Important Asymptotic Formulas

Textbook Reading:

Section 3.2, pp. 92-97.

Strongly Asymptotic Result for Polynomials

Proposition. Let $P(n)$ be the k^{th} degree polynomial $a_k n^k + \cdots + a_1 n + a_0$, where $a_k > 0$. Then,

$$P(n) \sim a_k n^k.$$

Corollary. $P(n) \in \Theta(n^k)$.

PSN. Prove the Proposition.

Polynomial and Exponential computing times

An algorithm has **polynomial computing time** if for some positive integer k

$$W(n) \in O(n^k)$$

Exponential order. For some real numbers a and b, where $1 < a \leq b$,

$$a^n \leq f(n) \leq b^n$$

An algorithm has **exponential computing time** if $W(n)$ has exponential order.

Polynomial order less than exponential order

Proposition. For any constants *k* and *a*, where a > 1,

$$O(n^k) \subset O(a^n).$$

We prove the Proposition using the Ratio Limit Theorem and repeatedly applying L'Hôpital's Rule. To compute that derivative of a^x note that $a = e^{\log_e a} = e^{\ln a}$, so that $a^x = (e^{\ln a})^x = e^{(\ln a)x}$.

$$(a^x)' = (e^{(\ln a)x})' = \ln a(e^{(\ln a)x}) = (\ln a)a^x.$$

$$\lim_{n\to\infty} \frac{n^k}{a^n} = \lim_{n\to\infty} \frac{kn^{k-1}}{(\ln a)a^n} = \lim_{n\to\infty} \frac{k(k-1)n^{k-2}}{(\ln a)^2 a^n} =$$

$$\cdots = \lim_{n\to\infty} \frac{k(k-1)\ldots(2)n}{(\ln a)^{k-1}a^n} = \lim_{n\to\infty} \frac{k!}{(\ln a)^k a^n} = 0.$$

The Proposition follows from the Ratio Limit Theorem.

Order Formula for series

$H(n) = 1 + 1/2 + \ldots + 1/n \sim \ln n \in \Theta(\log n)$.

$L(n) = \log(n!) = \log 1 + \log 2 + \ldots + \log n \in \Theta(n \log n)$.

$S(n,k) = 1^k + 2^k + \ldots + n^k \sim n^{k+1} / (k+1)$

Harmonic Series

The Harmonic Series *H*(*n*) is defined by

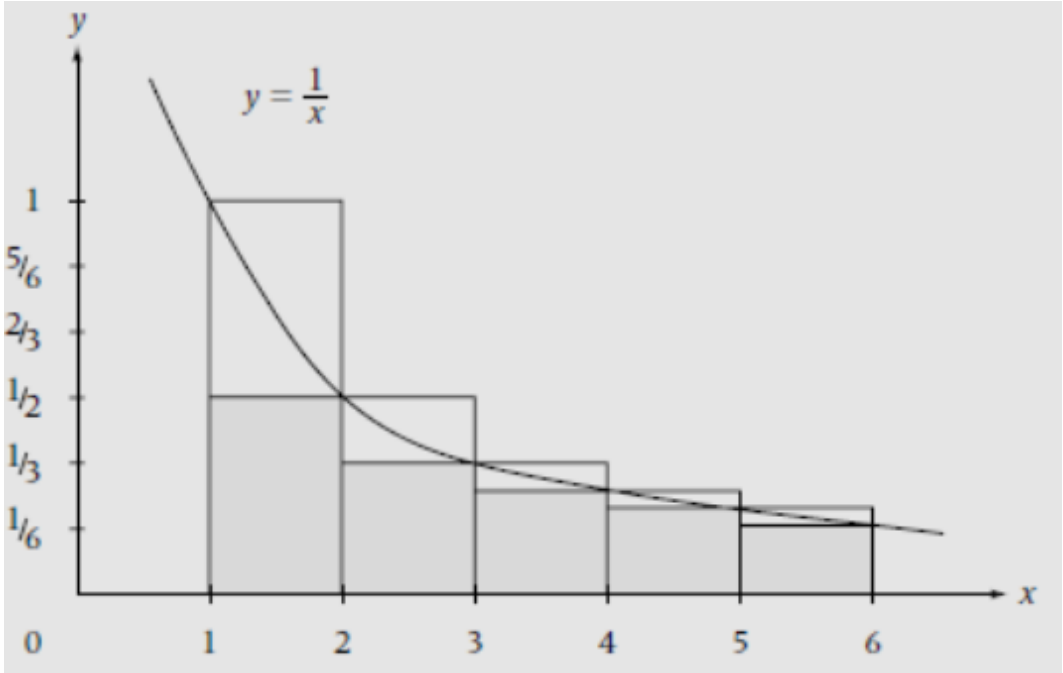
$$H(n) = 1 + \tfrac{1}{2} + \ldots + \tfrac{1}{n}.$$

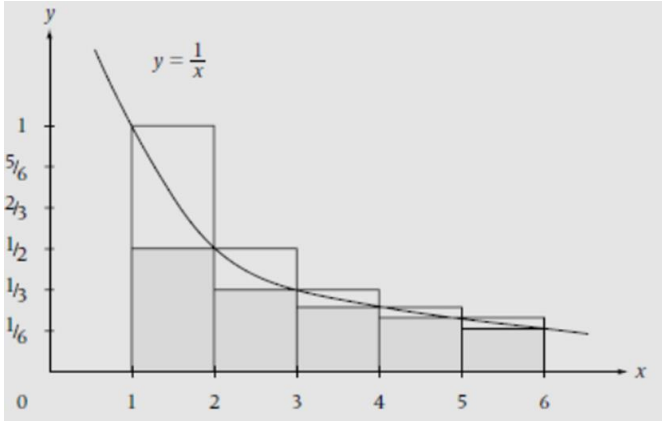
Proposition. $\frac{1}{n} + \ln n < H(n) < 1 + \ln n$.

Corollary. $H(n) \sim \ln n$.

This result will be applied later in this course to obtain a formula for the average complexity of *Quicksort* assuming a uniform distribution.

Proof. Consider the curve $y = 1/x$ from $x = 1$ to $x = n$ and the rectangles above and below as illustrated in the picture below.





Sum of areas of rectangles below curve $<$ $\int_1^n \frac{1}{x}\,dx$ $<$ Sum of areas of rectangles above curve

$$\Rightarrow \frac{1}{2} + \ldots + \frac{1}{n} < \ln n - \ln 1 < 1 + \frac{1}{2} + \ldots + \frac{1}{n-1}$$

$$\Rightarrow H(n) - 1 < \ln n < H(n) - \frac{1}{n}$$

Thus, we have

$$H(n) - 1 < \ln n \Rightarrow H(n) < 1 + \ln n$$

$$\ln n < H(n) - \frac{1}{n} \Rightarrow \frac{1}{n} + \ln n < H(n)$$

Rewriting we have $\frac{1}{n} + \ln n < H(n) < 1 + \ln n$, proving the Proposition.

Connections

Harmonic series was covered in Lab 6 in Models 1 ENED 1090 (which I taught in 2015)

Lab assignment Models 1

PSN. Prove

$f(n) \in \Theta(g(n))$

$\Leftrightarrow f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$

Sum of Logs

$L_2(n) = \log_2 1 + \log_2 2 + \cdots + \log_2 n = \log_2 n!$

Proposition. $L_2(n) \in \Theta(n \log_2 n)$.

Proof. We will use result we just proved that

$L_2(n) \in \Theta(n \log_2 n)$

$\Leftrightarrow L_2(n) \in O(n \log_2 n)$ and $L_2(n) \in \Omega(n \log_2 n)$

Proof $L_2(n) \in O(n \log_2 n)$.

$L_2(n) = \log_2 1 + \log_2 2 + \cdots + \log_2 n$

$< \log_2 n + \log_2 n + \cdots + \log_2 n = n \log_2 n$.

Proof $L_2(n) \in \Omega(n \log_2 n)$.

For convenience assume *n* is even.

$$\log_2 1 + \log_2 2 + \cdots + \log_2 n$$
$$> \log_2\left(\tfrac{n}{2}+1\right) + \log_2\left(\tfrac{n}{2}+2\right) + \cdots + \log_2 n$$
$$> \log_2 \frac{n}{2} + \log_2 \frac{n}{2} + \cdots + \log_2 \frac{n}{2}$$
$$= \frac{1}{2} n \log_2 \frac{n}{2} = \frac{1}{2} n \,(\log_2 n - 1)$$
$$\in \Omega(n \log_2 n).$$

This completes the proof that $L_2(n) \in \Theta(n \log_2 n)$

Order of sum of logs independent of base

As observed earlier the order of the log function does not depend on the base *b*, since $\log_2 n = (\log_2 b) \log_b n$. Letting $L_b(n) = \log_b 1 + \log_b 2 + \cdots + \log_b n$, we have

$$L_2(n) = (\log_2 b) L_b(n) \ .$$

It follows from the Proposition that $L_b(n) \in \Theta(n \log_b n)$.

Therefore we can drop the base and simply write

$L(n) = \log(n!) = \log 1 + \log 2 + \ldots + \log n \in \Theta(n \log n)$.

Application to lower bound for comparison sorting algorithms

- A **comparison-based** algorithm for sorting a list is based on making comparisons involving list elements and then making decisions based on these comparisons.
- Comparison-based algorithms make no a priori assumption about the nature of the list elements, other than knowing their **relative order**.
- For example, the lists (7.2, 1, 8,2), (108, 23.99, 123, 55), (Mary, Ann, Pete, Joe) and (30, π, 31, 12) of size 4 can all be regarded as having the same ordering as the permutation (3,1,4,2). In particular, they each require the same number of comparisons when input to a comparison-based sorting algorithm before they are put into increasing order.
- In the complexity analysis, we may reduce our input space for comparison-based sorting algorithms to the set of all **permutations** on the n integers 1, 2, . . . , n.

There are 3! = 6 permutations of 3 elements:

$$(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)$$

If we make a single comparison, this divides the set of all permutations into two classes C_1 and C_2, where C_1 satisfies the comparisons and C_2 doesn't.

For example, suppose we make the comparison $\pi[1] < \pi[3]$, comparing first and last elements. Then

$$C_1 = \{(1,2,3), (1,3,2), (2,1,3)\} \qquad \pi[1] < \pi[3],$$

$$C_2 = \{(2,3,1), (3,1,2), (3,2,1)\} \qquad \pi[1] > \pi[3],$$

As a second example, if we make the single comparison $\pi[1] < \pi[2]$ then

$$C_1 = \{(1,2,3), (1,3,2), (2,3,1)\} \qquad \pi[1] < \pi[2]$$

$$C_2 = \{(2,1,3), (3,1,2), (3,2,1)\} \qquad \pi[1] > \pi[2]$$

Now suppose we make a sequence of two comparisons, swapping elements that are out of order. Then, we obtain a division into 4 classes, where permutations in the same class have the same relative order for each comparison.

For example suppose we make the sequence of two comparisons $\pi[1] < \pi[3]$ and $\pi[1] < \pi[2]$. Then the 4 classes are

$$C_1 = \{(1,2,3), (1,3,2)\} \qquad \pi[1] < \pi[3] \text{ and } \pi[1] < \pi[2].$$

$$C_2 = \{(2,1,3)\} \qquad \pi[1] < \pi[3] \text{ and } \pi[1] > \pi[2].$$

$$C_3 = \{(2,3,1), (3,2,1)\} \qquad \pi[1] > \pi[3] \text{ and } \pi[1] < \pi[2].$$

$$C_4 = \{(3,1,2)\} \qquad \pi[1] > \pi[3] \text{ and } \pi[1] > \pi[2].$$

Lower Bound for Comparison-Based Sorting

- In general if we make a sequence of ***m* comparisons**, swapping elements that are out of order, for a list of size *n*, this will divide the set of all ***n*! permutations** into at most **2^m classes**, where permutations in the same class have the same relative order for each comparison and swap made.

- If some class contains two permutations π_1 and π_2 , then the algorithm can not be correct. Because it is comparison-based it must do **precisely the same sequence of operations** for π_1 and π_2 , so it couldn't sort both.

- It follows from the **Pigeon-Hole Principle** that the number of classes must be at least as great as the number of permutations. Thinking of the classes as pigeonholes and the permutations as pigeons, if the latter weren't true, at least one pigeonhole, i.e., class, would contain two (or more) pigeons, i.e., permutations.

Lower Bound Argument cont'd

- Thus, we have $2^m \geq \#\ of\ classes \geq n! \Rightarrow m\ \geq \log_2 n!$, and using are sum of logs results, we have $\boldsymbol{m\ \in \Omega(n \log n)}$. It follows that the worst-case complexity of any comparison-based sorting algorithm satisfies $\boldsymbol{W(n) \in \Omega(n\, log\, n)}$.
- This result is order sharp in the sense that there exist comparison-based algorithms, such as Mergesort, whose order worst-case complexity achieves this lower bound.
- Thus, the **problem** of comparison-based sorting has **order worst-case complexity** $\boldsymbol{n \log n}$.

Recurrence Relation for $S(n,k) = 1^k + 2^k + \ldots + n^k$

There is no known general formula for $S(n,k)$, but for any given k a formula can be computed using the recurrence relation:

$$S(n,k) = \frac{(n+1)^{k+1}}{k+1} - \frac{1}{k+1}\left(1 + \binom{k+1}{0} S(n,0) + \binom{k+1}{1} S(n,1) + \cdots + \binom{k+1}{k-1} S(n,k-1)\right)$$

Initial Condition: $S(n,0) = n$.

Applying this recurrence relation, a simple induction shows that the coefficient of the highest degree term in $S(n,k)$ is $\frac{n^{k+1}}{k+1}$, so that

$$\boldsymbol{S(n,k) \sim \frac{n^{k+1}}{k+1}} .$$

One iteration of the above recurrence relation yields the formula $S(n,1) = \frac{n(n+1)}{2}$, which we have already used in the average complexity analysis of Linear Search and Insertion Sort.

Why did the military arrest all the pigeons?

They were starting a coo.