n) = 2T(n/2) + n$

Asymptotic solution
$\theta(n^2)$

Sort $A = (27, 4, 9, 82, 69, 31, 95, 3)$





b. Merge sort

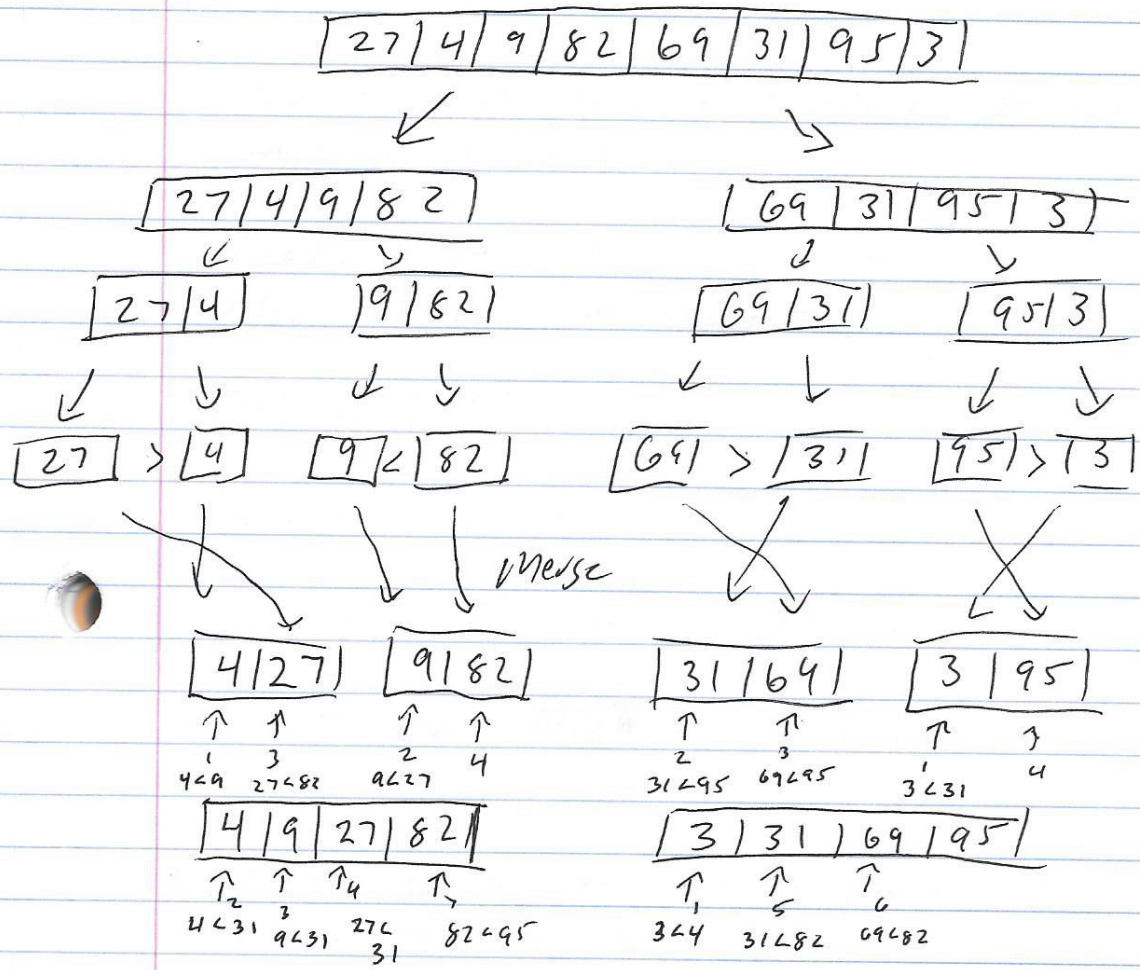
Describe Merge sort

Merge sort is a divide and conquer design algorithm where an array of size n is sorted by recursively splitting it into two sub arrays and sorting those subarrays. The recursion continues until there is only 1 element and then merges back up the two sub arrays by comparing the first element of each sub array and grabbing the lowest and moving the index up by 1. This is continued until sorted where $a_0 < a_1 < a_2 \cdots < a_n$

Recurrence Eq
$T(n) = 2T(n/2) + \Theta(n)$

Asymptotic notation
$\Theta(n \lg n)$

Sort A = (27, 4, 9, 82, 69, 31, 95, 3)



3 | 4 | 9 | 27 | 31 | 69 | 82 | 95 ← Sorted

2. Describe each by listing in/out and writing recurrence model for T(n) with exact solution

a. Linear search for v in a sequence of n numbers stored in array a

Input: A [0, ... n]
Output: If v is in array A

$T(n) = 2T(n/2) + n$

Use master theory

$a = 2 \quad b = 2 \quad f(n) = n \quad n^{\log_2 2} = n$

Case 2 → $T(n) = \Theta(n \lg n) \quad O(n^2)$

b. Binary search for v in a sequence of n numbers stored in array A

Input: A [0, 1, ..., n]
Output: If v is contained in array a

$T(n) = 2T(n/2) + \Theta(n)$

Use master theory

$a = 2 \quad b = 2 \quad f(n) = n \quad n^{\log_2 2} = n$

Case 2 → $T(n) = \Theta(n \lg n)$

C. DAC to compute multiplication of two square matrixes.

Input: $A_{n \times n}$ $B_{n \times n}$ Matrixes
Output: $C_{n \times n}$

$T(n) = 8T(n/2) + \Theta(n^2)$

Use Master Theory to solve

$a = 8 \quad b = 2 \quad f(n) = \Theta(n^2) = Cn^2$

$n^{\log_2 8} \rightarrow n^3 \qquad n^3 > Cn^2$

Case 1 $\rightarrow$ $T(n) = \Theta(n^3)$

D. Towers of Hanoi

Input: Three stacks with left stack sorted $a_1 a_2 c \ldots a_n$
Output: Three stacks with right stack sorted $a_1 a_2 \ldots a_n$
$\hookrightarrow$ other two stacks empty

$T(n) = 2T(n-1) + 1$

$\Theta(2^n - 1) \rightarrow \Theta(2^n)$

e. fast exponentiation

Input: b^n where $n > 0$
Output: b^n multiplied out

$T(n) = T(n/2) + 1$

~~$O(n^2)$~~

		Level
$T(1) =$	0	
$T(n) =$	1	0
$T(n-1) =$	1	1
	⋮	⋮
	1	$\lg n - 1$
$T(1)$	0	$\lg n$

$T(n) = 1 + 1 + 1 \ldots + 1 + 0$
$T(n) = \lg n$

3. $T(n) = 2T(n-1) + 1$ for $n > n$
$T(1) = 1$

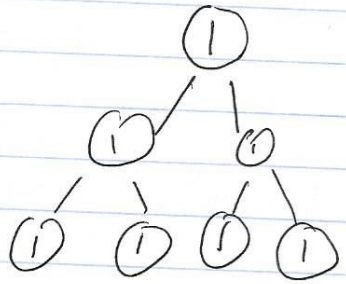
a) $T(n)$ — Level 0

$T(n-1)$ — Level 1

$T(n-2)$ — Level 2

⋮

(1) n times — Level n



b) $T(n) = 1 + 2 + \ldots + 2^{n-1}$
$= 2^n - 1$

c) $T(n) = 2T(n-1) + 1$

$= 2(2^{n-1} - 1) + 1$

$= 2(2^{n-1} - 1) + 1$

$= 4^{n-1} - 2 + 1$

$= 4^{n-1} - 1$

$= 2T(n) = 2 \cdot 2^{n-1}$

4. Compute mean m and the variance v of n numbers stored in A

$$m = \sum_{i=1}^{n} A[i] / n$$

$$v = \left[\sum_{i=1}^{n} A[i] - nm^2\right] / n-1$$

a. ~~for i = n to A.length do~~

sum := 0	1
m := 0	1
v := 0	1
for i = 1 to A.length do	n+1
sum = sum + A[i]	n
end	
m = sum / A.length	1
v = (sum − (n+m)²) / (A.length − 1)	1

b. loop invariants

sum = A[0] + A[1] + ... A[i]

c. notation

$T(n) = \theta(n)$