Lecture Notes: Asymptotics Beyond Polynomials

---

1. Review of Asymptotics

- For polynomial-like functions, constants in front don't matter.
  Example:
  - 1.5n2+2.2nlogn+3n~Θ(n2)1.5n^2 + 2.2n\log n + 3n \sim \Theta(n^2)1.5n2+2.2nlogn+3n~Θ(n2).
- Constants outside the function can be ignored in Θ\ThetaΘ-notation.

---

2. Exponential Functions

- **Key difference:** For exponential functions, constants *in the exponent* **cannot** be ignored.
- Example:
  - f(n)=1.2⋅2nf(n) = 1.2 \cdot 2^nf(n)=1.2⋅2n
  - g(n)=2.4⋅22ng(n) = 2.4 \cdot 2^{2n}g(n)=2.4⋅22n
  - Ignoring front constants: okay (1.2,2.41.2, 2.41.2,2.4).
  - Ignoring exponent constants: **not okay**.
- Why?
  - 22n=(2n)22^{2n} = (2^n)^222n=(2n)2.
  - This is squaring the function, not doubling.

- Correct asymptotics:
    - f(n)=O(g(n))f(n) = O(g(n))f(n)=O(g(n))
    - g(n)=Ω(f(n))g(n) = \Omega(f(n))g(n)=Ω(f(n))
    - But **not** f(n)=Θ(g(n))f(n) = \Theta(g(n))f(n)=Θ(g(n)).

---

3. Different Bases in Exponents

- Compare 2n2^n2n vs 3n3^n3n.
    - 3n=2(log23) n≈21.57n3^n = 2^{(\log_2 3)\,n} \approx 2^{1.57n}3n=2(log23)n≈21.57n.
    - Exponent bases matter; 2n2^n2n and 3n3^n3n are **not** Θ\ThetaΘ-equivalent.
- Constants multiplying outside (7.9⋅2n7.9 \cdot 2^n7.9⋅2n) can be ignored.
- **Rule of thumb:** Constants in front → ignore; constants in exponent/base → **keep**.

---

4. Logarithms

- Logarithms are inverses of exponentials:
    - If c=logabc = \log_a bc=logab, then ac=ba^c = bac=b.
    - Example: log216=4 ⇔ 24=16\log_2 16 = 4 \iff 2^4 = 16log216=4⇔24=16.
- Change of base rule:
  logan=logbnlogba\log_a n = \frac{\log_b n}{\log_b a}logan=logbalogbn
  → Any two log bases differ only by a constant.
- **Implication:**
    - log2n\log_2 nlog2n and log5n\log_5 nlog5n are Θ\ThetaΘ-equivalent.

- Logarithms of different bases can be treated as the same asymptotically.

---

5. Complexity Classes of Algorithms

- **Examples so far:**
  - Binary Search → Θ(logn)\Theta(\log n)Θ(logn).
  - Insertion Sort → Θ(n2)\Theta(n^2)Θ(n2).
  - Merge Sort → Θ(nlogn)\Theta(n \log n)Θ(nlogn).
- Growth comparison:
  logn≪n≪nlogn≪n2≪n3≪n4…\log n \ll n \ll n\log n \ll n^2 \ll n^3 \ll n^4 \dotslogn≪n≪nlogn≪n2≪n3≪n4…

---

6. Polynomial vs. Exponential Algorithms

- **Polynomial time algorithms** (nkn^knk for some constant kkk):
  - Efficient, tractable.
  - Examples: sorting (nlognn \log nnlogn), matrix multiplication (O(n3)O(n^3)O(n3)).
- **Exponential time algorithms** (2n,3n,n!2^n, 3^n, n!2n,3n,n!):
  - Intractable for large nnn.
  - Examples: SAT (satisfiability), Traveling Salesperson, Graph Coloring.

---

7. Intermediate Growth Rates

- Between polynomial and exponential:

    - Polylogarithmic exponents: $2^{(\log n)^2}$, $2^{(\log n)^4}$.
    - Called **quasi-polynomial (polylog)** time.
    - Examples: certain factoring algorithms, special graph algorithms.

---

8. Spectrum of Algorithm Complexities

- **Constant time:** $\Theta(1)$.
    - Example: check if integer n is even (look at last binary digit).
- **Logarithmic:** $\Theta(\log n)$.
- **Linear-logarithmic:** $\Theta(n \log n)$.
- **Polynomial:** $n^2, n^3, n^4, \dots$
- **Exponential:** $2^n, 3^n$.
- **Factorial:** $n!$.
- **Non-elementary:** Growth like $2^{2^{2^n}}$, even larger.

---

9. Key Takeaways

1. Constants **outside** exponentials → ignore.
2. Constants **in the exponent/base** → never ignore.
3. Logarithm bases differ by a constant → all $\Theta(\log n)$.
4. Algorithms fall into broad classes:

- Constant → Logarithmic → Polynomial → Quasi-polynomial → Exponential → Factorial → Non-elementary.

5. Polynomial-time algorithms are considered “efficient,” exponential-time are generally “intractable.”

✅ Next topic in the series: **complexity classes** (connecting algorithm families to PPP, NPNPNP, etc.).