Algorithms

Precise description of a mathematical procedure for solving a problem.

- Precise
- Correct
- Finite Running Time

Input → Output

Pseudo code description

Proof of Correctness

Running Time Analysis

Uniform Cost Model

Every primitive operation (+ - * / if < <= > >= !=)

take 1 unit of time each.

Compute time taken by an algorithm for an input of size n

and express RT in using Order notation (O(\text{log}n))

Sums of Sequences

Sum of arithmetic sequence (e.g., 1 + 2 + 3 + ... + n)

= \Theta(\text{Biggest term in sequence} \times \# \text{ of elements in sequence})

Sum of geometric sequence (e.g., 1 + 2 + 4 + 8 + ... + n)

= \Theta(\text{Biggest term in sequence})

Sum of harmonic sequence (e.g., 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + ... + \frac{1}{n})

= \Theta(\text{log} n)
Bubble Sort

BubbleSort (A,n)  //Sort A[1..n]
for i ← 1 to n do
    for j ← n downto i+1 do

Merge Sort

MergeSort (A, p, r)
// Sort A[p..r].  Let n = r-p+1
if p < r then
    q ← (p+r)/2
    MergeSort (A, p, q)
    MergeSort (A, q+1, r)
// Merge sorted subarrays
// A[p..q] and A[q+1..r]
Merge (A, p, q, r)

Merge (A, p, q, r)
Copy A[p..q] into L[1..q-p+1]
Copy A[q+1..r] into R[1..r-q]
Set L[q-p+2]=R[r-q+1]=∞
i ← 1;  j ← 1
for k ← p to r do
    if L[i] < R[j] then
        A[k] ← L[i++]
    else
        A[k] ← R[j++]
Running time of MergeSort \((A, p, r)\):

Let \(n = r - p + 1\). Let \(T(n)\) be \(RT\) of mergeSort when \(n = r - p + 1\).

Each of the recursive calls takes runs in \(T(n/2)\) each.

Recurrence for \(T(n)\):

\[
T(n) \leq 2T(n/2) + cn
\]

Base case:

\[
T(1) = 1
\]

Solving recurrence:

1. Iteration method

\[
T(n/2) \leq 2T(n/4) + cn/2
\]

\[
T(n) \leq 2 \left( 2T(n/4) + cn/2 \right) + cn
= 4T(n/4) + cn + cn
\]

Repeat:

\[
T(n/4) \leq 2T(n/8) + c \cdot n/4
\]

Substitute into previous inequality:

\[
T(n) \leq 4 \left( 2T(n/8) + c \cdot n/4 \right) + cn + cn
= 8T(n/8) + \frac{cn + cn + cn}{3cn}
\]

After \(k\) steps,

\[
T(n) \leq 2^k T(n/2^k) + k \cdot cn
\]

[Prove by induction on \(k\) that this is correct].

Choose \(k\) such that \(\frac{n}{2^k} = 1 \Rightarrow k = \log_2 n\)

\[
T(n) \leq 2^{\log_2 n} T(1) + \log n \cdot cn
= n \cdot 1 + c \cdot n/\log n = \Theta(n \log n)
\]
2. Substitution Method:

\[ T(n) \leq T(n/2) + n \]
\[ T(1) = 1 \]

**Guess:** \( T(n) = O(n) \).

**Verify:** Induction Hypothesis: \( T(n) \leq cn \) for some constant \( c \).

**Proof by induction \( n \geq 1 \):**

**Base:** \( n = 1 \) \( T(1) = 1 \leq c \cdot 1 \implies c \geq 1 \).

**Step:** Consider some \( n > 1 \).

\[ T(n) \leq T(n/2) + n \quad \text{by recurrence} \]
\[ \leq c \cdot n/2 + n \quad \text{by I.H.} \]
\[ = n \left( \frac{c}{2} + 1 \right) \]

For the proof to work, we need \( n \left( \frac{c}{2} + 1 \right) \leq cn \).

We need \( \frac{c}{2} + 1 \leq c \)

\[ \implies 1 \leq c - \frac{c}{2} = \frac{c}{2} \]
\[ \implies c \geq 2 \]

Choose \( c = 2 \) — base/step are valid.

\( \implies \) by induction, \( T(n) \leq 2n = O(n) \).

---

If guess is too big, verification will succeed but answer is not the best possible.

**e.g.:** \( T(n) \leq 2T(n/2) + n \) \quad **Guess:** \( T(n) = O(n^2) \)

\[ T(1) = 1 \]

Proof works but \( T(n) = O(n \log n) \) is a better guess.

**Verify:** \( T(n) \leq cn \log n + dn \) \quad constant.
\[ T(n) \leq T\left(\frac{7}{10} n\right) + T\left(\frac{n}{5}\right) + n \]

- Selection problem

\[ T(1) = 1 \]

Substitution: \( T(2n) = O(n) \).

3. Recursion tree method:

\[ T(n) \leq 2T\left(\frac{n}{2}\right) + n \]

\[ T(n) \leq T\left(\frac{7}{10} n\right) + T\left(\frac{n}{5}\right) + n \]

Worst case: \( n + \frac{9}{10} n + \left(\frac{9}{10}\right)^2 n + \ldots \)

\[ = O(n) \]

(Geometric sequence)