CS 2500: Algorithms

Lecture 7: Solving Recurrence Equations

Shubham Chatterjee

Missouri University of Science and Technology, Department of Computer Science

September 10, 2024

Questions and Concerns: Discussions

- Why is the course so hard?
- Why are you teaching “more” than what Dr. Morales is teaching in the other section?
- There are too many home works. We can’t keep up!

Solving Recurrence Equations

Last class. Solving Recurrence Equations using: Guess-and-Verify
This class. Solving recurrence equations using:

- Iteration/Substitution method
- Recurrence-tree method
- Telescoping/Difference method

Substitution Method

- Also known as iterative method.
- One of the main ways of solving recurrences.
- The solution is obtained by repeated substitution of the RHS of the recurrence till a pattern is obtained.
- **Forward substitution.** Solution obtained by repeated substitution from the base condition onwards.
- **Backward substitution.** Substitution starts from the last term and proceeds to the initial term.
- Both involve two steps:
  1. **Plug:** Substitute repeatedly.
  2. **Chug:** Simplify the expression.

Substitution Method: Example 1

Solve the following recurrence equation:

$$t_n = t_{n-1} + 3 \quad \text{where } t_1 = 4$$

Solution: Substituting the values of t_{n-1} in the recurrence equation:

$$\begin{aligned} t_n &= (t_{n-2} + 3) + 3 \\ &= t_{n-2} + 2 \times 3 \end{aligned}$$

Substitution Method: Example 1

By repeating the process, we can observe that:

$$t_n = t_{n-i} + i \times 3$$

When $i = n - 1$, the resulting equation would be as follows:

$$\begin{aligned} t_n &= t_{n-i} + i \times 3 \\ &= t_{n-(n-1)} + (n-1) \times 3 \\ &= t_1 + 3 \times (n-1) \end{aligned}$$

Since $t_1 = 4$, $t_n = 4 + 3 \times (n-1) = 3n + 1$

Substitution Method: Example 2: Compound Interest

Problem: Find the compound interest for the principal amount \$100 if the interest given by a bank is 3%. Formulate the recurrence equation and solve for the principal amount after the 50th month.

Solution: The principal for the current year depends on the principal from the previous year with 3% interest. The recurrence equation is:

$$t_n = t_{n-1} + 0.03 \cdot t_{n-1} = 1.03 \cdot t_{n-1}$$

Let $t_0 = 100$. The compound interest for the first few months is:

$$\begin{aligned} t_1 &= 1.03 \cdot t_0 \\ t_2 &= 1.03 \cdot t_1 = (1.03)^2 t_0 \\ &\vdots \\ t_n &= (1.03)^n \cdot t_0 \end{aligned}$$

For the 50th month, the solution is: $t_{50} = (1.03)^{50} t_0$

Substitution Method: Example 3

Solve the following recurrence equation:

$$t_n = n \cdot t_{n-1} \quad \text{for } n > 1 \quad t_0 = 1$$

Solution: Using the backward substitution method:

$$\begin{aligned} t_n &= nt_{n-1} \\ &= n(n-1)t_{n-2} \\ &= n(n-1)(n-2)t_{n-3} \\ &\vdots \\ &= n(n-1)(n-2)\ldots(1) \end{aligned}$$

Substitution Method: Example 3

At the ith step, the recurrence becomes:

$$t_n = n(n-1)(n-2)\dots(n-i)$$

When $n = i$, this simplifies to:

$$\begin{aligned} t_n &= n(n-1)(n-2)\dots(n-i) \\ &= n(n-1)(n-2)\dots(n-(n-1)) \\ &= n(n-1)(n-2)\dots 1 \\ &= n! \end{aligned}$$

Substitution Method: Example 4: Solving a Geometric Recurrence Equation

Solve the following recurrence equation:

$$t_n = 7t_{n-1}, \quad t_0 = 1$$

Solution:

$$\begin{aligned} t_1 &= 7t_0 = 7 \times 1 = 7 \\ t_2 &= 7t_1 = 7 \times 7 = 7^2 \\ t_3 &= 7t_2 = 7 \times 7^2 = 7^3 \\ &\vdots \\ t_n &= 7^n \end{aligned}$$

Thus, the solution to the recurrence is:

$$t_n = 7^n$$

Theorem: Recurrence of the Form $t_n = rt_{n-1}$

Statement: For the recurrence equation:

$$t_n = rt_{n-1} \quad n > 0 \quad t_0 = a$$

The solution is given by:

$$t_n = ar^n$$

Proof:

$$\begin{aligned}
t_n &= rt_{n-1} \\
&= r \times rt_{n-2} = r^2 t_{n-2} \\
&= r^3 t_{n-3} \\
&\vdots \\
t_n &= r^n t_0 = ar^n
\end{aligned}$$

Substitution Method: Example 5: Recurrence Equation for Tower of Hanoi

The recurrence relation for the Tower of Hanoi is:

$$t_n = 2t_{n-1} + 1, \quad \text{with } t_1 = 1$$

Solution: Using backward substitution, we expand the recurrence:

$$\begin{aligned} t_n &= 2t_{n-1} + 1 \\ &= 2\left(2t_{n-2} + 1\right) + 1 \\ &= 2\left(2\left(2t_{n-3} + 1\right) + 1\right) + 1 \\ &\vdots \\ &= 2^k t_{n-k} + 2^k - 1 \end{aligned}$$

When $k = n - 1$, we get:

$$t_n = 2^{n-1} t_1 + 2^{n-1} - 1 = 2^n - 1$$

Therefore, the minimum number of moves is:

$$t_n = 2^n - 1$$

Substitution Method: Example 6

Solve the recurrence relation using forward substitution.

$$t_n = t_{n-1} + 3 \quad \text{with initial condition} \quad t_0 = 4$$

Solution Compute a few terms using the recurrence relation:

$$\begin{aligned}
t_1 &= t_0 + 3 = 4 + 3 \\
t_2 &= t_1 + 3 = (4 + 3) + 3 = 4 + 2 \times 3 \\
t_3 &= t_2 + 3 = 4 + 2 \times 3 + 3 = 4 + 3 \times 3 \\
&\vdots \\
t_n &= 4 + n \times 3
\end{aligned}$$

The closed form of the recurrence is:

$$t_n = 3n + 4$$

Recurrence Tree Method

- The recurrence-tree method is another way of solving a recurrence equation.
- It is almost similar to the substitution method but used for obtaining the asymptotic bounds.

Recurrence Tree Method

How to solve: Steps:

1. Formulate the recurrence equation by visualizing the calls as a tree.
2. Collect the following information from the recurrence tree:
   (a) Level: Determine the level of the generated tree.
      - The level of a node is the length of the path from the root to the node.
      - The level of the root is 0.
      - The level of a tree is the length of the longest span from the root node to the leaf of a tree.
      - A leaf is a node that has no children.

   (b) Cost per level: The cost at every level has to be calculated. Use the level count and the amount of work done by the sub problems.
3. Express the complexity in terms of the total cost:
   (a) The total cost is the sum of the costs of all levels.
4. Verify the summation using the guess-and-verify method or another method if necessary.

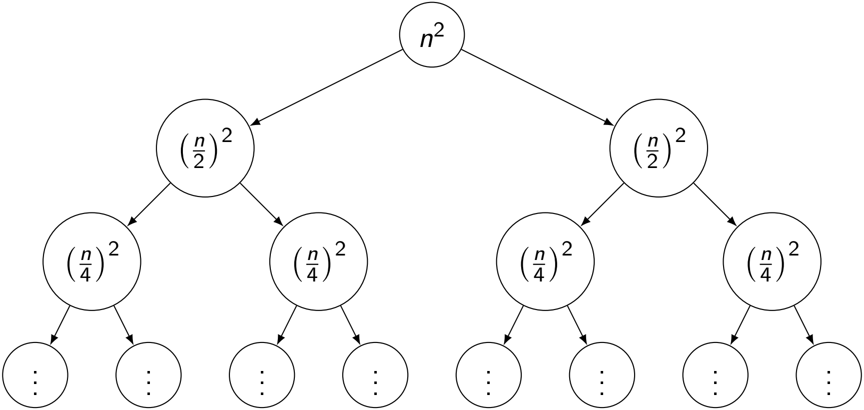
Recurrence Tree Method: Example 1

Solve using the recursion tree method:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + n^2 & \text{if } n > 1 \end{cases}$$

Goal: Expand the recurrence into a tree and determine the total cost by summing up the work done at each level.

Recurrence Tree Method: Example 1



Recurrence Tree Method: Example 1

Recursion Tree Structure Let's visualize the recurrence as a tree:
Level 0 (root):

$$T(n) = n^2$$

Level 1: Each subproblem is of size $n/2$, and there are 2 subproblems:

$$2T(n/2) = 2 \times \left(\frac{n}{2}\right)^2 = \frac{n^2}{2}$$

Level 2: Each subproblem is of size $n/4$, and there are 4 subproblems:

$$4T(n/4) = 4 \times \left(\frac{n}{4}\right)^2 = \frac{n^2}{4}$$

General Level i: At level i, there are 2^i subproblems, each of size $n/2^i$:

$$2^i \times \left(\frac{n}{2^i}\right)^2 = \frac{n^2}{2^i}$$

Recurrence Tree Method: Example 1

To find the total cost, we need to sum the work done at each level.

Cost at each level:

$$\text{Level 0: } n^2$$

$$\text{Level 1: } \frac{n^2}{2}$$

$$\text{Level 2: } \frac{n^2}{4}$$

$$\vdots$$

$$\text{Level } i: \frac{n^2}{2^i}$$

Total cost: Sum the geometric series:

$$T(n) = n^2 \left(1 + \frac{1}{2} + \frac{1}{4} + \dots \right) = n^2 \times \left(\frac{1}{1 - \frac{1}{2}} \right) = 2n^2$$

Recurrence Tree Method: Example 1

Height of the Recursion Tree. The height of the recursion tree is determined by the number of times we can divide n by 2 until we reach 1.

$$\text{Height of the tree} = \log_2 n$$

The total cost at the leaves is:

$$2^{\log_2 n} \cdot T(1) = n \cdot T(1) = O(n)$$

However, the cost at each level is dominated by the cost at the root $O(n^2)$.
Conclusion: The total complexity is:

$$T(n) = O(n^2)$$

Recurrence Tree Method: Example 2

We are given the recurrence:

$$T(n) = 3T(n/2) + n$$

Objective:

- Solve the recurrence using the Recurrence Tree Method.
- Find the time complexity of $T(n)$.

Recurrence Tree Method: Example 2

Steps in the Recurrence Tree:

- The problem size is reduced by half at each level.
- At each level, the number of subproblems triples.
- The additional work at each level is proportional to n.

Root Level (Level 0):

$$T(n) = n$$

Next Levels:

- At level 1: 3 subproblems of size $n/2$ each.
- At level 2: 9 subproblems of size $n/4$ each.

Recurrence Tree Method: Example 2

Visualizing the Recurrence Tree (Level 0, 1, and 2)

Level 0 (Root):

$$T(n) = n$$

Level 1:

$$3T(n/2) = 3 \times \frac{n}{2} = \frac{3n}{2}$$

Total cost at Level 1: $\frac{3n}{2}$

Level 2:

$$3^2T(n/4) = 9 \times \frac{n}{4} = \frac{9n}{4}$$

Total cost at Level 2: $\frac{9n}{4}$

General Pattern:

- At level i, the problem size is $\frac{n}{2^i}$ with 3^i subproblems.
- The total cost at level i is:

$$3^i \times \frac{n}{2^i} = \left(\frac{3}{2}\right)^i n$$

Cost at Each Level

At Level i:

- The number of subproblems at level i is 3^i.
- The size of each subproblem at level i is $n/2^i$.
- The cost at level i is:

$$3^i \times \frac{n}{2^i} = \left(\frac{3}{2}\right)^i n$$

Recurrence Tree Method: Example 2

Total Cost: The total cost of the recursion tree is the sum of the costs at each level.

$$T(n) = n + \frac{3n}{2} + \left(\frac{3}{2}\right)^2 n + \cdots + \left(\frac{3}{2}\right)^{\log_2 n} n$$

This forms a geometric series, but it stops after $\log_2 n$ levels.
Final Complexity: Since there are $\log_2 n$ levels and each contributes a cost proportional to n, the total complexity is:

$$T(n) = O(n \log n)$$

Telescoping Method

Definition: Telescoping is a technique used to simplify sums or recurrence relations by collapsing intermediate terms, leaving only the first and last terms.

Why is it called telescoping?

- The process is similar to collapsing a telescope: intermediate terms cancel out, leaving only the boundary terms.

Where is it used?

- Particularly useful in solving recursive sequences and summations where terms naturally cancel out.

Telescoping Method: Example 1

Consider the recurrence:

$$t_n = t_{n-1} + c$$

Each term is the previous term plus some constant c.
Let's telescope this recurrence by expanding it step-by-step:

$$\begin{aligned} t_n &= t_{n-1} + c \\ t_{n-1} &= t_{n-2} + c \\ t_{n-2} &= t_{n-3} + c \\ &\vdots \\ t_2 &= t_1 + c \end{aligned}$$

Telescoping Method: Example 1

Summing the Equations:

Now, let's add all the expanded terms:

$$t_n = t_1 + c + c + \cdots + c \quad \text{(added } n - 1 \text{ times)}$$

Simplifying the Sum:

$$t_n = t_1 + (n - 1)c$$

Telescoping Method: Example 2: Factorial

Now consider the recurrence:

$$t_n = nt_{n-1}$$

Here, each term depends on the previous term multiplied by n. Let's telescope this recurrence by expanding it step-by-step:

$$\begin{aligned} t_n &= nt_{n-1} \\ t_{n-1} &= (n-1)t_{n-2} \\ t_{n-2} &= (n-2)t_{n-3} \\ &\vdots \\ t_2 &= 2t_1 \end{aligned}$$

Telescoping Method: Example 3

We are given the recurrence:

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

Objective:

- Solve the recurrence using the telescoping method.
- Find the time complexity of $T(n)$.

Initial Condition: Assume $T(1) = d$ (where d is a constant).

Telescoping Method: Example 3

Let's expand the recurrence by substituting it step by step:

$$T(n) = T(n/2) + c$$
$$T(n/2) = T(n/4) + c$$
$$T(n/4) = T(n/8) + c$$

General Pattern:

$$T(n/2^i) = T(n/2^{i+1}) + c$$

We can continue expanding until the problem size becomes 1.

Telescoping Method: Example 3

After expanding the recurrence, we can sum the equations:

$$T(n) = T(n/2) + c$$
$$T(n/2) = T(n/4) + c$$
$$T(n/4) = T(n/8) + c$$

Adding up all these expansions, we get:

$$T(n) = T(1) + c \times (1 + 1 + \cdots + 1) \text{ (added } \log_2 n \text{ times)}$$

Telescoping Method: Example 3

The number of c terms is $\log_2 n$ because each time the problem size is halved, we add another c.
Summing the Series:

$$T(n) = T(1) + c \times \log_2 n$$

Substitute $T(1) = d$**:**

$$T(n) = d + c \times \log_2 n$$

Therefore, the recurrence grows logarithmically.

Telescoping Method: Example 3

Conclusion: The final time complexity of the recurrence $T(n) = T(n/2) + c$ is:

$$T(n) = O(\log_2 n)$$

Summary:

- The recurrence $T(n) = T(n/2) + c$ expands by halving the problem size at each step.
- The number of steps required to reach $T(1)$ is $\log_2 n$.
- Hence, the total complexity is $O(\log_2 n)$.