EECS 3101 - B Tutorial1 Notes

Shahin Kamali

York University

September 15, 2023

1. Prove that $100n/\log n \in O(n)$.
   Answer: We need to provide n_0 and M s.t. for all $n > n_0$ it holds that $100n/\log n \leq Mn$, that is, $100/\log n \leq M$. Many values of (n_0, M) ensure this inequality holds, e.g., we can set $n_0 = 2$ and $M = 100$. Then for any $n > n_0$, it holds that $100/\log n < 100 = M$.

2. Prove that $100n/\log n \in o(n)$.
   Answer: We need to provide n_0 s.t. for all $n > n_0$ it holds that $100n/\log n < Mn$ for *any value* of M, that is, $100/\log n < M$, or equivalently $100/M < \log n$, that is, $2^{100/M} < n$. Therefore, for any value of M, we can define n_0 to be any value larger than $2^{100/M}$ and then it holds that $100n/\log n < Mn$.

3. Prove that $n^2 + 2022n \in o(n^2 \log\log n)$.
   Answer: We need to provide n_0 s.t. for all $n > n_0$ it holds that $n^2 + 2022n < M(n^2 \log\log n)$ for any value of M. Suppose $n > 2022$, which gives $n^2 + 2022n < 2n^2$. Therefore, assuming $n > 2022$, in order to prove $n^2 + 2022n < M(n^2 \log\log n)$, it suffices to prove $2n^2 < M(n^2 \log\log n)$ or equivalently $2/M < \log\log n$, that is $2^{2^{2/M}} < n$. To conclude, for any value of M, we can define n_0 to be any value larger than $\max\{2022, 2^{2^{2/M}}\}$ and then it holds that $n^2 + 2022n < M(n^2 \log\log n)$.

4. Prove that $n \log n/2022 \in \Omega(n)$.
   Answer: We need to provide n_0 and M s.t. for all $n > n_0$ it holds that $n \log n/2022 \geq Mn$, that is, $\log n/2022 \geq M$. Many values of (n_0, M) ensure this inequality holds, e.g., we can set $n_0 = 2$ and $M = 1/2022$. Then for any $n > n_0$, it holds that $n \log n/2022 \geq Mn$.

5. Prove that $n \log n/2022 \in \omega(2022n)$.
   Answer: We need to provide n_0 s.t. for all $n > n_0$ it holds that $n \log n/2022 > M2022n$ for any value of M, that is, $\log n/2022 > 2022M$ or $n > 2^{2022^2 M}$. Therefore, for any value of M, we can define n_0 to be any value larger than $2^{2022^2 M}$ and then it holds that $n \log n/2022 > 2022n$.

6. Prove that $n^2 + 2022n \in \omega(n \log n)$.
   Answer: We need to provide n_0 s.t. for all $n > n_0$ it holds that $n^2 + 2022n > M(n \log n)$ for any value of M, that is, $n + 2022 > M \log n$. To prove $n + 2022 > M \log n$, it suffices to prove $n > M \log n$ or $n/\log n > M$. We note that for $n > 4$, it holds that $\log n < \sqrt{n}$; therefore, assuming $n > 4$, to prove $n/\log n > M$, it suffices to prove $n/\sqrt{n} > M$, that is $\sqrt{n} > M$ or $n > M^2$. To conclude, for any value of M, we can define n_0 to be any value larger than $\max\{4, M^2\}$ and then it holds that $n + 2022 > M \log n$.

7. Prove that $2^n \in \Theta(2^{n+2022})$. **Answer:** We need to provide n_0, M_1 and M_2 s.t. for all $n > n_0$ it holds that $M_1 2^{n+2022} \leq 2^n \leq M_2 2^{n+2022}$, or equivalently, $M_1 2^{2022} \leq 1 \leq M_2 2^{2022}$. Many values of (n_0, M_1, M_2) ensure these inequalities hold, e.g., we can set $n_0 = 1$, $M_1 = 1/2^{2022}$ and $M_2 = 1$.

8. State the relationship between 2^n and 2^{2n} and prove it!
Answer: We have $2^n \in o(2^{2n})$. To prove it, we need to show provide n_0 such that for all values of $n > n_0$, it holds that $2^n < M.2^{2n}$ for any value of M. That is, $1 < M \cdot 2^n$, or $\log(1/M) < n$. To conclude, for any value of M, we can define n_0 to be any value larger than $\log(1/M)$ and then it holds that $2^n < M.2^{2n}$.

9. Prove that if $f(n) \in O(g(n))$ and $g(n) \in o(h(n))$ then $f(n) \in o(h(n))$.
Answer: Since $f(n) \in O(g(n))$, there exist values of n_{00} and M_0 s.t. for all $n > n_{00}$, it holds that $f(n) < M_0 g(n)$. Moreover, since $g(n) \in o(h(n))$, there is a value of n_{01} s.t. for all $n > n_{01}$, it holds that $g(n) < M' h(n)$ for all values of M'. Therefore, as long as $n > \max\{n_{00}, n_{01}\}$, for any value of M', we can write
$$f(n) < M_0 M' h(n) \tag{1}$$
.
To prove $f(n) \in o(h(n))$, we need to provide n_0 s.t. for all values of M and for $n > n_0$ it holds that $f(n) < M h(n)$. Fix *any* value of M, and let $n_0 = \max\{n_{00}, n_{01}\}$. Apply Inequality (1) for $M' = M/M_0$ (we can do it because Inequality (1) holds for all values of M'). Then we can write $f(n) < M h(n)$ which completes the proof.

10. Prove that if $f(n) \in \Theta(g(n))$ and $g(n) \in \omega(h(n))$ then $f(n) \in \omega(h(n))$.
Answer: Since $f(n) \in \Theta(g(n))$, there exist values of n_{00}, M_1 and M_2 s.t. for all $n > n_{00}$, it holds that $M_1 g(n) \leq f(n) \leq M_2 g(n)$. Moreover, since $g(n) \in \omega(h(n))$, there is a value of n_{01} s.t. for all $n > n_{01}$, it holds that $g(n) > M' h(n)$ for all values of M'. Therefore, as long as $n > \max\{n_{00}, n_{01}\}$, for any value of M', we can write
$$f(n) \geq M_1 g(n) > M_1 M' h(n) \tag{2}$$
.
To prove $f(n) \in \omega(h(n))$, we need to provide n_0 s.t. for all values of M and for $n > n_0$ it holds that $f(n) > M h(n)$. Fix *any* value of M, and let $n_0 = \max\{n_{00}, n_{01}\}$. Apply Inequality (2) for $M' = M/M_1$ (we can do it because Inequality (1) holds for all values of M'). Then we can write $f(n) > M h(n)$ which completes the proof.

11. What is the time complexity of the following algorithm?

```
Algo2(A, n)
    max ← 0
    for i ← 1 to n do
        for j ← i to n do
            X ← 0
            for k ← i to j do
                X ← A[k]
                if X > max then
                    max ← X
    return max
```

Answer: The time complexity is (e, d, c are constant number of primitive operations):

$$
\begin{aligned}
T(n) &= e + \sum_{i=1}^{n} \sum_{j=i}^{n} (d + \sum_{k=i}^{j} c) \\
&= e + \sum_{i=1}^{n} \sum_{j=i}^{n} (d + (j - i + 1)c) \\
&= e + \sum_{i=1}^{n} \sum_{p=1}^{n-i+1} (d + pc) \\
&= e + \sum_{i=1}^{n} (d(n - i + 1) + c \sum_{p=1}^{n-i+1} p) \\
&= e + \sum_{i=1}^{n} (d(n - i + 1) + c(n - i + 1)(n - i + 2)/2) \\
&= e + \sum_{q=1}^{n} (dq + cq(q + 1)/2) \\
&= e + \sum_{q=1}^{n} ((d + c/2)q + cq^2/2) \\
&= e + (d + c/2) \sum_{q=1}^{n} q + c/2 \sum_{q=1}^{n} q^2 \\
&= e + (d + c/2)n(n + 1)/2 + c/2(n(n + 1)(2n + 1)/6) \\
&= n^3/6 + o(n^3) = \Theta(n^3)
\end{aligned}
$$

12. What is the time complexity of the following algorithm?

```
Algo4(n)
    A ← 0
    for i ← 1 to n do
        for j ← 1 to n do
            if j < n/3 then
                A ← A/(i − j)^2
                A ← A^100
            else
                k ← i
                while k > 1
                    A ← A^2022
                    k ← k/3
    return sum
```

Answer: The time complexity is (e, d, c are constant number of primitive operations):

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{n}(\sum_{j=1}^{n/3} c + \sum_{j=n/3+1}^{n} (e + d\log_3 i)) \\
&= \sum_{i=1}^{n}((n/3)c + (2n/3)(e + d\log_3 i)) \\
&= n(n/3)c + n(2n/3)e + d(2n/3)\sum_{i=1}^{n} \log_3 i \\
&= n(n/3)c + n(2n/3)e + \frac{d(2n/3)}{\log 3}\sum_{i=1}^{n} \log i \\
&= n(n/3)c + n(2n/3)e + \frac{d(2n/3)}{\log 3}\Theta(n\log n) \\
&= \Theta(n^2 \log n)
\end{aligned}
$$