CE100 Algorithms and Programming II

Solving Recurrences

Author: Asst. Prof. Dr. Uğur CORUH

## CE100 Algorithms and Programming II

## Week-2 (Solving Recurrences)

#### Spring Semester, 2021-2022

Download [DOC](ce100-week-2-recurrence.md_doc.pdf), [SLIDE](ce100-week-2-recurrence.md_slide.pdf), [PPTX](ce100-week-2-recurrence.md_slide.pptx)

## Solving Recurrences

## Outline

* Solving Recurrences
  + Recursion Tree
  + Master Method
  + Back-Substitution
* Divide-and-Conquer Analysis
  + Merge Sort
  + Binary Search
  + Merge Sort Analysis
  + Complexity
* Recurrence Solution

### Solving Recurrences

* Reminder: Runtime of *MergeSort* was expressed as a recurrence
* Solving recurrences is like solving differential equations, integrals, etc.
  + Need to learn a few tricks

### Recurrences

Recurrence: An equation or inequality that describes a function in terms of its value on smaller inputs.

Example :

### Recurrence Example

* Simplification: Assume
* Claimed answer :
* Substitute claimed answer in the recurrence:
* True when

### Technicalities: Floor / Ceiling

Technically, should be careful about the floor and ceiling functions (as in the book).

e.g. For merge sort, the recurrence should in fact be:,

But, it’s usually ok to:

* ignore floor/ceiling
* solve for the exact power of 2 (or another number)

### Technicalities: Boundary Conditions

* Usually assume: for sufficiently small
  + Changes the exact solution, but usually the asymptotic solution is not affected (e.g. if polynomially bounded)
* For convenience, the boundary conditions generally implicitly stated in a recurrence
  + assuming that
  + for sufficiently small

### Example: When Boundary Conditions Matter

Exponential function: Assume  
 e.g.

$$
\begin{rcases}
T(1)= 2 &\Rightarrow & T(n)= \Theta(2^n) \\
T(1)= 3 &\Rightarrow & T(n)= \Theta(3^n)
\end{rcases}
\text{ However } \Theta(2^n) \neq \Theta(3^n)
$$

The difference in solution more dramatic when:

### Solving Recurrences

We will focus on 3 techniques

* Substitution method
* Recursion tree approach
* Master method

### Substitution Method

The most general method:

* Guess
* Prove by induction
* Solve for constants

### Substitution Method: Example

Solve (assume )

1. Guess (need to prove and separately)
2. Prove by induction that for large (i.e. )
   * Inductive hypothesis: for any
   * Assuming ind. hyp. holds, prove

### Substitution Method: Example – cont’d

Original recurrence:

From inductive hypothesis:

Substitute this into the original recurrence:

* desired - residual
* when

### Substitution Method: Example – cont’d

So far, we have shown:

We can choose and

But, the proof is not complete yet.

**Reminder: Proof by induction:** *1.Prove the base cases* haven’t proved the base cases yet *2.Inductive hypothesis for smaller sizes* *3.Prove the general case*

### Substitution Method: Example – cont’d

* We need to prove the base cases
  + Base: for small (e.g. for )
* We should show that:
  + for , This holds if we pick big enough
* So, the proof of is complete
* But, is this a tight bound?

### Example: A tighter upper bound?

* Original recurrence:
* Try to prove that ,
  + i.e.  for all
* **Ind. hyp:** Assume that for
* **Prove the general case:**

### Example (cont’d)

Original recurrence: Ind. hyp: Assume that for Prove the general case:

$$
T(n) = 4T(n/2) + n \\
≤ 4c(n/2)^2 + n \\
= cn^2 + n \\
= O(n2) \Longleftarrow \text{ Wrong! We must prove exactly}
$$

### Example (cont’d)

**Original recurrence:** **Ind. hyp:** Assume that for **Prove the general case:**

* So far, we have:
* No matter which positive c value we choose, this does not show that
* Proof failed?

### Example (cont’d)

* What was the problem?
  + The inductive hypothesis was not strong enough
* **Idea:** Start with a stronger inductive hypothesis
  + Subtract a low-order term
* **Inductive hypothesis:** for
* **Prove the general case:**

### Example (cont’d)

**Original recurrence:**

**Ind. hyp:** Assume that for

Prove the general case: for choose

### Example (cont’d)

We now need to prove

for the base cases.

(implicit assumption)

for small enough (e.g. )

* We can choose c1 large enough to make this hold

We have proved that

### Substitution Method: Example 2

For the recurrence ,

prove that

i.e.  for any

**Ind. hyp:** for any

**Prove general case:**

since

Proof succeeded – no need to strengthen the ind. hyp as in the last example

### Example 2 (cont’d)

We now need to prove that for the base cases for (implicit assumption) for

is sufficiently small (i.e. constant)

We can choose small enough for this to hold

We have proved that

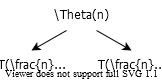
### Substitution Method - Summary

* Guess the asymptotic complexity
* Prove your guess using induction
  + Assume inductive hypothesis holds for
  + Try to prove the general case for
    - Note: prove the inequality ignore lower order terms, If the proof fails, strengthen the ind. hyp. and try again
* Prove the base cases (usually straightforward)

### Recursion Tree Method

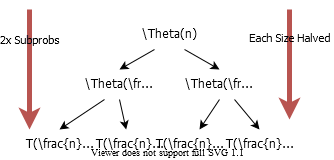
* A recursion tree models the runtime costs of a recursive execution of an algorithm.
* The recursion tree method is good for generating guesses for the substitution method.
* The recursion-tree method can be unreliable.
  + Not suitable for formal proofs
* The recursion-tree method promotes intuition, however.

### Solve Recurrence :



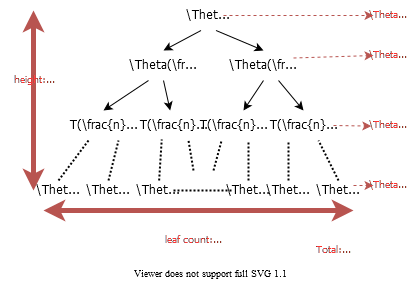
alt:“alt” height:250px center

### Solve Recurrence :



alt:“alt” height:450px center

### Solve Recurrence :



alt:“alt” height:500px center

| ### Example of Recursion Tree |
| --- |
| ### TODO |
| Continue From Slide : 03 Page : 27 |

## References

TODO