Lachlan Sinclair

Homework 2 CS 325

7/7/2019

Problem 1)

a) $T(n)=T(n-1)+2n$: Muster method. $T(n)=aT(n-b)+f(n)$ note:$f(n)$ is (n^d). a=1, b=1, d=1. Use the case where a=1 => **$T(n)= \Theta (n^{d+1})= \Theta (n^2)$.**

b) $T(n)=T(n-2)+3$: Muster method. a=1 , b=2, d=0. Use the case where a=1 => **$T(n)= \Theta (n^{d+1})= \Theta (n)$.**

c) $T(n)=4T(n/2)+3n^2$: Master method. A=4, b=2 => $n^{log\_b(a)}=n^{log\_2(4)}=n^2$. F(n)=$3n^2$ which is $\Theta (n^2)$. Therefor we can use case 2 since f(n) has the same order as the number of nodes. **$T(n)= \Theta (n^2lg(n))$.**

d) $T(n)=2T(n/4)+n^2$: Master method. A=2, b=4. $n^{log\_b(a)}=n^{log\_4(2)}=n^{1/2}$=sqrt(n). $f(n)=n^2$. Since $f(n)=\Omega(n^{log\_4(2+1)})= \Omega(n^{.79})$ we can use case 3. And 2*f(n/4) <= c*f(n), for c=2 and all sufficiently large n. Therefor **$T(n)= \Theta (n^2)$.**

Problem 2)

a) For a quaternary search algorithm, I would write code that recursively calls its self until either the value is found, or the start and end of the array subsection being looked at are equal. I then would calculate what one fourth of the array size is. Then set variables equal to the values of the array at position ¼, ½, ¾ by using the previously calculated ¼ size. Then I would check these 3 elements to see if they equal the value being search for, if one does return true. If not found, compare the value being searched to the three points of interest and pass the subsection of the array it will reside in if the array contains it. Repeat until found or the base case is met.

Pseudo code:

```
quaternarySearch(array, start, end, value)
        if(end<start) return false;
        quarter = (end-start)/4;
        middle=2*quarter;
        upper=3*quarter;
        if(array[quarter] == value) ||(array[middle] == value) || (array[upper] == value)
                return true;

        if(value< array[quarter])

                return quaternarySearch(array, start, quarter, value)

        else if( value < array[middle])

                return quaternarySearch(array, quarter, middle, value)

        else if (value < array[upper])
```

return quaternarySearch(array, middle, upper, value)

else

return quaternarySearch(array, upper, end, value)

b) T(n)=T(n/4)+ Θ (1)

c) Master method:
a=1, b=4 => $n^{log\_b(a)}=n^{log\_4(1)}=n^0=1$. F(n)= Θ (1). Therefor case 2 can be used. T(n)= Θ (lgn).
This is exactly the same running time as the binary search program. Its incredibly similar code, it basically does one extra check of a middle value on every recursive call. log_2(1) and log_4(1) both equal zero, which results in the same runtime.

Problem 3)

a) T(n) = 3T(2n/3)+ Θ(1)

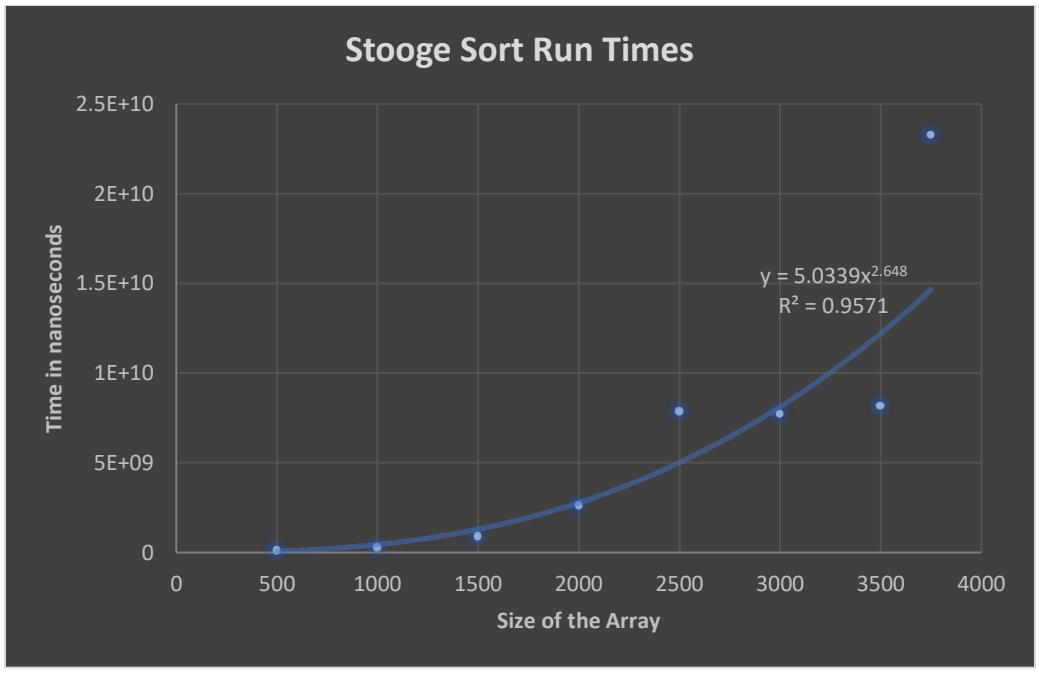
b) Master method:
a=3, b=3/2. $n^{log\_b(a)}=n^{log\_3/2(3)}=n^{2.71}$
f(n)=c, therefor case 1 applies. T(n)= Θ ($n^{2.71}$).

Problem 4)

c)

Stooge Sort						
Size of the Array	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average
500	98382108	115662125	109176783	99474580	99202045	104379528.2
1000	298465222	291262147	294073270	295687507	296168701	295131369.4
1500	853817347	858260548	869041076	870227209	892596307	868788497.4
2000	2619227930	2616035377	2683822848	2561317134	2699856429	2636051944
2500	7876488601	7873745533	7820379998	7985253871	7771006507	7865374902
3000	7639316622	7781846988	7663084560	7782114175	7776123371	7728497143
3500	7877599577	7861229885	7811317556	8869954127	8430026593	8170025548
3750	24019750407	23207090108	22975815261	23135235678	22951846388	23257947568

d) The power curve fits my data best. See the graph for the provided equation and R^2 value.



e) The actual runtime value obtained from running stooge sort on the flip server was quite close to the theoretical value. Using the power trendline my data produce an equation of the order 2.648, and in problem 3 the theoretical runtime I produced has an order of 2.71. Glancing at the data above it is clear that it does not perfectly match the trendline, after trying it on multiple platforms and using different optimization commands I'm pretty confident the plateau is because of the integer division by three that takes place.

f)