CS330 **Homework 2**

Topics covered: Time Efficiency Analysis

Deliverables: Write down your answers neatly on a piece of paper. Your name must be indicated clearly on the piece of paper. Then scan it into a.pdf file called cs330_yourid_2.pdf. If you have used more than one piece of paper, please scan them in order.

Objectives: To demonstrate the ability of analyzing the time efficiency of algorithms and recurrence solving.

Homework Questions (5 questions)

1. List the following functions according to their order of growth from the lowest to the highest. (Hint: you could start with using basic asymptotic efficiency classes) (10 points)

$$f_1(n) = n^{2.5}; \quad f_2(n) = \sqrt{2n}; \quad f_3(n) = n + 10$$

$$f_4(n) = 10^n; \quad f_5(n) = 100^n; \quad f_6(n) = n^2 logn$$

2. Consider the following algorithm. (10 points)

```
ALGORITHM Secret(A[0..n − 1])
//Input: An array A[0..n − 1] of n numbers
//Output: An n by n array B[0..n − 1] [0..n − 1]
for i ←0 to n − 1 do
  for j←i to n − 1 do
    Add up array entries A[i] through A[j]
    Store the result in B[i] [j]
return B
```

a) Suppose input A={1,2,3}, and each entry of B is initialized as 0, what is it's output? (2 points)
b) What is its basic operation? (2 points)
c) How many times is the basic operation executed? (3 points)
d) What is the efficiency class (Big O) of this algorithm? (3 points)

3. Solve the following recurrences using characteristic equation (linear homogeneous recurrence with constant coefficients). (10 points)

a) T(n) = 5T(n − 1) − 6T(n − 2) , initial conditions T(0) = 2,T(1) = 5 (5 points)
b) T(n) = 2T(n − 1) + 3 initial condition T(0) = 0 (5 points)

4. Solve recurrence $T(n) = T(n/2) + 1$, with initial condition ,$T(1) = 0$ by
   (a) drawing the recursion tree and summing up a series (5 points)
   (b) using Master's theorem (5 points)
5. Solve recurrence $T(n) = T(n-1) + n$, with initial condition $T(1) = 0$ by
   (a) drawing the tree and summing up a series (5 points)
   (b) using characteristic equation (5 points)

Goh Wei Zhe - CS330 Homework 2

1. $f_1(n) = n^{2.5}$. When n = 10, $f_1(10) = 10^{2.5} = 316.227766$

$f_2(n) = \sqrt{2n}$. When n = 10, $f_2(10) = \sqrt{2(10)} = \sqrt{20} = 4.47213$

$f_3(n) = n + 10$. When n = 10, $f_3(10) = 10 + 10 = 20$

$f_4(n) = 10^n$. When n = 10, $f_4(10) = 10^{10} = 1 \times 10^{10}$

$f_5(n) = 100^n$. When n = 10, $f_5(10) = 100^{10} = 1 \times 10^{20}$

$f_6(n) = n^2 \log n$. When n = 10, $f_6(n) = 10^2 \log 10 = 100$

Order of growth (lowest to highest): $f_2(n)$, $f_3(n)$, $f_6(n)$, $f_1(n)$, $f_4(n)$, $f_5(n)$

2a. Input: array A = {1,2,3} = A[3] , n = 3

Output: array B = B[n][n] = B[3][3]

Int sum initialized to 0.

When i = 0, j = 0. sum += A[j]. sum += A[0] = 0 + 1 = 1. B[0][0] = sum = 1.

When i = 0, j = 1. sum += A[j]. sum += A[1] = 1 + 2 = 3. B[0][1] = sum = 3.

When i = 0, j = 2. sum += A[j]. sum += A[2] = 3 + 3 = 6. B[0][2] = sum = 6.

Int sum reset back to 0.

B[1][0] = 0.

When i = 1, j = 1. sum += A[j]. sum += A[1] = 0 + 2 = 2. B[1][1] = sum = 2.

When i = 1, j = 2. Sum += A[j]. sum += A[2] = 2 + 3 = 5. B[1][2] = sum = 5.

Int sum reset back to 0.

B[2][0] = 0

B[2][1] = 0

When i = 2, j = 2. sum += A[j]. sum += A[2] = 0 + 3 = 3. B[2][2] = 3.

Output: B[3][3] = {1, 3, 6, 0, 2, 5, 0, 0, 3}

2b. Its basic operation is addition. ✓

2c. The basic operation executed 6 times. X -3 1.5

2d. if input array A size n = 1, output array B size = n^2 = 2, basic operation runs 1 time. T (1) = 1

-------------------------- n = 2, --------------------------- n^2 = 4, ------------------ runs 2 + 1 times. T (2) = 3

-------------------------- n = 3, --------------------------- n^2 = 9, ------------- runs 3 + 2 + 1 times. T (3) = 6

-------------------------- n = 4, --------------------------- n^2 = 16, ------ runs 4 + 3 + 2 + 1 times. T (4) = 10

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

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

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

= $N^2/2 + n/2$

= $0.5n^2 + 0.5n$

= $\cancel{0.5}n^2 + \cancel{0.5n}$ E $O(n^2)$

Therefore, the efficiency class of this algorithm is $O(n^2)$. X -3

3a. T(n) = 5T(n-1) – 6T (n – 2) //b = 0, recurrence is homogeneous

T (0) = 2, T (1) = 5, k-degree = = 2

$r^2 - 5r - (-6) = 0$

$r^2 - 5r + 6 = 0$

(r-3) (r-2) = 0

$r_1 = 3$ and $r_2 = 2$ ✓ //root is real and distinct, case 1

$T(n) = a_1 r_1^n + \ldots\ldots + a_k r_k^n$

$= a_1 r_1^n + a_2 r_2^n$

$T(0) = a_1(3)^0 + a_2(2)^0$

$2 = a_1 + a_2$

$a_2 = 2 - a_1$ ---------------------- Equation (1)

$T(1) = a_1(3)^1 + a_2(2)^1$

$5 = 3a_1 + 2a_2$ ------------------ Equation (2)

Substitute equation (1) into equation (2):

$5 = 3a_1 + 2(2 - a_1)$

$5 = 3a_1 + 4 - 2a_1$

$5 = a_1 + 4$

$a_1 = 1$

Substitute $a_1 = 1$ into equation (1):

$a_2 = 2 - a_1$

$= 2 - 1$

$= 1$

$T(n) = a_1 r_1^n + a_2 r_2^n$

$= (1)(3)^n + (1)(2)^n$

$= 3^n + 2^n$

$= 3^n \cancel{+ 2^n}$ E $O(3^n)$

3b. $T(n) = 2T(n - 1) + 3$ //b = 3, recurrence not homogeneous

$T(0) = 0$

$T(n) = 2T(n - 1) + 3$

$T(n + 1) = 2T(n + 1 - 1) + 3$

$= 2T(n) + 3$

$T(n + 1) - T(n) = 2T(n) + 3 - [2T(n - 1) + 3]$

$T(n + 1) = 3T(n) + 3 - 2T(n - 1) - 3$

$T(n + 1) = 3T(n) - 2T(n - 1)$ //b = 0, recurrence is homogeneous

//k – degree = 2

$r^2 - 3r - (-2) = 0$

$r^2 - 3r + 2 = 0$

$r_1 = 2$ and $r_2 = 1$ //roots are real and distinct, case 1

$T(n) = a_1 r_1^n + \ldots\ldots. + a_k r_k^n$

$= a_1 r_1^n + a_2 r_2^n$

$T(0) = a_1(2)^0 + a_2(1)^0$

$0 = a_1 + a_2$ --------------------- Equation (1)

$T(1) = 2T(1 - 1) + 3 = 3$

$T(1) = a_1(2)^1 + a_2(1)^1$

$3 = 2a_1 + a_2$ -------------------- Equation (2)

Substitute equation (1) into equation (2)

$a_1 = 3$

Substitute $a_1 = 1$ back to equation (1)

$0 = a_1 + a_2$

$0 = 3 + a_2$

$a_2 = - 3$

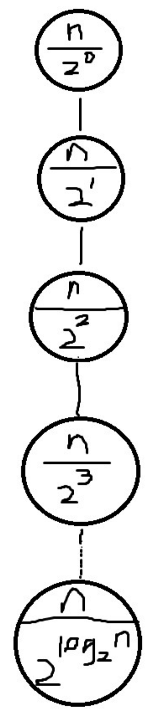
$T(n) = a_1 r_1^n + a_2 r_2^n$

$= (3)(2)^n + (-3)(1)^n$

$= 3(2^n) - 3$

$= 3(2^n) \cancel{-3}$ E $O(2^n)$

4a. $T(n) = T(n/2) + 1$ $\quad T(1) = 0$



Level	Size of node	Number of nodes in each level
0	1	$1^0 = 1$
1	1	$1^1 = 1$
2	1	$1^2 = 1$
$\log_2 n$	1	$1^{\log_2 n} = 1$

$T(n) = 1^0\,(1) + 1^1\,(1) + 1^2(1) + \ldots.. + 1^{\log_2 n}\,(1)$

$= (1)\,(1^0 + 1^1 + 1^2 + \ldots.\ 1^{\log_2 n})$

$= (1)\,(\log_2 n + 1)$

$= \log_2 n + 1$

$= \log_2 n \cancel{+ 1}$ E $O(\log_2 n)$

4b. $T(n) = T(n/2) + 1$ $\quad T(1) = 0$

$T(n) = aT(n/b) + F(n)$ $\quad a = 1, b = 2$

$F(n) = 1$ $\quad n^{\log_b a} = n^{\log_2 1} = n^0 = 1$ $\quad$ //Matched, Case 2

$F(n) = 1$ E $\Theta((n^{\log_2 1})\,(\log n)^{k})$ $\quad$, let $k = 0$

$F(n) = 1$ E $\Theta(n^0)\,(1))$

$F(n) = 1$ E $\Theta\,(1)(1))$

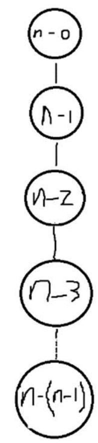
$F(n) = 1$ E $\Theta\,(1)$

$T(n)$ E $\Theta((n^{\log_2 1})\,(\log n)^{k+1})$

$T(n)$ E $\Theta\,(\log n)$

5a. $T(n-1) + n \qquad T(1) = 0$

Level	Size of node	Number of nodes in each level
0	n	$1^0 = 1$
1	$n - 1$	$1^1 = 1$
2	$n - 2$	$1^2 = 1$
N - 1	$n - (n - 1)$	$1^{n-1} = 1$



$T(n) = 1^0(n) + 1^1(n - 1) + 1^2(n - 2) + \ldots + 1^{n-1}(n - (n-1))$

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

$= n(n + 1)/2$

$= 0.5n^2 + 0.5n$

$= \cancel{0.5}n^2 \cancel{+ 0.5n}\ E\ O(n^2)$

5b. $T(n) = T(n-1) + n \qquad T(1) = 0$

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

$T(n+1) - T(n) = T(n+1-1) + n - [T(n-1) + n]$ //Subtraction Method

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

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

$r^2 - 2r - (-1) = 0$

$r^2 - 2r + 1 = 0$

$r_1 = r_2 = 1$

$T(n) = a_1 r_1^n + a_2 n r_2^n$

$= a_1(1)^n + a_2 n (1)^n$

$= a_1 + na_2$

$T(1) = a_1 + (1)a_2$

$0 = a_1 + a_2$ ------------------------- Equation (1)

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

$T(2) = a_1 + (2)a_2$

$2 = a_1 + 2a_2$ ------------------------ Equation (2)

Substitute equation (1) into equation (2)

$2 = 0 + a_2$

$a_2 = 2$

Substitute $a_2 = 2$ back to equation (1)

$0 = a_1 + 2$

$a_1 = -2$

$T(n) = a_1 r_1^n + a_2\, n\, r_2^n$

$\quad = (-2)(1)^n + (2)(n)(1)^n$

$\quad = 2n - 2$

$\quad = \cancel{2}n \cancel{-2}$ E $O(n)$

-5

cs330su21-a.sg

Dashboard / My courses / cs330su21-a.sg / Week 5 (7 June - 13 June) / Assignment 2 - Time Efficiency Analysis

Assignment 2 - Time Efficiency Analysis

cs330-hw2.pdf 8 June 2021, 2:50 PM

Submission status

Submission status	Submitted for grading
Grading status	Graded
Due date	Monday, 14 June 2021, 11:59 PM
Time remaining	Assignment was submitted 2 hours 15 mins early
Last modified	Monday, 14 June 2021, 9:43 PM
File submissions	cs330_weizhe.goh_2.pdf 14 June 2021, 9:43 PM
Submission comments	Comments (0)

Edit submission Remove submission

You can still make changes to your submission.

Feedback

Grade	40.50 / 50.00
Graded on	Monday, 21 June 2021, 6:25 PM
Graded by	Hong Wei CHUA
Annotate PDF	Wei Zhe GOH_15658_0.pdf 21 June 2021, 6:25 PM View annotated PDF...

$f_1(n) = n^{2.5}$; $f_2(n) = \sqrt{2n}$; $f_3(n) = n + 10$

$f_4(n) = 10^n$; $f_5(n) = 100^n$; $f_6(n) = n^2 logn$

step1. use table,
step2. limit method

$f_1(n) = n^{2.5}$ $f_2(n) = \sqrt{2} \cdot n^{0.5}$ $f_3(n) = n^1 + 10$

$f_4(n) = 10^n$ $f_5(n) = 100^n$ $f_6(n) = n^2 (\log n)$?

$f_2(n)$	$f_3(n)$	$f_6(n)$	$f_1(n)$	$f_4(n)$	$f_5(n)$
$n^{0.5}$	n^1	$n^2 \log n$	$n^{2.5}$	10^n	100^n

$f_1(n)$ $n^{2.5} \Rightarrow n^2 \cdot n^{0.5}$

$f_6(n)$ $n^2 \log_e n$

$$\lim_{n\to\infty} \frac{f_6(n)}{f_1(n)} = \lim_{n\to\infty} \frac{\cancel{n^2} \log n}{\cancel{n^2} \cdot n^{0.5}} = \lim_{n\to\infty} \frac{(\log_e n)'}{(n^{0.5})'}$$

$$= \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2} n^{-0.5}} = 2 \lim_{n\to\infty} \frac{n^{0.5}}{n} = 0$$

ALGORITHM *Secret*(A[0..$n-1$])
//Input: An array A[0..$n-1$] of n numbers
//Output: An n by n array B[0..$n-1$] [0..$n-1$]
1. **for** $i \leftarrow 0$ **to** $n-1$ **do**
2. **for** $j \leftarrow i$ **to** $n-1$ **do**
3. Add up array entries $A[i]$ through $A[j]$
4. Store the result in $B[i][j]$
5. **return** B

0

n iteration

< n iteration

Basic Operation: it is not a constant time operation

n.

$A = \{1,2,3\}$

d) (1+

a) $B = \begin{bmatrix} 1 & 3 & 6 \\ 0 & 2 & 5 \\ 0 & 0 & 3 \end{bmatrix}$

3x3

b) Add up:

c) $(1 + 2 + \dots + n)$

d) Add up: $A[i]$ through $A[j..n-1]$
maximum number: $f(n) = n$.

i=0	j=0	A[0] ... A[0]	1
	j=1	A[0] ... A[1]	2
	j=2	A[0] ... A[2]	3
i=1	j=1	A[1] ... A[1]	1
	j=2	A[1] ... A[2]	2
i=2	j=2	A[2] ... A[2]	1

ALGORITHM *Secret*(A[0..$n-1$])
//Input: An array A[0..$n-1$] of n numbers
//Output: An n by n array B[0..$n-1$] [0..$n-1$]
1. **for** $i \leftarrow 0$ **to** $n-1$ **do**
2. **for** $j \leftarrow i$ **to** $n-1$ **do**
3. Add up array entries $A[i]$ through $A[j]$
4. Store the result in $B[i][j]$
5. **return** B

n

$A=\{1,2,3\}$

		# times operation	
$i=0$	$j=0$	1	
	$j=1$	1	3
	$j=2$	1	
$i=1$	$j=1$	1	
	$j=2$	1	2.
$i=2$	$j=2$.	-	1
$(n-1)$			

Generalize it to n.

c) # times basic operation.

$1+2+\ldots+n$

d) order of growth for add up

$O(n)$

$(j-i)$

$i-i$
$i+1-i$
$i+2-i$

$T(n)=(1+2+\ldots n)\cdot n \in O(n^3)$

$n-1-i \in O(n)$

ALGORITHM *Secret*(A[0..$n-1$])
//Input: An array A[0..$n-1$] of n numbers
//Output: An n by n array B[0..$n-1$] [0..$n-1$]
1. **for** $i \leftarrow 0$ **to** $n-1$ **do**
2. **for** $j \leftarrow i$ **to** $n-1$ **do**
3. Add up array entries $A[i]$ through $A[j]$
4. Store the result in $B[i][j]$
5. **return** B

$A=\{1,2,3\}$

c) if basic operation: add up, $\leftarrow O(n)$

times basic operation executed is $(1+2+\ldots+n) \in O(n^2)$

d) $\underbrace{(1+2+\ldots+n)}_{\#\text{ time}}\cdot\, n$.

$\hookrightarrow$ time spending on add-two for add-up

$\in O(n^3)$

add up consisting of addition

maximum $(n-1-i+1)$

$\rightarrow (n-i)[i=0 \ldots n-1]$

$\approx n$.

ALGORITHM *Secret(A*[0..$n-1$]*)*
//Input: An array $A[0..n-1]$ of n numbers
//Output: An n by n array $B[0..n-1][0..n-1]$
1. **for** $i \leftarrow 0$ **to** $n-1$ **do**
2. **for** $j \leftarrow i$ **to** $n-1$ **do**
3. Add up array entries $A[i]$ through $A[j]$
4. Store the result in $B[i][j]$
5. **return** B

n.
$< n$.
$< n$.

$A=\{1,2,3\}$

b) addition (in add up)

c). $< n^3$

d) addition $\Rightarrow O(1)$

$T(n) \in O(n^3)$

$$n^2 \cdot n \Rightarrow n^3$$

$$n^3 \cdot 1 \Rightarrow n^3.$$

$T(n) = 5T(n-1) - 6T(n-2)$, initial conditions $T(0) = 2, T(1) = 5$

$$r^2 - 5r + 6 = 0 \Rightarrow (r-2)(r-3) = 0$$

$$r_1 = 2 \quad r_2 = 3$$

$$T(n) = \alpha_1 \cdot 2^n + \alpha_2 \cdot 3^n$$

$$\left.\begin{array}{l} T(0) = \alpha_1 + \alpha_2 = 2 \\ T(1) = 2\alpha_1 + 3\alpha_2 = 5 \end{array}\right\} \quad \begin{array}{l} \alpha_1 = 1 \\ \alpha_2 = 1 \end{array}$$

$$T(n) = 2^n + 3^n \in \Theta(3^n)$$

T(n) = 2T(n − 1) + 3 initial condition T(0) = 0

$T(1) = 2 \times 0 + 3$

$T(n+1) = 2T(n) + 3.$

$T(n+1) - T(n) = 2T(n) + \cancel{3} - 2T(n-1) - \cancel{3}$

$T(n+1) = 3T(n) - 2T(n-1)$

$T(n) = \alpha_1 2^n + \alpha_2 \cdot 1^n$

$= \alpha_1 \cdot 2^n + \alpha_2$

$r^2 - 3r + 2 = 0$

$(r-2)(r-1) = 0$

$r_1 = 2 \quad r_2 = 1$

$$\left.\begin{aligned} T(0) &= \alpha_1 + \alpha_2 = 0 \\ T(1) &= 2\alpha_1 + \alpha_2 = 3 \end{aligned}\right\}$$

$\alpha_1 = 3 \quad \alpha_2 = -3$

$T(n) = 3 \cdot 2^n + (-3) \in \Theta(2^n)$

T(n) = T(n/2) + 1, with initial condition $\boxed{T(1) = 0}$

	# level	# subproblem	cost of f(n)
n	0	1	1
$n/2$	1	1	1
$n/2^2$	2	1	1
⋮		⋮	⋮
1	$\log_2 n$	1	0

$\log_2 n$

1

$$T(n) = \underbrace{1 + 1 + 1 + \cdots}_{\log_2 n} \underbrace{0}_{1} = \log_2 n \cdot 1 + 0 \times 1 \in \Theta(\log_2 n)$$

T(n) = T(n/2) + 1, with initial condition ,T(1) = 0

master $a=1$ $b=2$ $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

$f(n) = 1$ f(n) compatable with $n^{\log_b a}$

case 2: $f(n) \in \Theta(n^{\log_b a} \cdot (\lg n)^k)$ $k \geq 0$

$\Theta(\log_2 n)$ $1 \in \Theta(1 \cdot (\lg n^k))$? Let $k = 0$.

$1 \in \Theta(1)$ ✓.

$T(n) \in \Theta(n^{\log_b a} \cdot (\lg n)^{k+1}) = \Theta(1 \cdot (\lg n)^1) \in \Theta(\lg n)$

T(n) = T(n-1) + n, with initial condition T(1) = 0

	# level	# subproblems	cost of f(n)
n	0	1	n
H = n. n-1	1	1	n-1
n-2	2	1	n-2
⋮	⋮	⋮	
1	n-1.	1	1

$\in \Theta(n^2)$ T(1) = 0

$T(n) = n + (n-1) + (n-2) + \dots + 2 + 1 + 0$

T(n) = T(n-1) + n, with initial condition T(1) = 0

$(r-1)^3 = 0$

$r_1 = r_2 = r_3 = 1$

$T(n) = \alpha_1 \cdot 1^n + \alpha_2 \cdot n \cdot 1^n + \alpha_3 \cdot n^2 \cdot 1^n$

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

$T(n+1) - T(n) = T(n) \cancel{+n} + 1 - T(n-1) \cancel{-n}$

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

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

$T(n+2) - T(n+1) = 2T(n+1) \underline{-T(n)} \cancel{+1} \underline{-2T(n)} + T(n-1) \cancel{-1}$

$T(n+2) = 3T(n+1) - 3T(n) + T(n-1)$

3 - dgree.

$\boxed{r^3 - 3r^2 + 3r - 1 = 0}$ - ~.