演算法 #Hw2

賴冠廷 陳昀靖

Q1.

- For each of the following pairs of functions, say whether $f(n) = O(g(n))$ and/or $f(n) = \Omega(g(n))$. Justify your answers.

	$f(n)$	$g(n)$
(a)	$100n + logn$	$n + (logn)^2$
(b)	$n2^2$	3^n

Q1.

- 1(a)
  - $f(n) = 100n + logn, g(n) = n + (logn)^2$
  - 令 $x = logn, n = 2^x$, 則 $f'(x) = 100 * 2^x + x, g'(x) = 2^x + x^2$
  - 猜測 $f(n) = \Omega\big(g(n)\big)$
    - 則須滿足存在 c 與 N 使得所有 x $\geq N$, 時$f'(x) \geq cg'(x)$
    - 取 x > 1, c=1 時, $f'(x) > 1 * g'(x)$
    - 因此可以找$N = 2^1, c = 1$滿足$\because n \geq N, f(n) \geq cg(n)$
    - 所以 $f(n) = \Omega\big(g(n)\big)$
  - 猜測 $f(n) = \mathrm{O}\big(g(n)\big)$
    - 則須滿足存在 c 與 N 使得所有 x $\geq N$, 時$f'(x) \leq cg'(x)$
    - 取 x > 2, c=100 時, $f'(x) < 100 * g'(x)$
    - 因此可以找$N = 2^2, c = 100$滿足$\because n \geq N, f(n) \leq cg(n)$
    - 所以 $f(n) = O\big(g(n)\big)$

Q1.

- 1(b)
  - $f(n) = n2^n, g(n) = 3^n$
  - 兩式同除以2^n，得到$f'(n) = n, g'(n) = (\frac{3}{2})^n$
  - 猜測 $f(n) = \Omega\big(g(n)\big)$
    - 則須滿足存在 c 與 N 使得所有 $n \geq N$,時$f'(n) \geq cg'(n)$
    - 指數成長的比一次項快，因此無法找到 c 與 N 滿足此條件
    - 所以 $f(n) \neq \Omega\big(g(n)\big)$
  - 猜測 $f(n) = \mathrm{O}\big(g(n)\big)$
    - 則須滿足存在 c 與 N 使得所有 $n \geq N$,時$f'(n) \leq cg'(n)$
    - 取 n > 1, c=1 時, $f'(n) < 1 * g'(n)$
    - 因此可以找$N = 1, c = 1$滿足$n \geq N, f(n) \leq cg(n)$
    - 所以 $f(n) = O\big(g(n)\big)$

Q2.

- Use Equation 1, to prove that, for every positive integer k,

$$\sum_{i=1}^{n} i^k = O(n^{k+1})$$

Bounding a summation by an integral If $f(x)$ is a monotonically increasing continuous function, then

$$\sum_{i=1}^{n} f(x) \leq \int_{x=1}^{x=n+1} f(x)dx. \qquad (1)$$

Q2.

- $\sum_{i=1}^{n} i^k \leq \int_{x=1}^{x=n+1} x^k dx$

  $= \frac{x^{k+1}}{k+1} + c \Big]_{x=1}^{x=n+1}$

  $= \frac{(n+1)^{k+1}-1}{k+1}$

  Thus, it's $O(n^{k+1})$.

Q3.

- Find the asymptotic behavior of the function $T(n)$ defined by the recurrence relation

$$T(n) = T\left(\frac{n}{2}\right) + \sqrt{n}, T(1) = 1$$

  You can consider only values of n that are powers of 2.

Q3.

- $T(n) = T\left(\frac{n}{2}\right) + \sqrt{n}, n = 2^m$

$$= \left(T\left(\frac{n}{4}\right) + \sqrt{\frac{n}{2}}\right) + \sqrt{n}$$

$$= \left(\left(\ldots\left(T\left(\frac{n}{2^m}\right) + \sqrt{\frac{n}{2^{m-1}}}\right) + \cdots\right) + \sqrt{\frac{n}{2}}\right) + \sqrt{n}$$

$$= T(1) + \sqrt{n}\sum_{i=0}^{i=m-1}\left(\frac{1}{\sqrt{2}}\right)^i = O(\sqrt{n})$$

Q4.

- What is the time complexity of the following program segment? Your answer should be as tight as possible.

```
for i from 1 to n
        for j from 1 to i
                for k from 1 to n
                        · · · ( constant – time operations)
                end for
        end for
end for
```

Q4.

- if i = 1 → j = 1 → k = 1…n → n 次

  if i = 2 → j = 1 → k = 1…n → 2n 次
  j = 2 → k = 1…n

  if i = 3 → j = 1 → k = 1…n → 3n 次
  j = 2 → k = 1…n
  j = 3 → k = 1…n

  …

Q4.

- $(1 + 2 + 3 + \cdots + n)n$

  $= \frac{(1+n)n}{2} n$

  $= \frac{n^3+n^2}{2}$

  Thus, we get time complexity is $O(n^3)$.

Q5.

- Consider the following procedure of insertion sort. In order to sort a list of n elements, we remove the maximum element from the list, recursively sort the rest $n - 1$ elements, and then append the maximum element to the sorted $n - 1$ elements. Assume (1) the running time of removing the maximum element in a list of n elements is n, and (2) the running time of appending one element to a list is 1. Write a recurrence relation for the running time of this version of insertion sort. Give an exact solution for the recurrence relation.

Q5.

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

$$= (InsertSort(n-2) + \big((n-1)+1\big) + (n+1)$$

$$= \big((\ldots InsertSort(0) + (1+1)) + 3\big) + \ldots) + (n+1)$$

$$InsertSort(0) = 0$$

$$0 + \big(2 + 3 + 4 + 5 + \cdots + n + (n+1)\big)$$

$$= 0 + \frac{(2 + (n+1))((n+1) - 1)}{2}$$

$$= \frac{n^2 + 3n}{2}$$

Thus, $O(n^2)$