Analysis of Algorithms - Assignment 2

Colby Rush

February 10, 2014

1 Section 2.2

1.1 No. 2

1. $n(n+1)/2 \in O(n^3)$ is true
2. $n(n+1)/2 \in O(n^2)$ is true
3. $n(n+1)/2 \in \Theta(n^3)$ is false
4. $n(n+1)/2 \in \Omega(n)$ is false

1.2 No. 3

1. $(n^2+1)^{10} \in \Theta(n^{12})$ because $(n^2+1)^{10}$ will always be almost the same as n^{12}
2. $\sqrt{10n^2+7n+3} \in \Theta(n^2)$ because $\sqrt{n^2}+\sqrt{n}$ is roughly equal to n^2
3. $(2n)log(n+2)^2+(n+2)^2(log(n/2)) \in \Theta(n^3(log(n))$ because $(2n)*(n+2)^2$ is roughly n^3, and $log(n+2)^2 * log(/2)$ is roughly $log(n)$
4. $2^{n+1}+3^{n-1} \in \Theta(2^n)$ because both terms are in that growth order already
5. $log_2 n \in \Theta(log_2 n)$ because base must be same to have same growth order

1.3 No. 4

1. Yes, they do prove this fact, because as the value is increasinly plugged into each corresponding fuction, it will show a higher growth once a threshold is reached
2. $log(6) = .78$, $6 = 6$, $6log_2 6 = 15.6$, $6^2 = 36$, $6^3 = 216$, $2^6 = 74$, $6! = 720$ and so on

1.4 No. 5

1. $\sqrt[3]{n}$; $ln^2 n$; $0.001n^4+3n^3+1$; $5lg(n+100)^{10}$; 3^n; 2^{2n}; $(n-2)!$

1.5 No. 12

1. Since we have no idea which direction to door is in, we have to just pick a side, and after a number of steps, assume we went the wrong way and return to the origin and walk that same number of steps in the opposite direction. So, 1 step forward, 2 steps back, 3 steps forwards, 4 steps back, etc etc. This amounts out to $n(2n-1)$ steps to find the door.

2 Section 2.3

2.1 No. 2

1. $\Theta(g(n)) = (i^4 + 2i^2 + 1) * (n-1)$
2. $\Theta(g(n)) = (logi^2) * (n-3)$
3. $\Theta(g(n)) = (i+1)(2^{i-1}) * (n)$
4. $\Theta(g(n)) = ((i+j) * (j-1)) * (n)$

2.2 No. 4

1. It computes the sum of all squares of n
2. Basic operation is to add current square of n to previous sums
3. It is executed n times
4. Efficiency class is linear, since it has only one operation based on the size of n, and will increase at a constant rate
5. The only suggestion I can make is to adjust the for loop to count from 1 to absolute value of n, so the algorithm can also accept negative intergers. Otherwise, you cannot write a simpler version of the basic operation.

2.3 No. 6

1. Checks if two numbers in a matrix A are the same
2. Basic operation is to check if A[i,j] = A[j,i], where i is the current spot in the matrix, and j is a scrolling check of each entry in the matrix
3. It is executed $(n-1)(n-2)$ times
4. Efficiency class in n^2 times, since there are two nested loops, and so as n increases, it effects the amount of times each loop is executed
5. The algorithm seems the best it can be, as both loops are required to check through each entry while still remembering the current one to compare with

2.4 Handout a

1. $(n) * (n^2) * (n^2) = n^5$ steps

2.5 Handout b

1. $\sqrt(n) * 2\sqrt(n) + 3n = 5n$ steps

3 Section 2.4

3.1 No. 3

1. $M(n) = M(n-1) + 3$ for $n > 1$; $M(1) = 1$. Backward substitutions eventually yield an $n-1$ number of times the basic step will be executed, meaning the number of steps taken is equal to the input size.

2. It is the same algorithm, however it is significantly shorter due to the recursive calls. Arguably the better algorithm, just harder to conceive of. Unless size is a huge issue, there is no problem using either one.

3.2 No. 9

1. It computes the largest value in an array A

2. $M(n) = M(n-1)+1$ for $n > 1$; $M(1) = 1$. Backwards substitution yields a $n-2$ number of times the basic step is done.

4 Appendix B

4.1 Problem A

1. The characteristic equation is $x^2 - 8x + 15 = 0$, or $(x-5)(x-3) = 0$, so the general soluion is $T(n) = c_1 5^n + c_2 3^n$. To find c_1 and c_2, we solve equation one, which is $1 = c_1 5^(0) + c_2 3^(0)$, which yields $1 = c_1 + c_2$, or that $c_1 = 1 - c_2$. Then we solve the second equation, which is $4 = c_1 5^1 + c_2 3^1$, which yields $4 = 5(1-c_2) + 3c_2$. Going furthers, it becomes $-1 = -2c_2$, or that $c_2 = 1/2$, meaning $c_1 = 1/2$ as well. This yields the final answer $T(n) = (5^n)/2 + (3^n)/2$.

4.2 Problem B

1. The characteristic equation is $x^2 - 6x + 9 = 0$, or $(x-3)(x-3) = 0$, so the general solution is $T(n) = c_1 3^n + c_2 3^n$. To find c_1 and c_2, we solve equation one, which is $5 = c_1 3^(0) + c_2 3^(0)$, which yields $5 = c_1 + c_2$, or that $c_1 = 5 - c_2$. Then we solve the second equation, which is $9 = c_1 3^1 + c_2 3^1$, which yields $9 = 3c_1 + 3c_2$, or $3 = c_1 + c_2$. I got stuck here, unable to figure out what was wrong with my method.