

Course Algorithms · 7M06116 – Computer Sciences and Technologies

Mathematical Analysis of Nonrecursive Algorithms

Textbook: A. Levitin, Section 2.3

Acting Professor Ualiyeva Irina Maratovna

Today’s goal

Turn pseudocode with loops into a **precise formula** for its running time.

- a five-step general plan
- the summation toolkit
- five worked examples with different loop shapes
- worst, best and average cases

The general plan

1. **Input size:** choose the parameter, e.g. n
2. **Basic operation:** usually in the innermost loop
3. **Cases:** does the count depend on the particular input?
4. **Set up a sum** for the number of executions
5. **Simplify** it and find the order of growth

The plan is mechanical on purpose: it forces every assumption into the open.

Summation toolkit: two rules

$$\sum_{i=l}^{u} c\, a_i = c \sum_{i=l}^{u} a_i$$

$$\sum_{i=l}^{u} (a_i \pm b_i) = \sum_{i=l}^{u} a_i \pm \sum_{i=l}^{u} b_i$$

Counting the terms from *l* to *u*:

$$\sum_{i=l}^{u} 1 = u - l + 1$$

Standard formulas

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \in \Theta(n^2)$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \in \Theta(n^3)$$

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1 \in \Theta(2^n)$$

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

Example 1: the largest element

```
MaxElement(A[0..n−1])
  maxval ← A[0]
  for i ← 1 to n − 1 do
    if A[i] > maxval
      maxval ← A[i]
  return maxval
```

Basic operation: the **comparison**. It runs on every iteration.

No case analysis needed.

$$C(n) = \sum_{i=1}^{n-1} 1 = n - 1 \in \Theta(n)$$

Example 2: element uniqueness

```
UniqueElements(A[0..n−1])
  for i ← 0 to n − 2 do
    for j ← i + 1 to n − 1 do
      if A[i] = A[j] return false
  return true
```

The count **depends on the input**: if the first two elements are equal, we stop after one comparison.

Uniqueness: the worst case

$$C_{worst}(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n-1-i)$$

$$C_{worst}(n) = (n-1) + \ldots + 1 = \frac{(n-1)n}{2} \in \Theta(n^2)$$

- Worst case: **Θ(n²)**. Best case: **Θ(1)**.
- So say “worst-case Θ(n²)” or “O(n²)”, not just “Θ(n²)”.

Example 3: matrix multiplication

```
for i ← 0 to n − 1 do
  for j ← 0 to n − 1 do
    C[i, j] ← 0
    for k ← 0 to n − 1 do
      C[i, j] ← C[i, j] + A[i, k] · B[k, j]
```

$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1 = n^3 \in \Theta(n^3)$$

$$T(n) \approx (c_m + c_a)\, n^3$$

Example 4: a dependent bound

```
for i ← 1 to n do
  for j ← 1 to i do
    count ← count + 1
```

$$\sum_{i=1}^{n} \sum_{j=1}^{i} 1 = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \in \Theta(n^2)$$

Selection sort and bubble sort have this shape. The leading term is always **n²/2**.

Example 5: a doubling counter

```
i ← 1
while i ≤ n do
  // basic operation
  i ← 2 · i
```

After *t* steps $i = 2^t$. The loop stops when $2^t > n$.

$$\lfloor \log_2 n \rfloor + 1 \in \Theta(\log n)$$

$$\log_a n = \frac{\log_b n}{\log_b a}$$

The base of the logarithm changes only a constant factor.

Common mistakes

- **Counting the wrong thing:** pick the operation done most often
- **Size vs. value:** a loop to the value n is exponential in its digits
- **Mixing cases:** say which case a bound refers to
- **Hidden costs:** early exits, library calls

Check yourself: count the basic operation in code and compare with C(n).

Summary

- Five steps: size, basic operation, cases, sum, simplify
- Loops become sums; nested loops become nested sums
- Maximum: $\Theta(n)$; uniqueness: $\Theta(n^2)$ worst, $\Theta(1)$ best
- Matrix multiplication: $\Theta(n^3)$
- Doubling or halving counters: $\Theta(\log n)$

Review questions

1. List the five steps. Why is the comparison, not the assignment, the basic operation in MaxElement?
2. Derive the worst case of UniqueElements. Why is “$\Theta(n^2)$ time” imprecise?
3. Evaluate $\sum_{i=0}^{n-1} \sum_{j=i}^{n-1} 1$ and give its order of growth.
4. Average comparisons in sequential search with success probability p?