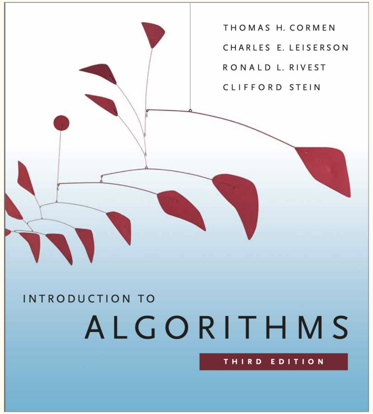
Data Structures and Algorithms

Tutorial 4. Solving recurrences

Today's topic is covered in detail in

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. **Introduction to Algorithms.** The MIT Press 2009.



I Foundations

Objectives

- Divide and Conquer
- Recurrences
- Solving recurrences: Substitution method
- Solving recurrences: Master method

Divide and Conquer

1. Divide
2. Conquer
3. Combine

Divide and Conquer recurrence

$$T(n) = a \cdot T(n/b) + f(n)$$

Overall running time
of a recursive function

Divide and Conquer recurrence

\# of recursive calls

$$T(n) = a \cdot T(n/b) + f(n)$$

Overall running time
of a recursive function

Divide and Conquer recurrence

of recursive calls

$$T(n) = a \cdot T(n/b) + f(n)$$

Overall running time
of a recursive function

Size of input
after going into
recursive call

Divide and Conquer recurrence

of recursive calls

Cost of
divide + combine

$$T(n) = a \cdot T(n/b) + f(n)$$

Overall running time
of a recursive function

Size of input
after going into
recursive call

Exercise: extracting recurrence relation

Exercise 4.1. Derive a recurrence to characterize the running time of the following recursive function:

SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)

$n = A.rows$
let C be a new $n \times n$ matrix
if $n == 1$
    $c_{11} = a_{11} \cdot b_{11}$
else partition A, B, and C as in equations (4.9)
    $C_{11} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{11})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{12}, B_{21})
    $C_{12} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{12})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{12}, B_{22})
    $C_{21} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{21}, B_{11})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{22}, B_{21})
    $C_{22} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{21}, B_{12})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{22}, B_{22})
return C

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}, \qquad (4.9)$$

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solution. We guess that $T(n) = O(n \log n)$.

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solution. We guess that $T(n) = O(n \log n)$. The substitution method requires us to prove that $T(n) \leq c \cdot n \cdot \log n$ for an appropriate choice of $c > 0$.

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solution. We guess that $T(n) = O(n \log n)$. The substitution method requires us to prove that $T(n) \leq c \cdot n \cdot \log n$ for an appropriate choice of $c > 0$.
Inductive hypothesis: for all $m < n$ (including $m = n/2$) we have $T(m) \leq c \cdot m \cdot \log m$.

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solution. We guess that $T(n) = O(n \log n)$. The substitution method requires us to prove that $T(n) \leq c \cdot n \cdot \log n$ for an appropriate choice of $c > 0$.
Inductive hypothesis: for all $m < n$ (including $m = n/2$) we have $T(m) \leq c \cdot m \cdot \log m$.
So, $T(n) =$

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solution. We guess that $T(n) = O(n \log n)$. The substitution method requires us to prove that $T(n) \leq c \cdot n \cdot \log n$ for an appropriate choice of $c > 0$.

Inductive hypothesis: for all $m < n$ (including $m = n/2$) we have $T(m) \leq c \cdot m \cdot \log m$.

So, $T(n) = 2T(n/2) + n$

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solution. We guess that $T(n) = O(n \log n)$. The substitution method requires us to prove that $T(n) \leq c \cdot n \cdot \log n$ for an appropriate choice of $c > 0$.
Inductive hypothesis: for all $m < n$ (including $m = n/2$) we have $T(m) \leq c \cdot m \cdot \log m$.
So, $T(n) = 2T(n/2) + n \leq 2c \cdot n/2 \cdot \log (n/2) + n$

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solution. We guess that $T(n) = O(n \log n)$. The substitution method requires us to prove that $T(n) \leq c \cdot n \cdot \log n$ for an appropriate choice of $c > 0$.

Inductive hypothesis: for all $m < n$ (including $m = n/2$) we have $T(m) \leq c \cdot m \cdot \log m$.

So, $T(n) = 2T(n/2) + n \leq 2c \cdot n/2 \cdot \log (n/2) + n$

$= c \cdot n \cdot \log n - c \cdot n \cdot \log 2 + n$

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solution. We guess that $T(n) = O(n \log n)$. The substitution method requires us to prove that $T(n) \leq c \cdot n \cdot \log n$ for an appropriate choice of $c > 0$.
Inductive hypothesis: for all $m < n$ (including $m = n/2$) we have $T(m) \leq c \cdot m \cdot \log m$.
So, $T(n) = 2T(n/2) + n \leq 2c \cdot n/2 \cdot \log (n/2) + n$
$= c \cdot n \cdot \log n - c \cdot n \cdot \log 2 + n \leq c \cdot n \cdot \log n - c \cdot n + n$

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solution. We guess that $T(n) = O(n \log n)$. The substitution method requires us to prove that $T(n) \leq c \cdot n \cdot \log n$ for an appropriate choice of $c > 0$.
Inductive hypothesis: for all $m < n$ (including $m = n/2$) we have $T(m) \leq c \cdot m \cdot \log m$.
So, $T(n) = 2T(n/2) + n \leq 2c \cdot n/2 \cdot \log (n/2) + n$
$= c \cdot n \cdot \log n - c \cdot n \cdot \log 2 + n \leq c \cdot n \cdot \log n - c \cdot n + n \leq c \cdot n \cdot \log n$ (when $c > 1$).

Solving recurrences: substitution method

1. Guess the form of solution.
2. Use mathematical induction to show that the solution works.

Example. Determine upper bound on the recurrence

$$T(n) = 2T(n/2) + n$$

Solution. We guess that $T(n) = O(n \log n)$. The substitution method requires us to prove that $T(n) \leq c \cdot n \cdot \log n$ for an appropriate choice of $c > 0$.
Inductive hypothesis: for all $m < n$ (including $m = n/2$) we have $T(m) \leq c \cdot m \cdot \log m$.
So, $T(n) = 2T(n/2) + n \leq 2c \cdot n/2 \cdot \log (n/2) + n$
$= c \cdot n \cdot \log n - c \cdot n \cdot \log 2 + n \leq c \cdot n \cdot \log n - c \cdot n + n \leq c \cdot n \cdot \log n$ (when $c > 1$).
To complete the proof, we need to check the boundary conditions (base of induction).

Exercise: substitution method

Exercise 4.2. Show that the solution of $T(n) = T(n - 1) + n$ is $O(n^2)$.

Exercise 4.3. (*) Solve the recurrence $T(n) = 3T(\sqrt{n}) + \log n$. Consider change of variables $m = \log n$. Your solution should be asymptotically tight. Do not worry whether values are integral.

Attendance

https://baam.duckdns.org

Solving recurrences: the master method

The master theorem is a recipe that is easy to use for many naturally occurring divide-and-conquer recurrences.

All you have to do is **memorize 3** cases of the master theorem.

Theorem 4.1 (Master theorem)

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n) ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Solving recurrences: the master method

The master theorem is a recipe that is easy to use for many naturally occurring divide-and-conquer recurrences.

All you have to do is **memorize** 3 cases of the master theorem.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Exercise: the master method

Exercise 4.4. Apply the master method to the following recurrences:

1. $T(n) = 2T(n/4) + 1$
2. $T(n) = 2T(n/4) + \sqrt{n}$
3. $T(n) = 2T(n/4) + n$
4. $T(n) = 2T(n/4) + n^2$

Exercise: the master method

Exercise 4.5. Can the master method be applied to the following recurrence?

$$T(n) = 4T(n/2) + n^2 \log n$$

Why or why not? Give an asymptotic upper bound for this recurrence.

Exercise 4.6. (*) Consider the regularity condition $a \cdot f(n/b) \leq c \cdot f(n)$ for some constant $c < 1$, which is part of case 3 of the master theorem. Give an example of constants $a \geq 1$ and $b > 1$ and a function $f(n)$ that satisfies all the conditions in case 3 of the master theorem **except** the regularity condition.

Summary

- Divide and Conquer
- Recurrences
- Solving recurrences: substitution method
- Solving recurrences: the master method

Summary

- Divide and Conquer
- Recurrences
- Solving recurrences: substitution method
- Solving recurrences: the master method

See you next week!