Lecture 3: Summations and Recurrences

(CLRS A, 4.1)

May 20th, 2002

1 Review

- Asymptotic growth: O, Ω, Θ
  - We often think of $f(n) = O(g(n))$ as corresponding to $f(n) \leq g(n)$.
  - Similarly, $f(n) = \Theta(g(n))$ corresponds to $f(n) = g(n)$
  - Similarly, $f(n) = \Omega(g(n))$ corresponds to $f(n) \geq g(n)$
  - One can also define o and ω
    - $f(n) = o(g(n))$ corresponds to $f(n) < g(n)$
    - $f(n) = \omega(g(n))$ corresponds to $f(n) > g(n)$
- Growth rate of standard functions:
  - polynomials versus exponentials: $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$, for any $a > 1, b > 0$.
  - polynomials versus polylogarithmics: $\lim_{n\to\infty} \frac{\log^a n}{n^b} = 0$, for any $a, b > 0$.

1.1 Log's

- Base 2 logarithm comes up all the time (from now on we will always mean $\log_2 n$ when we write $\log n$).
  - Number of times we can divide n by 2 to get to 1 or less.
  - Number of bits in binary representation of n.
  - Inverse function of $2^n = 2 \cdot 2 \cdot 2 \cdots 2$ (n times).
  - Way of doing multiplication by addition: $\log(ab) = \log(a) + \log(b)$
- Note:
  - $\log_a n = \frac{\log_b n}{\log_b a}$
  - $\log n << \sqrt{n} << n$

2 Summations

When analyzing insertion-sort we used

$\boxed{\sum_{k=1}^{n} k = 1+2+3+\cdots+n = \frac{n(n+1)}{2} = \Theta(n^2)}$ (*Arithmetic series*)

How can we prove this?

- Asymptotic:

  Often good estimates can be found by using the largest value to bound others:

  $\sum_{k=1}^{n} k \leq \sum_{k=1}^{n} n = n \cdot \sum_{k=1}^{n} 1 = n^2 = O(n^2)$

  Another trick: Splitting the sum:

  $\sum_{k=1}^{n} k = \sum_{k=1}^{n/2-1} k + \sum_{k=\frac{n}{2}}^{n} k \geq \sum_{k=1}^{n/2-1} 0 + \sum_{k=\frac{n}{2}}^{n} k \geq (\frac{n}{2})^2 = \Omega(n^2)$.

  $\Downarrow$

  $\sum_{k=1}^{n} k = \Theta(n^2)$

- Precise (**proof by induction!**):

  - Basis: $n = 1 \Rightarrow \sum_{k=1}^{1} = 1$

    $\frac{n(n+1)}{2} = \frac{1 \cdot 2}{2} = 1$

  - Induction:

    Assume it holds for n: $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$

    Show it holds for $n+1$: $\sum_{k=1}^{n+1} k = \frac{(n+1)(n+2)}{2} = \frac{1}{2}n^2 + \frac{3}{2}n + 1$

    Proof:

$$\begin{aligned}\sum_{k=1}^{n+1} k &= \sum_{k=1}^{n} k + (n+1) \\ &= \frac{n(n+1)}{2} + (n+1) \\ &= \frac{1}{2}n^2 + \frac{1}{2}n + n + 1 \\ &= \frac{1}{2}n^2 + \frac{3}{2}n + 1\end{aligned}$$

In general we can prove that $\boxed{\sum_{k=1}^{n} k^d = \Theta(n^{d+1})}$

Another important sum: $\boxed{\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots x^n = \frac{x^{n+1}-1}{x-1} = O(x^n)}$ (*Geometric series*)

- Proof by induction:
  - Basis: $n = 1 \Rightarrow \quad \sum_{k=0}^{1} x^k = 1 + x$
    $\frac{x^{n+1}-1}{x-1} = \frac{x^2-1}{x-1} = \frac{(x+1)(x-1)}{(x-1)} = x + 1$
  - Induction:
    Assume holds for n: $\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$
    Show it holds for $n+1$: $\sum_{k=0}^{n+1} x^k = \frac{x^{n+2}-1}{x-1}$
    Proof:

$$\begin{aligned}
\sum_{k=0}^{n+1} x^k &= \sum_{k=0}^{n} x^k + x^{n+1} \\
&= \frac{x^{n+1}-1}{x-1} + x^{n+1} \\
&= \frac{x^{n+1}-1+x^{n+1}(x-1)}{x-1} \\
&= \frac{x^{n+1}-1+x^{n+2}-x^{n+1}}{x-1} \\
&= \frac{x^{n+2}-1}{x-1}
\end{aligned}$$

- Asymptotic (we don't need to know result to do induction!):
  Consider for example that we want to prove that $\sum_{k=0}^{n} 3^k = O(3^k)$, that is, that $\sum_{k=0}^{n} 3^k \leq c3^n$ for some c.
  - Basis: $n = 1 \Rightarrow \quad \sum_{k=0}^{1} 3^x = 1 + 3 = 4$
    $c3^1 = c3$
    Ok if $c > 4/3$
  - Induction:
    Assume holds for n: $\sum_{k=0}^{n} 3^k \leq c3^n$
    Show holds for $n+1$: $\sum_{k=0}^{n+1} 3^k \leq c3^{n+1}$
    Proof:

$$\begin{aligned}
\sum_{k=0}^{n+1} 3^k &= \sum_{k=0}^{n} 3^k + 3^{n+1} \\
&\leq c3^n + 3^{n+1} \\
&= c3^{n+1}(1/3 + 1/c) \\
&\leq c3^{n+1}
\end{aligned}$$

    If $1/3 + 1/c < 1$ which holds if $c > 3/2$

Another important sum: $\boxed{\sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = O(\log n)}$ (*Harmonic Series*)

3 Recurrences

- Last time we discussed divide-and-conquer algorithms

> **Divide and Conquer**
>
> To Solve P:
>
> 1. *Divide* P into smaller problems $P_1, P_2, P_3 ..... P_k$.
> 2. *Conquer* by solving the (smaller) subproblems recursively.
> 3. *Combine* solutions to $P_1, P_2, ... P_k$ into solution for P.

- Analysis of divide-and-conquer algorithms leads to recurrences.

- Merge-sort lead to the recurrence $T(n) = 2T(n/2) + n$
  - or rather, $T(n) = \begin{cases} \Theta(1) & \text{If } n = 1 \\ T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + \Theta(n) & \text{If } n > 1 \end{cases}$
  - but we will often cheat and just solve the simple formula (equivalent to assuming that $n = 2^k$ for some constant k, and leaving out base case and constant in Θ).

3.1 Substitution method

- Idea: Make good guess and prove by induction.

- Lets solve $T(n) = 2T(n/2) + n$ using substitution
  - Guess $T(n) \leq cn \log n$ for some constant c (that is, $T(n) = O(n \log n)$)
  - Proof:
    * Basis: Function constant for small constant n
    * Induction:
      Assume holds for $n/2$: $T(n/2) \leq c\frac{n}{2} \log \frac{n}{2}$
      Show holds for n: $T(n) \leq cn \log n$
      Proof:

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &\leq 2(c\frac{n}{2} \log \frac{n}{2}) + n \\ &= cn \log \frac{n}{2} + n \\ &= cn \log n - cn \log 2 + n \\ &= cn \log n - cn + n \end{aligned}$$

      So ok if $c \geq 1$

- $T(n) = \Omega(n \log n)$ can be proved similarly.

- How do we make a good guess?
  - Something of an art!
  - Try different bounds (e.g. $\Omega(n)$ easy, show $O(n^2) \Rightarrow$ guess $O(n \log n)$)

- Note: *changing variables* can sometimes help
  - Example: Solve $T(n) = 2T(\sqrt{n}) + \log n$

    Let $m = \log n \Rightarrow 2^m = n \Rightarrow \sqrt{n} = 2^{m/2}$
    $T(n) = 2T(\sqrt{n}) + \log n \Rightarrow T(2^m) = 2T(2^{m/2}) + m$

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

$$
\begin{aligned}
T(2^m) = 2T(2^{m/2}) + m &\Rightarrow S(m) = 2S(m/2) + m \\
&\Rightarrow S(m) = O(m \log m) \\
&\Rightarrow T(n) = T(2^m) = S(m) = O(m \log m) = O(\log n \log\log n)
\end{aligned}
$$

- Next time we will discuss another method for solving recurrences.