Text Book:
Introduction to the Design and Analysis of Algorithms
Author: Anany Levitin
2^{nd} Edition

Recurrence

Recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs

Recurrences can take many forms

Example:

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

Important Recurrence Types

- Decrease-by-one recurrences

A decrease-by-one algorithm solves a problem by exploiting a relationship between a given instance of size *n* and a smaller size *n – 1*.

Example: n!

The recurrence equation has the form

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

- Decrease-by-a-constant-factor recurrences

A decrease-by-a-constant algorithm solves a problem by dividing its given instance of size *n* into several smaller instances of size *n/b*, solving each of them recursively, and then, if necessary, combining the solutions to the smaller instances into a solution to the given instance.

Example: binary search.

The recurrence has the form

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

Methods to solve recurrences

- Substitution Method
  - Mathematical Induction
  - Backward substitution
- Recursion Tree Method
- Master Method (Decrease by constant factor recurrences)

Example1:

$T(n) = T(n-1) + 1 \quad n>0 \quad T(0) = 1$

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

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

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

...

$= T(n-i) + i$

...

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

Example2:

$T(n) = T(n-1) + 2n - 1 \quad T(0)=0$

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

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

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

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

...

$= T(n-i) + 2(n-i+1) + \ldots + 2n - i$

...

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

= 0 + 2 + 4 +...+ 2n - n

= 2 + 4 +...+ 2n - n

*= 2*n*(n+1)/2 - n*

// arithmetic progression formula 1+...+n = n(n+1)/2 //

= O(n²)

Example 3:

$$T(n) = T(n/2) + 1 \quad n > 1$$
$$T(1) = 1$$

$$
\begin{aligned}
T(n) &= T(n/2) + 1 \\
&= T(n/2^2) + 1 + 1 \\
&= T(n/2^3) + 1 + 1 + 1 \\
&\ldots\ldots \\
&= T(n/2^i) + i \\
&\ldots\ldots \\
&= T(n/2^k) + k \quad (k = \log n) \\
&= 1 + \log n \\
&= O(\log n)
\end{aligned}
$$

Example 4:

$$T(n) \;\; = 2T(n/2) + cn \qquad n > 1 \qquad T(1) = c$$

$$
\begin{aligned}
T(n) &= 2T(n/2) + cn \\
&= 2(2T(n/2^2) + c(n/2)) + cn = 2^2 T(n/2^2) + cn + cn \\
&= 2^2(2T(n/2^3) + c(n/2^2)) + cn + cn = 2^3 T(n/2^3) + 3cn \\
&\ldots\ldots \\
&= 2^i T(n/2^i) + icn \\
&\ldots\ldots \\
&= 2^k T(n/2^k) + kcn \quad (k = \log n) \\
&= nT(1) + cn\log n = cn + cn\log n \\
&= O(n\log n)
\end{aligned}
$$

Example 5:

$T(n)=2T(\sqrt{n})+1$ $T(1)=1$

Assume $n=2^m$

Which gives recurrence

$T(2^m)=2T(2^{m/2})+1$

Assume $T(2^m)=S(m)$

Which gives recurrence

$S(m)=2S(m/2)+1$

Solving using backward substitution (reference example3) gives

$S(m)=m+2$

- ⇨ $T(n)=O(\log n)$