3. Recurrences

- A recurrence is an equation defining a function f(n) recursively in terms of smaller values of n.

- E.g., the running time of Merge-Sort, if n is a power of 2, is:

  $T(n) = \Theta(1)$ if $n = 1$

  $T(n) = 2\,T(n/2) + \Theta(n)$ if $n > 1$

  For arbitrary n > 0, the running time is

  $T(n) = \Theta(1)$ if $n = 1$

  $T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n)$ if $n > 1$

  Why?

- We use 3 methods for solving recurrences
  - Substitution Method
  - Iteration Method
  - Master Method

Floors and Ceilings

- For any real number x,

  $\lfloor x \rfloor$ = greatest integer less than or equal to x

  $\lceil x \rceil$ = least integer greater than or equal to x

- For any integer n,

  $\lceil n / 2 \rceil + \lfloor n / 2 \rfloor = n$

- For integers a ≠ 0 and b ≠ 0,

  $\lceil \lceil n / a \rceil / b \rceil = \lceil n / (a\,b) \rceil$

  $\lfloor \lfloor n / a \rfloor / b \rfloor = \lfloor n / (a\,b) \rfloor$

Logarithms

- Definition: For any a, b , c:

  $\log_b a = c \quad \Leftrightarrow \quad b^c = a$

- We use:

  $\lg n = \log_2 a$ (binary logarithm)

  $\ln n = \log_e a$ (natural logarithm)

- Properties (writing log for a logarithm with arbitrary base):

  $a = b^{\log_b a}$

  $\log (a\, b) = \log a + \log b$

  $\log a^n = n \log a$

  $\log_b a = (\log_c a) / (\log_c b)$ (*)

  $\log (1/a) = - \log a$

  $\log_b a = 1/\log_a b$

  $a^{\log_b n} = n^{\log_b a}$

- (*) implies that e.g. $\Theta(\lg n) = \Theta(\log_c n)$ for any c.

  The base of the logarithm is irrelevant for asymptotic analysis!

Forward Substitution Method ...

Ü Guess a solution.

Ü Verify by induction.

- For example, for

  $T(n) = 2\, T(\lfloor n / 2 \rfloor) + n$ and $T(1) = 1$

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

- Induction Goal:

  $T(n) \leq c\, n \lg n$, for some c and all $n > n_0$

- Induction Hypothesis:

  $T(\lfloor n / 2 \rfloor) \leq c \lfloor n / 2 \rfloor \lg \lfloor n / 2 \rfloor$

- Proof of Induction Goal:

  $$
  \begin{aligned}
  T(n) &= 2\, T(\lfloor n / 2 \rfloor) + n \\
  &\leq 2\, (c \lfloor n / 2 \rfloor \lg \lfloor n / 2 \rfloor) + n \\
  &\leq c\, n \lg (n / 2) + n \\
  &= c\, n \lg n - c\, n \lg 2 + n \\
  &= c\, n \lg n - c\, n + n \\
  &\leq c\, n \lg n \qquad \text{provided } c \geq 1
  \end{aligned}
  $$

… Forward Substitution Method

- So far the restrictions on c, n_0 are only $c \geq 1$
- Base Case:

  $T(n_0) \leq c\ n \lg n$

  Here, $n_0 = 1$ does not work, since $T(1) = 1$ but $c\ 1 \lg 1 = 0$.

  However, taking $n_0 = 2$ we have:

  $T(2) = 4 \qquad 2 \lg 2 = 2$

  so

  $T(2) \leq c\ 2$

  holds provided $c \geq 2$.

48

Summations…

- Linearity:

  $$(\sum k \mid 1 \leq k \leq n \bullet c a_k + b_k)$$
  $$= c\,(\sum k \mid 1 \leq k \leq n \bullet a_k) + (\sum k \mid 1 \leq k \leq n \bullet b_k)$$

  Use for asymptotic notation:

  $$(\sum k \mid 1 \leq k \leq n \bullet \Theta(f(k))) = \Theta(\sum k \mid 1 \leq k \leq n \bullet f(k))$$

  In this equation, the Θ-notation on the left hand side applies to variable k, but on the right-hand side, it applies to n.

- Arithmetic Series:

  $$(\sum k \mid 1 \leq k \leq n \bullet k) = n\,(n + 1) / 2 = \Theta(n^2)$$

- Geometric (or Exponential) Series: If $x \neq 1$ then

  $$(\sum k \mid 0 \leq k \leq n \bullet x^k) = (x^{n+1} - 1) / (x - 1)$$

49

… Summations

- Infinite Decreasing Geometric Series: If $|x| < 1$ then

$$(\sum k \mid 0 \leq k < \infty \bullet x^k) = 1 / (1 - x)$$

- Harmonic Series:

$$\begin{aligned} H_n &= 1 + 1 / 2 + 1 / 3 + \ldots + 1 / n \\ &= (\sum k \mid 1 \leq k \leq n \bullet 1 / k) \\ &= \ln n + O(1) \end{aligned}$$

- Further series obtained by integrating or differentiating the formulas above.

  For example, by differentiating the infinite decreasing geometric series and multiplying with x we get:

$$(\sum k \mid 0 \leq k < \infty \bullet k\, x^k) = x / (1 - x)^2$$

50

Iteration (Backward Substitution) Method …

Ü Express the recurrence as a summation of terms.

Ü Use techniques for summations.

- For example, we iterate

$$T(n) = 3\, T(\lfloor n / 4 \rfloor) + n$$

  as follows:

$$\begin{aligned} T(n) &= n + 3\, T(\lfloor n / 4 \rfloor) \\ &= n + 3\, (\lfloor n / 4 \rfloor + 3\, T(\lfloor n / 16 \rfloor)) \\ &= n + 3\, (\lfloor n / 4 \rfloor + 3\, (\lfloor n / 16 \rfloor + 3\, T(\lfloor n / 64 \rfloor))) \\ &= n + 3 \lfloor n / 4 \rfloor + 9 \lfloor n / 16 \rfloor + 27\, T(\lfloor n / 64 \rfloor) \end{aligned}$$

- The i-th term in the series is $3^i \lfloor n / 4^i \rfloor$

  We have to iterate until $\lfloor n / 4^i \rfloor = 1$, since $T(1) = \Theta(1)$, or equivalently until $i > \log_4 n$.

51

… Iteration (Backward Substitution) Method

- We continue:

$$
\begin{aligned}
T(n) &= n + 3 \lfloor n / 4 \rfloor + 9 \lfloor n / 16 \rfloor + 27\, T(\lfloor n / 64 \rfloor) \\
&\le n + 3\, n / 4 + 9\, n / 16 + 27\, n / 64 + \ldots + 3^{\log_4 n}\, \Theta(1) \\
&\quad \{\text{as } a^{\log_b n} = n^{\log_b a}\} \\
&\le n\, (\textstyle\sum i \mid 0 \le i < \infty \cdot (3 / 4)^i\,) + \Theta(n^{\log_4 3}) \\
&\quad \{\text{decreasing geometric series:} \\
&\quad (\textstyle\sum k \mid 0 \le k < \infty \cdot x^k) \;=\; 1 / (1 - x)\} \\
&\le 4\, n + \Theta(n^{\log_4 3}) \\
&\quad \{\log_4 3 < 1\} \\
&= 4\, n + o\,(n) \\
&= O\,(n)
\end{aligned}
$$

The Master Theorem

- Let $a \ge 1$ and $b > 1$ be constants and $f(n)$ be a function. Assume

  $T(n) = a\, T(n/b) + f(n)$

  where n/b stands for $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then
  - $T(n) = \Theta(n^{\log_b a})$ if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$,
  - $T(n) = \Theta(n^{\log_b a} \lg n)$ if $f(n) = \Theta(n^{\log_b a})$
  - $T(n) = \Theta(f(n))$ if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$ and if $a\, f(n/b) \le c\, f(n)$ for some $c < 1$ and sufficiently large n.

- Note 1: This theorem can be applied to divide-and-conquer algorithms, which are all of the form

  $T(n) = a\, T(n/b) + D(n) + C(n)$

  where $D(n)$ is the cost of dividing and $C(n)$ the cost of combining.

- Note 2: Not all possible cases are covered by the theorem.

Merge Sort with the Master Theorem

- For arbitrary $n > 0$, the running time of Merge-Sort is

  $T(n) = \Theta(1)$ if $n = 1$

  $T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n)$ if $n > 1$

  We can approximate this from below and above by

  $T(n) = 2\,T(\lfloor n/2 \rfloor) + \Theta(n)$ if $n > 1$

  $T(n) = 2\,T(\lceil n/2 \rceil) + \Theta(n)$ if $n > 1$

  respectively. According to the Master Theorem, both have the same solution which we get by taking

  $a = 2$, $b = 2$, $f(n) = \Theta(n)$.

  Since $n = n^{\log_2 2}$, the second case applies and we get:

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

Binary Search with the Master Theorem

- The Master Theorem allows us to ignore the floor or ceiling function around n/b in $T(n/b)$ in general.
- Binary Search has for any $n > 0$ a running time of

  $T(n) = T(n/2) + \Theta(1)$.

  Hence $a = 1$, $b = 2$, $f(n) = \Theta(1)$. Since $1 = n^{\log_2 1}$ the second case applies and we get:

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

Towers of Hanoi with the Master Theorem (a bit odd application)

- The Towers of Hanoi algorithm has for any $n > 0$ a running time of

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

  In order to bring this into a form such that the Master Theorem is applicable, we rename $n = \lg m$:

  $$\begin{aligned} T(\lg m) &= 2\ T(\lg m - 1) + 1 \\ &= 2\ T(\lg m - \lg 2) + 1 \\ &= 2\ T(\lg (m/2)) + 1 \end{aligned}$$

  Defining $S(m) = T(\lg m)$ we get the new recurrence:

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

  Hence $a = 2$, $b = 2$, $f(m) = 1$. Since $1 = m^{\log_2 2 - 1}$ the first case applies with $\varepsilon = 1$ and we get:

  $S(m) = \Theta(m)$

  With $S(m) = T(\lg m)$ and $n = \lg m$ we finally get:

  $T(n) = \Theta(2^n)$