Algorithms and Problem Solving (15B11CI411)

EVEN 2022

Module 1: Lecture 3

Jaypee Institute of Information Technology (JIIT)

A-10, Sector 62, Noida

Recurrences and Running Time

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

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

- Recurrences arise when an algorithm contains recursive calls to itself
- What is the actual running time of the algorithm?
- Need to solve the recurrence

Example Recurrences

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

Analysis of BINARY-SEARCH

Alg.: BINARY-SEARCH (A, first, last, x)

if (first > last) ← constant time: c_1

return FALSE

mid ← $\lfloor$(first+last)/2$\rfloor$ ← constant time: c_2

if x = A[mid] ← constant time: c_3

return **TRUE**

if (x < A[mid])

BINARY-SEARCH (A, first, mid-1, x) ← same problem of size n/2

if (x > A[mid])

BINARY-SEARCH (A, mid+1, last, x) ← same problem of size n/2

- $T(n) = c + T(n/2)$
  - T(n) – running time for an array of size n

How to solve Recurrence Relations

- Substitution method
- Iteration method
- Recursion Tree method
- Master method

Substitution Method

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

Substitution method

- Guess a solution
  - $T(n) = O(g(n))$
  - Induction goal: apply the definition of the asymptotic notation
    - $T(n) \leq c\, g(n)$, for some $c > 0$ and $n \geq n_0$
  - Induction hypothesis: $T(k) \leq c\, g(k)$ for all $k < n$
- Prove the induction goal
  - Use the **induction hypothesis** to find some values of the constants c and n_0 for which the **induction goal** holds

Example 1: $\boldsymbol{T(n) = T(n-1) + n}$

- Guess: $T(n) = O(n^2)$
  - Induction goal: $T(n) \le c\,n^2, for\ some\ c\ and\ n \ge n_0$
  - Induction hypothesis: $T(n-1) \le c(n-1)^2\ for\ all\ k < n$
- Proof of induction goal:

$$
\begin{aligned}
T(n) &= T(n-1) + n \\
&\le c\,(n-1)^2 + n \\
&= cn^2 - (2cn - c - n) \\
&\le cn^2\ if\!:\ 2cn - c - n \ge 0 \Rightarrow c \ge n/(2n-1) \\
&\qquad \Rightarrow c \ge 1/(2 - 1/n) \\
&\qquad\qquad \Rightarrow n_0 = 1\ and\ c \ge 1
\end{aligned}
$$

Example 2- Show that $T(n)=2T(\frac{n}{2})+n$ is $O(n\log n)$ by substitution method

Proof: To prove $T(n) = O(n\log n)$, we need to prove that $T(n) <= cn\log n$.

Assume that it is true for values smaller than n.

Induction Step: Assume it is true for $n=\frac{n}{2}$

i.e., $T(\frac{n}{2}) <= 2.c.\ (\frac{n}{2}) \log (\frac{n}{2})$ is true

Now we have to show that it is true for $n=n$

i.e. $T(n) \leq cnlogn$

$$T(n) \leq 2.T\left(\frac{n}{2}\right)+n$$

$$\leq 2(c.\left(\frac{n}{2}\right).\log\left(\frac{n}{2}\right))+n$$

$$\leq cn\log\left(\frac{n}{2}\right)+n \leq cn\log n - cn\log 2 + n$$

$$\leq cn\log n - cn + n \leq cn\log n\ \forall c \geq 1$$

Thus $T(n) = O(nlogn)$

Example 2: Binary Search

$\boldsymbol{T(n) = c + T(n/2)}$

- $Guess\text{: } T(n) = O(logn)$
  - Induction goal: $T(n) \leq d\ logn, for\ some\ d\ and\ n \geq n_0$
  - Induction hypothesis: $T(n/2) \leq d \log(n/2)$
- Proof of induction goal:

$$
\begin{aligned}
T(n) &= T(n/2) + c \\
&\leq d \log(n/2) + c \\
&= d\ logn - d + c = d \log n - (d - c) \\
&\leq d\ logn \quad if\text{: } \ d - c \geq 0, d \geq c
\end{aligned}
$$

- $What\ about\ n_0$

The Iteration Method

Steps followed to solve any recurrence using iterating methods are:

- Expend the recurrence
- Express the expansion as a summation by plugging the recurrence back into itself until you see a pattern.
- Evaluate the summation by using the arithmetic or geometric summation formulae

Example 1:

$$T(n) = \begin{cases} n + T(n-1), & if n > 1 \\ 1, & if\ n \geq 1 \end{cases}$$

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

$= n + (n-1) + T(n-2)$

$= n + (n-1) + (n-2) + T(n-3)$

$= \ \dots..$

$= n + (n-1) + (n-2)\ + \cdots. + T(1)\,, T(1) = \ 1$, we can write the above equation as

$= 1 + 2 + 3 + \cdots \dots. + (n-1) + n$

$= n(n+1)/2 = (n^2 + n)/2\ \leq\ n^2$

$= O(n^2)$

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

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

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

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

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

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

Assume $n = 2^k$

$$T(n) = c + \underbrace{c + \ldots + c}_{\text{k times}} + T(1)$$

$$= c\log n + T(1)$$

$$= O(\log n)$$

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

Assume: $n = 2^k$

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

$$
\begin{aligned}
T(n) &= n + 2T(n/2) \\
&= n + 2(n/2 + 2T(n/4)) \\
&= n + n + 4T(n/4) \\
&= n + n + 4(n/4 + 2T(n/8)) \\
&= n + n + n + 8T(n/8) \\
\ldots &= kn + 2^k T(n/2^k) \\
&= kn + 2^k T(1) \\
&= nlogn + nT(1) = O(nlgn)
\end{aligned}
$$