Recurrence Relations

Recurrence Relations

- Recurrence relations are used to determine the running time of recursive programs.
- Recurrence relations themselves are recursive.
- $T(0)$: time to solve problem of size 0. This is the base Case
- $T(\mathrm{n})$: time to solve problem of size n. This is the recursive Case

Recurrence Relation for Linear Search

- Study code
- $T(0) = 0$
- $T(n) = 1 + T(n - 1)$

Solving a Recurrence Relation through Telescoping

- $T(n) = 1 + T(n - 1)$
  $T(n - 1) = 1 + T(n - 2)$
  $T(n - 2) = 1 + T(n - 3)$
  …
  $T(0) = 0$

  $T(n) = 1 + 1 + \ldots + 1$ (n times)
  $T(n) = n$
  $T(n) = O(n)$

Recurrence Relation for Binary Search

- Study code
- $T(0) = 0$
- $T(n) = 2 + T(n / 2)$

Solving a Recurrence Relation through Telescoping

- $T(n) = 2 + T(n / 2)$
  $T(n / 2) = 2 + T(n / 4)$
  $T(n / 4) = 2 + T(n / 8)$
  …
  $T(0) = 0$

  $T(n) = 2 + 2 + \ldots + 2$ (log n times)
  $T(n) = 2*\log(n)$
  $T(n) = O(\log(n))$

Recurrence Relation for Merge Sort **Best** Case

- Study code
- $T(n) = n/2 + 2T(n / 2)$

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

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

  …

  $T(0) = 0$

  $T(n) = n/2 + 2(n/4) + 4(n/8) +$… ((log n) terms)

  $T(n) = n/2 + n/2 + n/2 +$… ((log n) terms)

  $T(n) = n/2*\log(n) = O(n \log(n))$

Recurrence Relation for Merge Sort **Worst** Case

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

  $T(n / 2) = (n/2 -1) + 2T(n/4)$

  $T(n / 4) = (n/4 -1) + 2T(n/8)$

  …

  $T(0) = 0$

  $T(n) = (n-1) + 2(n/2-1) + 4(n/4-1) +$ …

  $= (n-1) + (n-2) + (n-4) +$ …

  $= \log(n)*n + (-1 - 2 - 4 ...)$ [sum of powers of 2, from 0 to log(n)-1]

  $= n*\log(n) – (n - 1)$ [$2^{(\log(n)-1)+1}-1 = n-1$]

  $= (n-1)*\log(n) + 1$

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