

COMP 3711 2024

Notes by Marcus Chan

March 8, 2024

Contents

1	Tutorials	2
1.1	Tutorial 1 - Asymptotic notation	2
2	Lectures	3
2.1	Lecture 1 - Asymptotic Notation	3
2.1.1	Lecture 2 - Running time of sorting	5
2.1.2	Selection sort	5
2.1.3	Insertion sort	5
2.1.4	Lecture 3 - Divide and conquer	7

1 Tutorials

1.1 Tutorial 1 - Asymptotic notation

1. (a) true
(b) false
(c) true
(d) false
(e) false
(f) false
(g) true
2. (a) true
(b) false
(c) false
3. (a) yes
(b) no.

let $T_i(n) = i \cdot n$ and $f(n) = n$

$$\begin{aligned} g(n) &= \frac{n(n+1)}{2} \cdot f(n) \\ &= O(n^2 f(n)) \\ &= O(n^3). \end{aligned}$$

$$\therefore g(n) \neq O(f(n)) \text{ and } \neq O(n(f(n))).$$

note: this is because

$$\sum_{i=1}^n O(f(n)) \neq O\left(\sum_{i=1}^n f(n)\right).$$

2 Lectures

2.1 Lecture 1 - Asymptotic Notation

Theorem 2.1

Upper bounds $T(n) = O(f(n))$.

if exists constants $c > 0$ and n_0 such that $\forall n \geq n_0, T(n) \leq c \cdot f(n)$

Theorem 2.2

Lower bounds $T(n) = \Omega(f(n))$

if exists constants $c > 0$ and n_0 such that $\forall n \geq n_0, T(n) \geq c \cdot f(n)$

Theorem 2.3

Tight bounds $T(n) = \Theta(f(n))$

if $T(n) = O(f(n))$ and $T(n) = \Omega(f(n))$

constant $9999^{9999^{9999}} < \text{logarithmic } \log^{10} n < \text{polynomial } n^{0.1} < n \log n < n^2 < \text{exponential } 2^n$

$$1. 9999^{9999^{9999}} = \Theta(1) = O(\log(\log(n)))$$

$$2. \log(\log(n)) = O(\log n)$$

$$3. n^{100} = O(2^n) \text{ Let } n^{100} = c \cdot 2^n \text{ s.t. for } n \geq n_0 n^{100} \leq c \cdot 2^n$$

$$100 * \log(n) = c \cdot n \log(2)$$

$$n = \frac{100}{c} * \log(n)$$

.

$$\therefore \forall c, n^{100} = O(2^n)$$

Theorem 2.4

Common expressions

1. $\max(f(n), g(n)) = \Theta(f(n) + g(n))$
2. $\log \sqrt{n} = \Omega(\sqrt{\log(n)})$
3. $\log(2^n) = \Theta(\log(3^n))$
4. $\sum_{i=1}^n \frac{1}{i} = \Theta(\log n)$

2.1.1 Lecture 2 - Running time of sorting**2.1.2 Selection sort**

Algorithm 1 Selection sort

```

for  $i \leftarrow 1$  to  $n$  do
  for  $j \leftarrow i + 1$  to  $n$  do
    if  $A[j] < A[i]$  then
      swap  $A[j], A[i]$ 
    end if
  end for
end for

```

number of comparisons : $\sum_{i=1}^n (n - i) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$

Theorem 2.5

Correctness of selection sort

Prove by induction. Assume the algorithm sorts every array of size $n-1$ correctly.

1. It first puts the smallest item in $A[1]$
2. then runs selection sort on $A[2 \dots n]$ (by induction this is correct)
3. since $A[1]$ is smaller than every other items, the array is sorted

2.1.3 Insertion sort

Algorithm 2 Insertion sort

```

for  $i \leftarrow 2$  to  $n$  do
   $j \leftarrow i - 1$ 
  while  $j \geq 1$  and  $A[j] > A[j + 1]$  do
    swap  $A[j]$  and  $A[j + 1]$ 
     $j \leftarrow j - 1$ 
  end while
end for

```

1. number of comparisons : $\sum_{i=1}^n (i - 1) = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$
2. average case analysis: only half of the keys are compared
3. best case: on sorted data, takes $O(n)$ time

Theorem 2.6

Comparison of running time

1. $O(\log n) \cup \Theta(2^{\log_2 \log_2 n})$
2. $O(n^4) \cup O(n^3)$

Exercise 2.7Prove that $\log(n!) = \Theta(n \log n)$

1. first, prove $\log(n!) = O(n \log n)$

$$\begin{aligned}
 \log(n!) &= \sum_{i=1}^n \log(i) \\
 &\leq \sum_{i=1}^n \log(n) \\
 &= O(n \log n).
 \end{aligned}$$

2. then, prove $\log(n!) = \Omega(n \log n)$

$$\begin{aligned}
 \log(n!) &= \sum_{i=1}^n \log(i) \\
 &\geq \sum_{i=\frac{n}{2}}^n \log(i) \\
 &\geq \sum_{i=\frac{n}{2}}^n \log\left(\frac{n}{2}\right) \\
 &= \frac{n}{2} \log \frac{n}{2} \\
 &= \frac{n}{2} (\log n - \log 2) \\
 &= \Omega(n \log n).
 \end{aligned}$$

$$\therefore \log(n!) = \Theta(n \log n)$$

2.1.4 Lecture 3 - Divide and conquer

Theorem 2.8

Run time analysis of Binary search

$$\begin{aligned}
T(n) &= T\left(\frac{n}{2}\right) + 2 \text{ if } n > 1, \text{ with } T(1) = 1 \\
&= T\left(\frac{n}{2^2}\right) + 2 \\
&= \dots \\
&= T\left(\frac{n}{2^{\log_2 n}}\right) + 2 \log_2 n \\
&= 1 + 2 \log_2 n.
\end{aligned}$$

Examples

1. Rotated sorted array
2. Find the last 0

Algorithm 3 Tower of hanoi(n, peg1, peg2, peg3)

```

1: if n=0 then return
2: else
3:   Tower of hanoi(n-1, peg1, peg2, peg3)           ▷ T(n-1)
4:   move the only disc from peg 1 to peg 3           ▷ T(1)
5:   Tower of hanoi(n-1, peg2, peg1, peg3)           ▷ T(n-1)
6: end if

```

Theorem 2.9

Recurrence of Tower of hanoi:

$$\begin{aligned}
T(n) &= 2T(n-1) + 1 \\
&= 2(2T(n-2) + 1) + 1 \\
&= \dots \\
&= 2^{n-1}T(1) + 2^{n-2} + 2^{n-3} + \dots + 2^2 + 2 + 1 \\
&= 2^n - 1.
\end{aligned}$$

Algorithm 4 Merge sort(A, l, r)

```
1: if  $l = r$  then return
2: else
3:    $\text{mid} \leftarrow \frac{l+r}{2}$ 
4:   Merge Sort(A, l, mid)  $\triangleright T(\frac{n}{2})$ 
5:   Merge Sort(A, mid+1, r)  $\triangleright T(\frac{n}{2})$ 
6:   Merge(A, l, mid, r)  $\triangleright O(n)$ 
7: end if
```

Theorem 2.10

Analysis of merge sort

$$\begin{aligned} T(n) &\leq T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + O(n) \\ &= 2^i T\left(\frac{n}{2^i}\right) + in && \text{where } i = \log_2 n \\ &= n \log_2 n + n. \end{aligned}$$