Homework Complexity 1

Jiaxin Wu

1 Exercise One

Sort the functions below according to their growth (so following O(·)). You can give the answer as a sequence of numbers (for example: 1 − 5 − 9 − 8 − 7 − 4 −2 − 3 − 6). Start with the slowest growing function.

1. n	4. n^2	7. $n!$
2. $n - n^3 + 7n^5$	5. $n\ln(n)$	8. e^n
3. 2^n	6. $\sqrt{n}$	9. $\ln(\ln(n))$

Conclusion: 9 – 6 – 1 – 5 – 4 – 2 – 3 – 8 - 7

2 Exercise Two

Suppose we have a computer which can perform 1 million (= 10^6) operations per second. The nine formulas below denote the running time of some algorithms (measured in number of operations) depending on the number of elements n we feed to the algorithm. Determine for each algorithm how many elements can be processed in 1 minute.

1. n N=6×10^7	**5. n ln(n)** 3950157	**9. n^{100}** 1
2. n^2 $\sqrt{60\times10^6} \approx$ 7746	**6. n log(n)** 8309550	**10. 4^n** $\log_4(6\times10^7) \approx$ 12
3. n^3 $\sqrt[3]{60\times10^6} \approx$ 391	**7. 2^n** $\log_2(6\times10^7) \approx$ 25	
4. n! 11	**8. $n\sqrt{n}$** $\sqrt[3]{36\times10^{14}} \approx$ 153261	

3 Exercise Three

3.1 A

1. g(n)=O(f(n))
2. both
3. g(n)=O(f(n))
4. g(n)=O(f(n))
5. both
6. g(n)=O(f(n))
7. f(n)=O(g(n))

3.2 B

Both f(n)=O(g(n)) and g(n)=O(f(n)):
$f(n) = \log(n^2) = 2\log(n)$; $g(n) = \log(n)$
Take c = 2, N=1 to see that $\log(n) \leqslant 2\times2\log(n)$ for all $n > N$;
Take c = 3, N=1 to see that $2\log(n) \leqslant 3\times\log(n)$ for all $n > N$;
So, both f(n)=O(g(n)) and g(n)=O(f(n))

4 Exercise Four

Sum_of_reciprocals (n) =	—
i = 1	$C_1 = 1$
H = 0	$C_2 = 1$
while i <= n :	$C_3 = n+1$
H = H + (1/i)	$C_4 = n$
i = i+1	$C_5 = n$
print (H)	$C_6 = 1$

$$T = 1+1+n+1+n+n+1 = 3n+4 = O(n)$$

5 Exercise Five

factorial (n) =	—
result = 1	$c_1 = 1$
for i in rang(1, n+1) =	$c_2 = n+1$
result *= i	$c_3 = n$
return result	$c_4 = 1$

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

,

6 Exercise Six

Algorithm 1: Bubble Sort

Input: a list A

for $i = 1$ **to** $A.length - 1$ **do**	$c_1 = 1$
for $j = A.length$ **down to** $i+1$ **do**	$c_2 = n$
if $A[j] < A[j-1]$ **then**	$c_3 = \sum_{j=2}^{n} t_j$
exchange $A[j]$ with $A[j-1]$	$c_4 = \sum_{j=2}^{n} (t_j - 1)$
	$c_5 = \sum_{j=2}^{n} (t_j - 1)$

$$T(n) = 1 + c_2 n + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$$

the best condition, $t_j = 1$

$$T(n) = 1 + c_2 n + c_3 (n-1)$$
$$= (c_2 + c_3) n - c_3 + 1$$

the worst condition, $t_j = j$

$$T(n) = \left(\frac{c_3}{2} + \frac{c_4}{2} + \frac{c_5}{2}\right) n^2 + \left(c_2 + \frac{c_3}{2} - \frac{c_4}{2} - \frac{c_5}{2}\right) n - c_3 + 1$$
$$= O(n^2)$$

7 Exercise Seven

Algorithm 2:

Input: n	1
$s = 0$	1
$i = 0$	1
while $i < n$ **do**	$n+1$
$\quad s = s + 1$	n
$\quad i = i + 1$	n

$$
\begin{aligned}
T &= 1+1+1+n+1+n+n \\
&= 3n+1 \\
&= O(n)
\end{aligned}
$$

Algorithm 3:

Input: n	1
$s = 0$	1
$i = 0$	1
while $i < n$ **do**	$n+1$
$\quad j = 0$	n
$\quad$ **while** $j < n$ **do**	$(n+1)^2$
$\qquad s = s + 1$	$(n+1)^2$
$\qquad j = j + 1$	$(n+1)^2$
$\quad i = i + 1$	$n+1$

$$
\begin{aligned}
T &= 1+1+1+n+1+n+(n+1)^2+(n+1)^2+(n+1)^2+n+1 \\
&= 3n^2+9n+8 \\
&= O(n^2)
\end{aligned}
$$

[

Algorithm 4:

Input: n	1
$s = 0$	1
$i = 0$	1
while $i < n$ **do**	$n+1$
$\quad j = 0$	n
$\quad$ **while** $j < n \times n$ **do**	$n(n^2+1)$
$\qquad s = s + 1$	$n \cdot n^2$
$\qquad j = j + 1$	$n \cdot n^2$
$\quad i = i + 1$	n

$$
\begin{aligned}
T &= 1+1+1+n+1+n+n(n^2+1)+n\cdot n^2+n\cdot n^2+n \\
&= 3n^3+4n+4 \\
&= O(n^3)
\end{aligned}
$$

Algorithm 5:	
Input: n	1
$s = 0$	1
$i = 0$	1
while $i < n$ **do**	$n+1$
$j = 0$	n
while $j < i$ **do**	n
$s = s + 1$	$n-1$
$i = i + 1$	n

$$T = 1 + 1 + 1 + n+1 + n + n + n-1 + n$$
$$= 5n + 3$$
$$= O(n)$$

Algorithm 6:	
Input: n	1
$s = 0$	1
$i = 0$	1
while $i < n$ **do**	$n+1$
$j = 0$	n
while $j < i$ **do**	$\sum_1^n i$
$k = 0$	$\sum_1^{n-1} i$
while $k < j$ **do**	$\sum_1^{n-1} i \cdot \sum_1^{n-1} i$
$s = s + 1$	$\sum_1^{n+} i \cdot \sum_1^{n-1} (i-1)$
$k = k + 1$	$\sum_n^{n-1} i \cdot \sum_1^{n-1} (i-1)$
$j = j + 1$	$\sum_1^{n-1} i$
$i = i + 1$	n

$$T = O(n^4)$$