CS 325 - HW 2 Solution

Problem 1: Give the asymptotic bounds for T(n) in each of the following recurrences. Make your bounds as tight as possible and justify your answers. (-0.5 for O instead of Θ) **(1.5 point each)**

a) $T(n) = 2T(n-2) + 1$ **(1.5 pt)**

a = 2, b = 2, f(n) = $\Theta(n^0)$ so d = 0

$T(n) = \Theta(2^{n/2})$ or $\Theta((2^{1/2})^n) = \Theta((\sqrt{2})^n)$

Using the Muster Method

b) $T(n) = T(n-1) + n^3$ **(1.5 pt)**

a = 1, b = 1, f(n) = $\Theta(n^3)$ so d = 3

$T(n) = \Theta(n^4)$

Using the Muster Method

c) $T(n) = 2T\left(\frac{n}{6}\right) + 2n^2$ **(2 pt)**

a = 2, b = 6, $\log_6(2) = 0.38$ $n^{0.38}$,f(n) = $2n^2$

$n^{0.38}$ = O(f(n) = $2n^2$)

Case 3:

2f(n/6) = $2*2n^2/6^2 = n^2/9$ <= $2cn^2$ *which is true* for c = 1/9

So T(n) = $\Theta(2n^2)$ or $\Theta(n^2)$

CS 325 - HW 2 Solution

Problem 2: (5 points total)

a) Verbal description & Pseudocode (3pts)

Code similar to code below

```
function quatSearch (A, key, start, end)
    if (start > end)
        return false;
    q = INT(end-start)/4
    q1= start + q;
    q2 = start + 2q;
    q3 = start + 3q;

   if (A[q1] == key)
        return true;
   else if (A[q2] == key)
        return true;
   else if (A[q3] == key)
        return true;
   else if (key < A[q1])
        return quatSearch (A, key, start, q1-1);
    else if (key > A[q3])
        return quatSearch (A, key, q3+1, end);
     else if (key > A[q2])
        return quatSearch (A, key, q2+1,q3-1);
    else
        return quatSearch (A, key, q1+1,q2-1);
```

b) Recurrence: (1pt)

T(n) = T(n/4) + 3 or T(n) = T(n/4) + 3k or T(n) = T(n/4) + k or T(n) = T(n/4) + Θ(1) (1 pt)

c) Solution : (1 pt)

Using the master method. May use any method.

a=1, b=4, $\log_4 1=0$, $n^0 = 1$

compare f(n) = Θ(1) so case 2.

T(n) = Θ(lgn) (1 pt)

CS 325 - HW 2 Solution

Problem 3: Design and analyze a divide and conquer algorithm that determines the minimum and maximum value in an unsorted list (array). Write pseudo-code for the min_and_max algorithm, give the recurrence and determine the asymptotic complexity of the algorithm. Compare the running time of the recursive min_and_max algorithm to that of an iterative algorithm for finding the minimum and maximum values of an array.

a) Verbal description & Pseudocode (3pts)

Similar to code below**.**

MIN-MAX(A)

If |A|=1 then return min=max=A[0]
Divide A into two equal subsets A1 and A2

(min_1, max_1) = MIN-MAX(A_1)
(min_2, max_2) = MIN-MAX(A_2)

If $min_1 <= min_2$ then min = min_1
Else min = min_2
If $max_1 >= max_2$ then max = max_1
Else max = max_2
Return (min,max)

b) Recurrence (1 pts) $T(n) = 2T(n/2) + 2$ **or** $T(n) = 2T(n/2) + c$

c) Solution (1 pt) : $T(n)$ is $\Theta(n)$ May use any method to solve

CS 325 - HW 2 Solution

Problem 4: Consider the following algorithm for sorting. **(5 points)**

```
STOOGESORT(A[0 ... n - 1])
    if n = 2 and A[0] > A[1]
        swap A[0] and A[1]
    else if n > 2
        k = ceiling(2n/3)
        STOOGESORT(A[0 ... k - 1])
        STOOGESORT(A[n - k ... n - 1])
        STOOGESORT(A[0 ... k - 1])
```

a) **(1pt)** Explain why the STOOGESORT algorithm sorts its input. (This is not a formal proof).

The base case either has one element or two elements, which are correctly sorted. The three recursive calls overlap by >n/3 elements (by the rounding-up choice). Call these elements the overlap elements, the first n/3 elements the prefix elements and the last n/3 elements the suffix elements. After the first recursive call, the prefix elements are smaller than the overlap elements. After the second recursive call the overlap elements are smaller than the suffix elements; the suffix elements are sorted. So the prefix elements are smaller than the suffix elements. The final recursive call sorts the prefix and overlap elements.

b) **(1pt)** Would STOOGESORT still sort correctly if we replaced k = ceiling(2n/3) with k = floor(2n/3)? If yes prove if no give a counterexample. (Hint: what happens when n = 4?)

No. A counterexample should be given. For example, consider the input list [0 3 1 2]. n = 4 and ⌊2n/3⌋=2 so A[0 ... k − 1] = [0 3] which does not change in the recursive call and A[n− k...n− 1] = [1 2] does not change in the recursive call. The list does not get sorted.

c) **(1pt)**

$T(n)=3T(2n/3) + \Theta(1)$ or $T(n)=3T(2n/3) + c$ or $T(n)=3T(2n/3) + 3$

d) **(2pt)**

Master method a=3, b = 3/2, $\log_{(3/2)}3 = 2.71$

$f(n)=c = O(n^{2.71})$

$T(n) = \Theta(n^{\log 3/2\ 3})$ or $\Theta(n^{2.71})$

CS 325 - HW 2 Solution

Problem 5: *(10 points)*

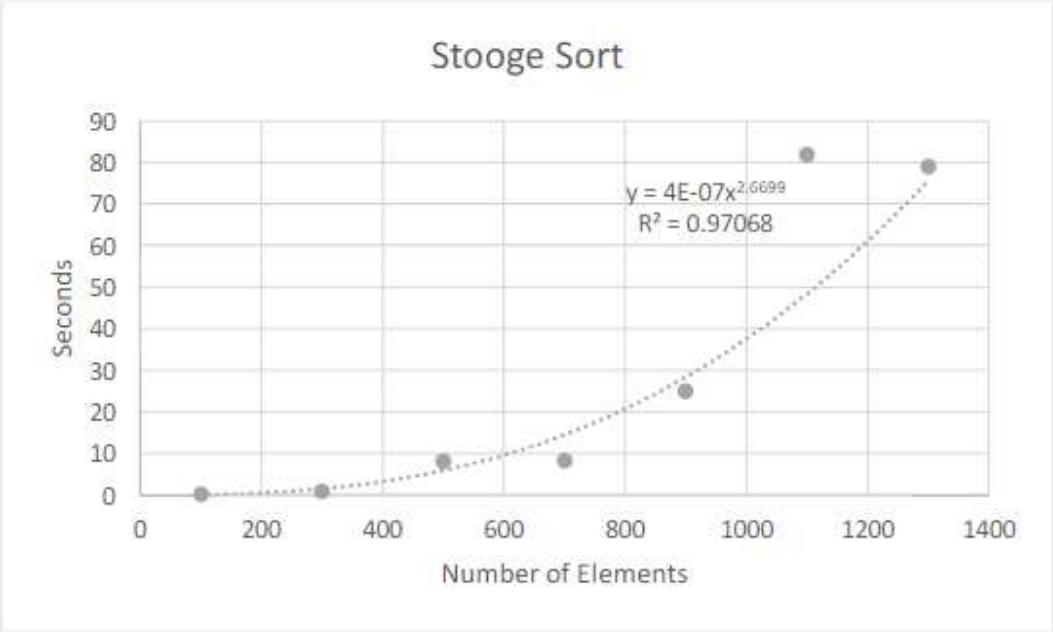
a) Implement STOOGESORT from Problem 4 to sort an array/vector of integers. Implement the algorithm in the same language you used for the sorting algorithms in HW 1. Your program should be able to read inputs from a file called "data.txt" where the first value of each line is the number of integers that need to be sorted, followed by the integers (like in HW 1). The output will be written to a file called "stooge.out".

- **README 1 pt**
- **Code compiles and executes- 3pts**
- **Test with data.txt and correct output to stooge.out – 2 pts**

Submit a copy of all your code files and a README file that explains how to compile and run your code in a ZIP file to TEACH. We will only test execution with an input file named data.txt.

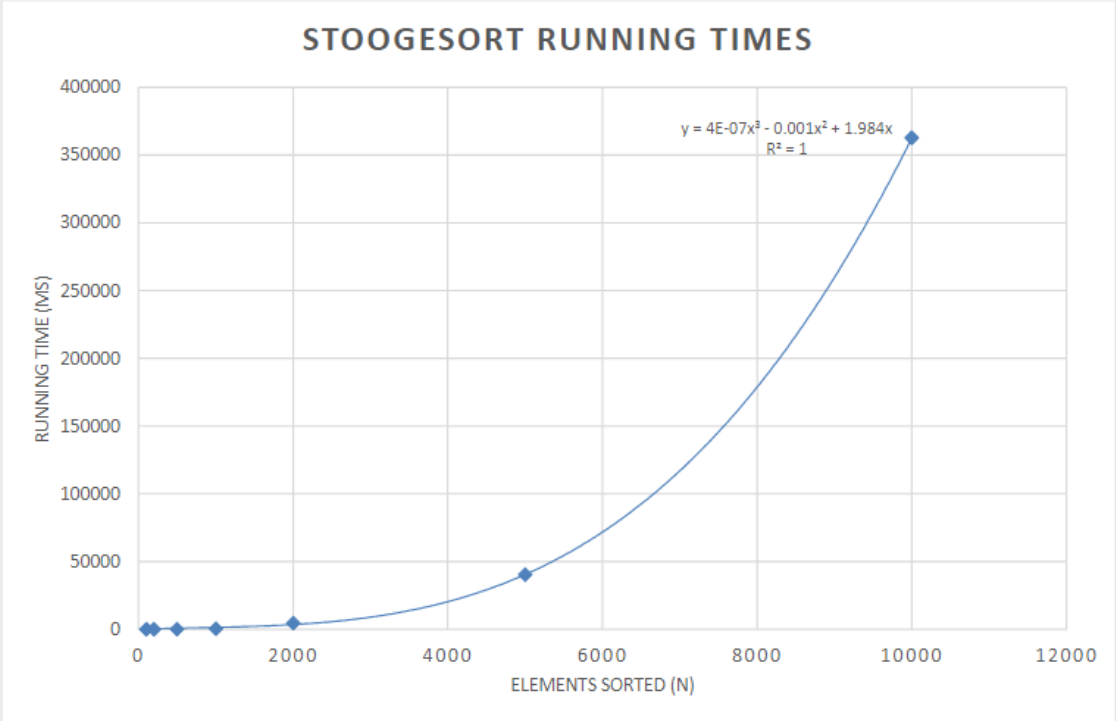
Example graphs below. All any polynomial curve with the degree 3<=d<=2 received full credit.

(-.5) for higher order curves.



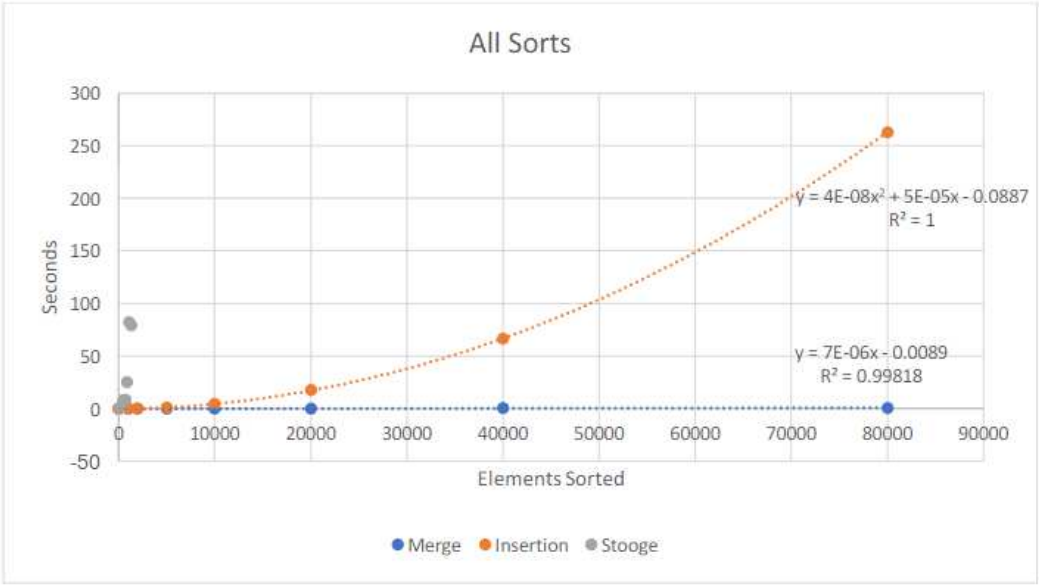
CS 325 - HW 2 Solution

5d. Curve: $y = 4E\text{-}07x^3 - 0.001x^2 + 1.984x$, $R^2 = 1$

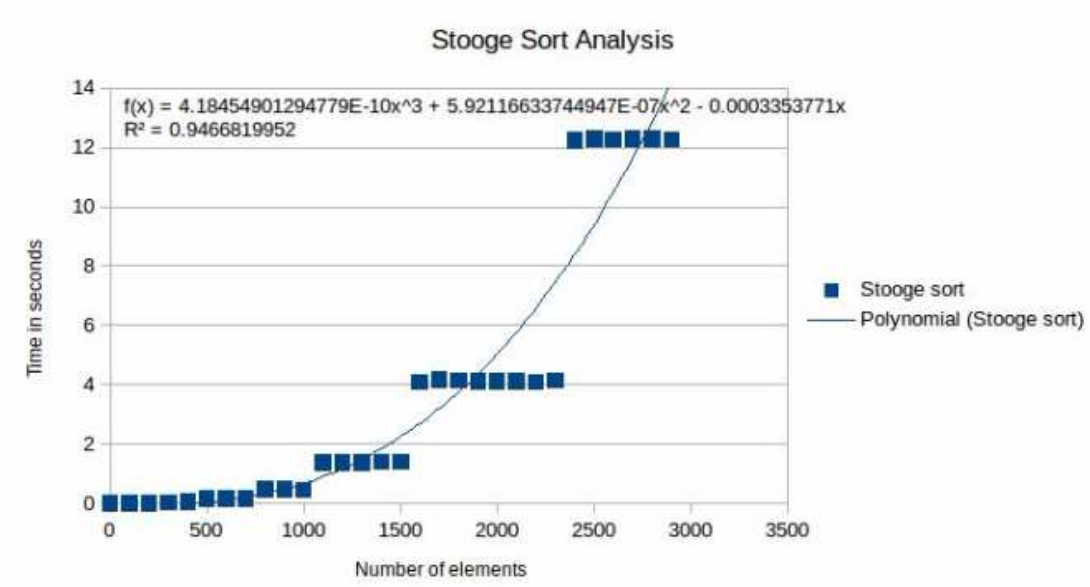


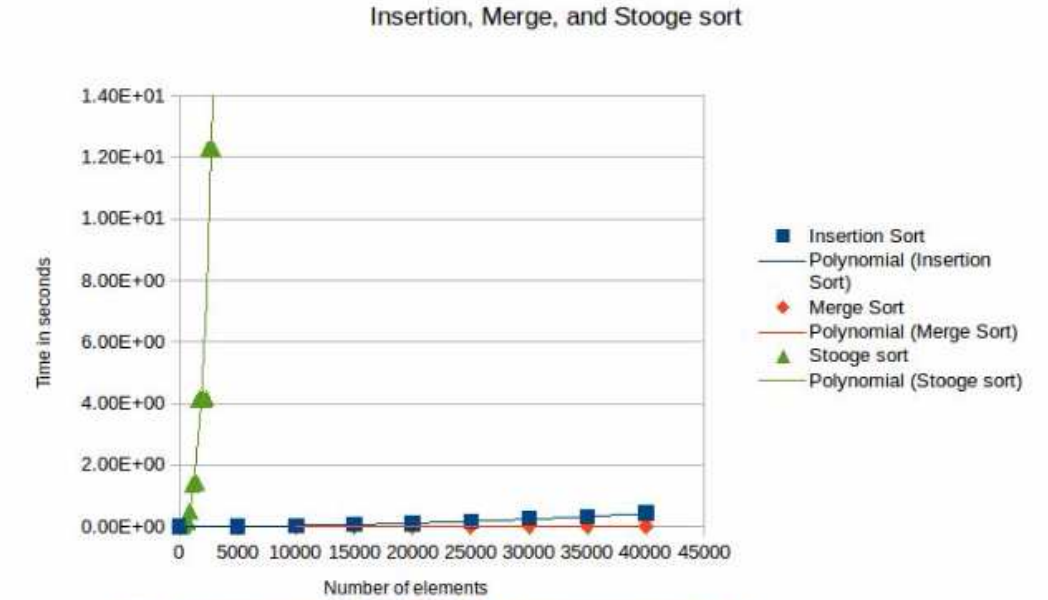
b) What type of curve best fits the StoogeSort data set? Give the equation of the curve that best "fits" the data and draw that curve on the graphs of created in part c).

c)



CS 325 - HW 2 Solution





f(x) = 2.59894516426387E-10x^2 + 2.88731919397116E-07x
R² = 0.999833273
f(x) = 3.30263558441559E-13x^2 + 1.57245235662338E-07x + 5.61518703030307E-05
R² = 0.9994686691
f(x) = 4.18454901294779E-10x^3 + 5.92116633744947E-07x^2 - 0.0003353771x
R² = 0.9466819952

CS 325 - HW 2 Solution

d) A power function curve fit it best. The equation for the curve of best fit is $9*10^{-6}x^{2.727}$.