CSCI 3100: ALGORITHMS

Chapter 4

Divide-and-Conquer

Recurrences

REVIEW & OVERVIEW

- Last time
  - Lower bound on comparison sorts
  - Formal proofs
  - First homework
- This time
  - Divide and conquer
  - Recurrences and recursion

DIVIDE AND CONQUER ANALOGY

RECURRENCES AND RUNNING TIME

- Recurrences arise when an algorithm contains recursive calls to itself
- Running time is represented by an equation or inequality that describes a function in terms of its value on smaller inputs.

$$T(n) = T(n-1) + n$$

- What is the actual running time of the algorithm? i.e. $T(n) = ?$
- Need to solve the recurrence
  - Find an explicit formula of the expression
  - Bound the recurrence by an expression that involves n

EXAMPLE RECURRENCES

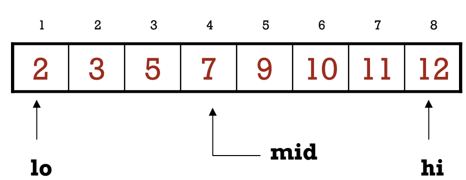
- $T(n) = T(n-1) + n$ $\Theta(n^2)$
  Recursive algorithm that loops through the input to eliminate one item
- $T(n) = T(n/2) + c$ $\Theta(\lg n)$
  Recursive algorithm that halves the input in one step
- $T(n) = T(n/2) + n$ $\Theta(n)$
  Recursive algorithm that halves the input but must examine every item in the input
- $T(n) = 2T(n/2) + 1$ $\Theta(n)$
  Recursive algorithm that splits the input into 2 halves and does a constant amount of other work

BINARY-SEARCH

- Finds if x is in the **sorted** array A[lo…hi]

Alg.: BINARY-SEARCH (A, lo, hi, x)

if (lo > hi)
    return FALSE
mid ← $\lfloor$(lo+hi)/2$\rfloor$
if x = A[mid]
    return TRUE
if (x < A[mid])
    BINARY-SEARCH (A, lo, mid-1, x)
if (x > A[mid])
    BINARY-SEARCH (A, mid+1, hi, x)



EXAMPLE 1

A[8] = {1, 2, 3, 4, 5, 7, 9, 11}

lo = 1 hi = 8 **x = 7**

1	2	3	4	5	6	7	8
1	2	3	(4)	5	7	9	11

mid = 4, lo = 5, hi = 8

				5	6	7	8
1	2	3	4	5	(7)	9	11

mid = 6, A[mid] = x
Found!

EXAMPLE 1I

A[8] = {1, 2, 3, 4, 5, 7, 9, 11}

lo = 1 hi = 8 **x = 6**

1	2	3	4	5	6	7	8
1	2	3	(4)	5	7	9	11
↑ low							↑ high

lo = 1, hi = 8, mid = 4.
A[4]=4

1	2	3	4	5	(7)	9	11
				↑ low			↑ high

lo = 5, hi = 8, mid = 6,
A[6]=7

1	2	3	4	(5)	7	9	11
				↑↑			

lo = 5, hi = 5, mid = 5,
A[5]=5

1	2	3	4	5	7	9	11
				high ↑	↑ low		

lo = 6, hi = 5 → **NOT FOUND!**

ANALYSIS OF BINARY-SEARCH

Alg.: BINARY-SEARCH (A, lo, hi, x)

```
if (lo > hi)
    return FALSE                  ← constant time: c1
mid ← ⌊(lo+hi)/2⌋                 ← constant time: c2
if x = A[mid]
    return TRUE                   ← constant time: c3
if ( x < A[mid] )
    BINARY-SEARCH (A, lo, mid-1, x)   ← same problem of size n/2
if ( x > A[mid] )
    BINARY-SEARCH (A, mid+1, hi, x)   ← same problem of size n/2
```

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

METHODS FOR SOLVING RECURRENCES

- **Substitution** method
- **Iteration/Recursion-tree** method
- **Master** method

10

SUBSTITUTION METHOD

- Make a guess
- Prove that your guess is correct

<u>Example</u>: $T(n) = c + T(n/2)$

Guess: $T(n) = O(n)$

Proof:

- Base case: $n = 1$, $T(n)$ = constant = $O(n)$
- Inductive step: assume true for $n/2$:
  - $T(n/2) \leq cn/2$ for all $n \geq n_0$
  - Complete the proof …

SUBSTITUTION METHOD – EXAMPLE 2

- $T(n) = n + 2T(n/2) = O(n\lg(n))$

THE ITERATION METHOD

Convert the recurrence into a summation and solve it using a known series

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

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

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

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

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

Assume $n=2^k$ **then** $k = \lg n$ **and**

$T(n) = \underbrace{c + c + c + c + c + \ldots +}_{(k \text{ times})} T(n/2^k)$

$T(n) = k * c + T(1)$

$T(n) = c\lg n$

ITERATION METHOD – EXAMPLE 2

$T(n) = n + 2T(n/2)$ **Assume** $n=2^k \Rightarrow k = \lg n$

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

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

$= n + n + 4T(n/4)$

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

$= n + n + n + 8T(n/8)$

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

$= kn + 2^kT(n/2^k)$

$= n\lg n + nT(1)$

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

METHODS FOR SOLVING RECURRENCES

- **Substitution** method
- **Iteration/Recursion-tree** method
- **Master** method

15

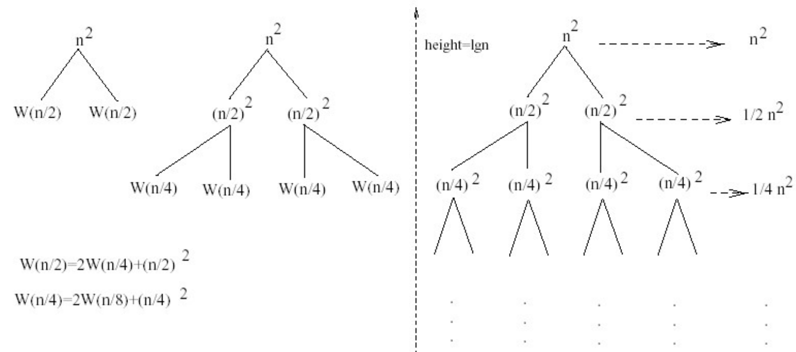
THE RECURSION-TREE METHOD

Convert the recurrence into a tree:

- Each node represents the cost incurred at various levels of recursion
- Sum up the costs of all levels

Used to "guess" a solution for the recurrence

EXAMPLE 1 $W(N) = 2W(N/2) + N^2$

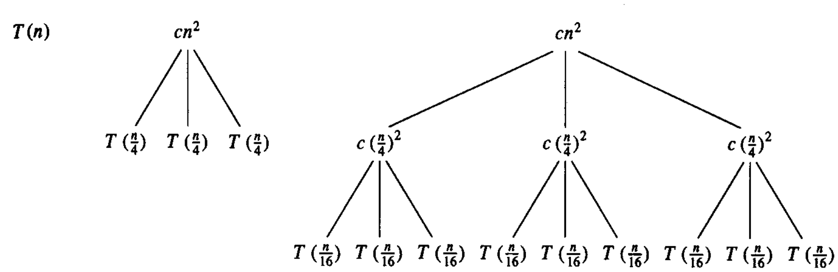


- Subproblem size at level i = $n/2^i$
- **At level i:** Cost of each node = $(n/2^i)^2$ # of nodes = 2^i Total cost = $(n^2/2^i)$
- h = Height of the tree ➔ $n/2^h=1$ ➔ **h = lgn**
- Total cost at all levels:

$$W(n) = \sum_{i=0}^{\lg n} \frac{n^2}{2^i} = n^2 \sum_{i=0}^{\lg n} \left(\frac{1}{2}\right)^i \le n^2 \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = n^2 \frac{1}{1-\frac{1}{2}} = 2n^2$$

➔ **W(n) = $O(n^2)$**

EXAMPLE 2 $T(N) = 3T(N/4) + CN^2$



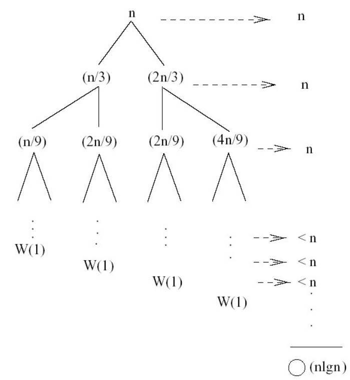
- Subproblem size at level i = $n/4^i$
- **At level i:** Cost of each node= $c(n/4^i)^2$ # of nodes= 3^i Total cost =$cn^2(3/16)^i$
- h = Height of the tree ➔ $n/4^h=1$ ➔ **h = $\log_4 n$**
- Total cost at all levels: (last level has $3^{\log_4 n} = n^{\log_4 3}$ nodes)

$$T(n) = \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) \le \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) = \frac{1}{1-\frac{3}{16}} cn^2 + \Theta\left(n^{\log_4 3}\right) = O(n^2)$$

➔ **T(n) = $O(n^2)$**

EXAMPLE 3

W(n) = W(n/3) + W(2n/3) + n



19