COSC 336 Data Structures and Algorithms

Mathematical tools for the analysis of algorithms

Professor: Marius Zimand

1 Important functions

The functions that appear commonly in the analysis of algorithms are: logarithms, polynomials, exponential, factorial.

1.1 Polynomials.

A polynomial function is a function of the type

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0,$$

where the coefficients $a_k, \ldots a_0$ are constants. This is a polynomial of degree k.

Typically in algorithms $a_k > 0$, which we assume henceforth.

The behavior of a polynomial of degree k is controlled by the dominating term n^k. $p(n) = \Theta(n^k)$.

The simple but important fact to remember is that $\frac{n^k}{n^{k-1}}$ goes to ∞ as n goes to ∞.

So if $p(n), q(n)$ are two polynomials and the degree of p is larger than the degree of q, then $q(n)/p(n)$ goes to 0, which is also written as $q(n) = o(p(n))$, or, equivalently, $p(n) = \omega(q(n))$. Ex: $3n^3 + 2n^2 + 5n + 7 = O(n^4 + 2n + 7)$

Actually $3n^3 + 2n^2 + 5n + 7 = \Theta(n^3)$

1.2 Exponential function.

The exponential function has the form $f(n) = a^n$ for some constant $a > 1$ (It is possible that $a < 1$, but such situations are less common in algorithms).

Consider a process in which a parameter n has initially the value 1 and doubles at each iteration:

$$1 \to 2 \to 4 \to \ldots \to 2^n$$

Thus in n iterations, the parameter becomes exponentially large. In general if the initial value is a_0 and at each iteration the parameter gets multiplied by q, then at iteration n, it becomes $a_0 q^n$.

The exponential function also appear when counting different objects. Thus, the number of binary strings of length n is 2^n, and more generally the number of words of length n over an alphabet with q letters is q^n.

Main Properties:

- $a^0 = 1$.
- $a^{-1} = 1/a$.
- $a^m \cdot a^n = a^{m+n}$.
- $(a^m)^n = a^{mn}$
- The exponential function grows faster than any polynomial. Formally, for every $k \geq 0$, $\lim_{n\to\infty} \frac{n^k}{a^n} = 0$.

  For example, $n^{3000} < 2^{0.01n}$ if n is sufficiently large.

  Proof sketch: We write $a = 1 + c$, with $c > 0$. Then

  $$a^n = 1 + \binom{n}{1}c + \binom{n}{2}c^2 + \ldots + \binom{n}{k+1}c^{k+1} + \ldots + c^n \text{ (binomial formula).}$$

  So,

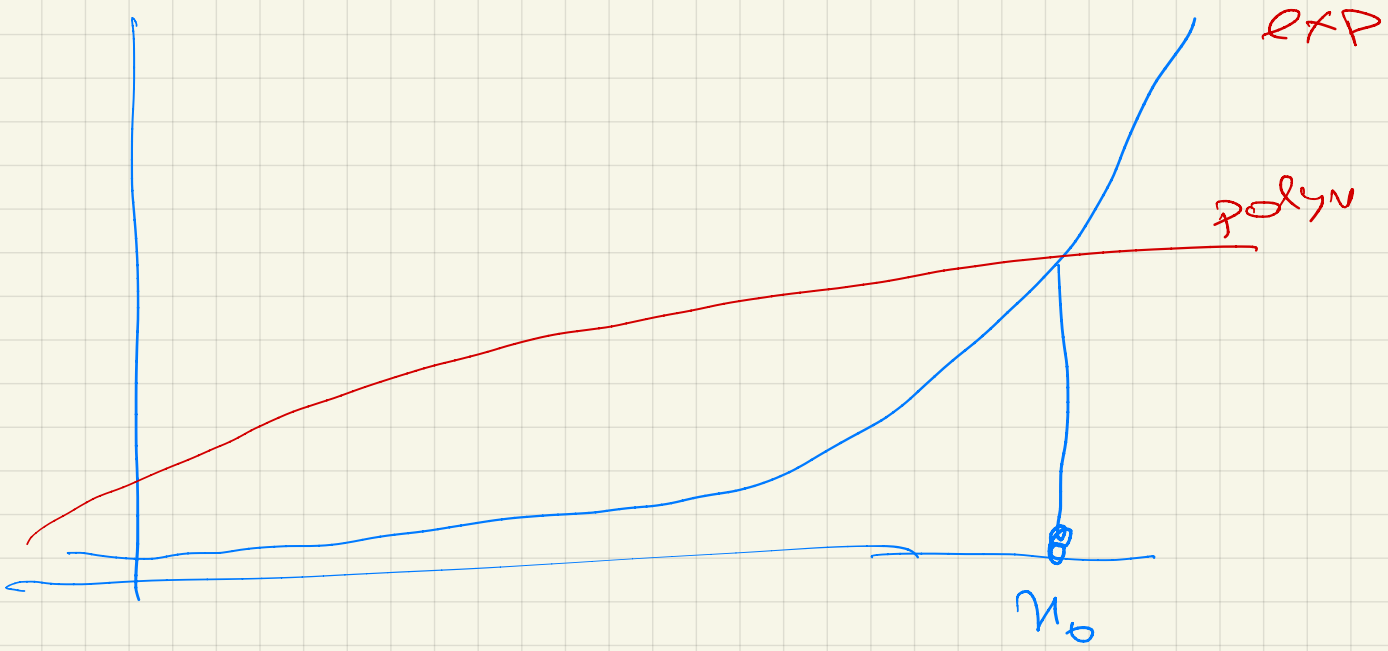
  $a^n > \binom{n}{k+1}c^{k+1} = \frac{n(n-1)\ldots(n-k)}{(k+1)!}c^{k+1}$.

  Then,

  $$\frac{n^k}{a^n} < \frac{n^k}{n(n-1)\ldots(n-k)} \cdot (k+1)!/c^{k+1}.$$

  The right hand side goes to 0 because $(k+1)!/c^{k+1}$ is a constant and in the fraction the numerator is a polynomial of degree k, and the denominator is a polynomial of degree $k+1$.

(1)



(2) $2^{2n} = 2^n \cdot 2^n$, so $2^{2n} = \omega(2^n)$, not $O(2^n)$.

We cannot drop a multiplicative constant in the exponent

$$2^{n+10} = 2^n \cdot 2^{10} = 1024 \cdot 2^n = \Theta(2^n)$$

We drop additive constants in the exponent

1.3 Logarithms.

By definition $\log_b n$ is the number a such that $b^a = n$. The number b is called the base and it is required that $b \neq 1$.

For example, $\log_2 n$ is the number a such that $2^a = n$. By default in computer science, if the base is not specified, it is equal to 2.

Some occurrences of the logarithmic function

Consider a process that depends on a parameter n, and suppose the parameters halves at each step:

$$n \to n/2 \to n/2^2 \to \ldots \to n/2^k \ldots$$

It can be shown that the number of steps till the parameter becomes ≤ 1 for the first time is $\lceil \log_2 n \rceil \simeq \log n$.

The length in bits for writing an integer n in base 2 is $\lceil \log(n+1) \rceil \approx \log n$.

Main Properties

32 → 100000 ← base 2 (6 bits); base 10

- $a = 2^{\log a}$ for all $a > 0$. In general $a = b^{\log_b a}$.
- $\log(ab) = \log a + \log b$ and $\log a/b = \log a - \log b$
- $\log a^b = b \log a$ Ex: $\log(n^2) = 2 \log n$
- formula for changing the base: $\log_{\text{new-base}} a = \frac{\log_{\text{old-base}} a}{\log_{\text{old-base}} \text{new-base}}$.

  Example: $\log_2 5 = \log_{10} 5 / \log_{10} 2$.
- By default, if the base is not written explicitly, it is 2.
- The logarithm even raised to a constant power grows slower than any polynomial functions. More formally, $\lim_{n\to\infty} \frac{(\log n)^k}{n} = 0$.

  This is easy to see. We denote $m = \log n$ and then $n = 2^m$ and the ratio becomes $m^k/2^m$ which goes to 0 as we know from the properties of the exponential function. For example, $(\log n)^{1000} < n^{0.001}$ if n is sufficiently large.

log < polyn.

Repeated Halving : $n \to \frac{n}{2} \to \frac{n}{4} \to \cdots \to 1$ (or less)

- $32 \xrightarrow{1} 16 \xrightarrow{2} 8 \xrightarrow{3} 4 \xrightarrow{4} 2 \xrightarrow{5} 1$

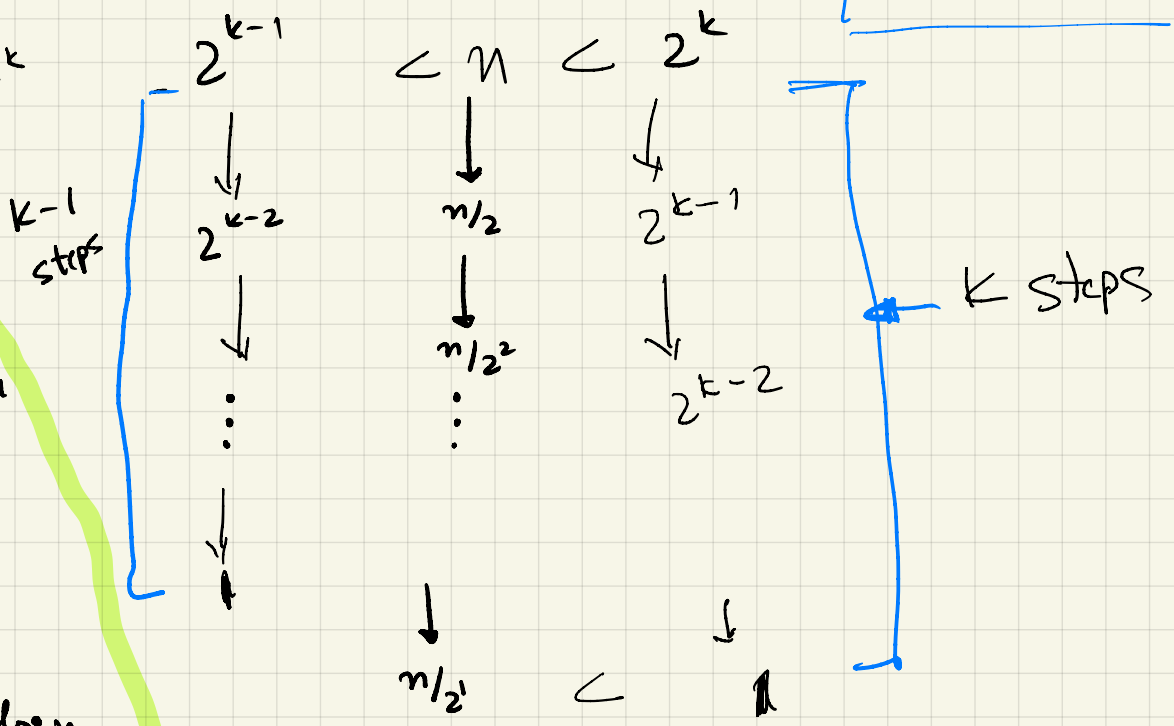
$5 = \log_2 32$

- $2^k \to 2^{k-1} \to 2^{k-2} \to \cdots \to 2^0 = 1$

no. of steps : k

- If n is not a power of 2. → $n = 45$, $32 < 45 < 64$

$\exists k$ s.t $2^{k-1} < n < 2^k$



- $n < 2^k$

$\Rightarrow k > \log n$

- $n > 2^{k-1}$

$\Rightarrow \log n > k-1$

$k = \lceil \log n \rceil \simeq \log n$

$$n \longrightarrow \frac{n}{b} \longrightarrow \frac{n}{b^2} \longrightarrow \cdots \longrightarrow \frac{n}{b^k} \quad , b > 1$$

It takes $\log_b n$ steps to get from n to ≤ 1.

$$\log_b n = \Theta(\log n)$$

1.4 Factorial function

$n! = 1 \cdot 2 \cdot \ldots \cdot n$.

Ex: $3! = 1 \cdot 2 \cdot 3 = 6$

Some occurrences of the factorial

The number of permutations of n different object is $n!$.

"n choose k"

The number of combinations of k chosen from n is $\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\ldots(n-k+1)}{k!}$.

Approximating the factorial

Using calculus, we can derive the following approximation of $n!$:

(Stirling approximation): $n! \approx \sqrt{2\pi n}(n/e)^n$.

While the Stirling approximation is somewhat more difficult to derive, we can easily show the following bounds: $(n/2)^{n/2} < n! < n^n$.

Indeed $n! = 1 \cdot 2 \cdot \ldots \cdot n < n \cdot n \cdot \ldots \cdot n = n^n$, and

$n! > (n/2+1) \cdot \ldots \cdot n > (n/2)(n/2)\ldots(n/2)$ ($n/2$ factors).

Thus $n!$ grows faster than the exponential 2^n.

Also, $\log(n!) = \Theta(n \log n)$.

Order of growth of some important functions

constant $< \log\log n < \log n < \log^k n$(for $k > 1$) $< n < n^k$(for $k > 1$) $< 2^n < n! < 2^{2^n}$

a, b, c

a, b, c
a, c, b
b, a, c
b, c, a
c, a, b
c, b, a

6 permutations

$6 = 3!$

$\binom{n}{k}$: the number of ways to select k objects out of n objects.

read: "n choose k" or "combinations of k from n"

a, b, c, d

committees of 2 people

a, b
a, c
a, d
b, c
b, d
c, d

$\binom{4}{2} = \frac{4 \cdot 3}{1 \cdot 2} = 6$

$$\binom{n}{k} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{k!}$$

$$\binom{7}{3} = \frac{7 \cdot 6 \cdot 5}{1 \cdot 2 \cdot 3} = 35$$

$$n! = 1 \cdot 2 \cdot \ldots \cdot n < n \cdot n \cdot n \cdot \ldots \cdot n = n^n$$

$$n! = 1 \cdot 2 \cdot \underbrace{\ldots \cdot n}_{} = 1 \cdot 2 \cdot \ldots \cdot \frac{n}{2} \cdot \left(\frac{n}{2}+1\right) \cdot \ldots \cdot n$$

$$\geq \left(\frac{n}{2}+1\right)\left(\frac{n}{2}+2\right) \cdots \cdot n$$

$$\geq \frac{n}{2} \cdot \frac{n}{2} \cdot \ldots \cdot \frac{n}{2} = \left(\frac{n}{2}\right)^{n/2}$$

So $\left(\frac{n}{2}\right)^{n/2} < n! < n^n$

Take log. $\log\left(\frac{n}{2}\right)^{n/2} < \log n! < \log n^n$

$$\frac{n}{2} \cdot \log \frac{n}{2} < \log(n!) < n \cdot \log n$$

So: $\log(n!) = \Theta(n \cdot \log n)$

a, b, c → permutations

a, b, c
a, c, b
b, a, c
b, c, a
c, a, b
c, b, a

No. of permutations

$3! = 6$

- 4 persons: a, b, c, d
- We want to make a committee of 2
- How many committees are possible?

$\binom{4}{2}$ "4 choose 2"

a, b
a, c
a, d
b, c
b, d
c, d

all possible committees

$$\binom{4}{2} = \frac{4 \cdot 3}{1 \cdot 2} = 6$$

General formula for "n choose k"

$$\binom{n}{k} = \frac{n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-k+1)}{1 \cdot 2 \cdot 3 \cdot \ldots \cdot k}$$

Ex: $\binom{7}{3} = \frac{7 \cdot 6 \cdot 5}{1 \cdot 2 \cdot 3} = 35$

2 Sums, and Series

The setting is that there is a sequence $u_1, u_2, \ldots, u_n, \ldots$ (many times it is an infinite sequence).

Let $S_n = u_1 + \ldots + u_n$. We want to find S_n or to estimate S_n, which means to find lower and upper bounds for S_n.

Arithmetic series $a, a+d, \ldots, a+(n-1)d, \ldots$ (so, recalling the general setting, $u_n = a+(n-1)d$)

$S_n = na + d\frac{n(n-1)}{2}$

Important particular case: $1 + 2 + \ldots + n = \frac{n(n+1)}{2}$.

Geometric series $a, a \cdot r, a \cdot r^2, \ldots, a \cdot r^{n-1}$, where $r \neq 1$.

$S_n = a \cdot \frac{r^n - 1}{r-1}$.

Important particular case: $1 + r + \ldots + r^{n-1} = \frac{r^n-1}{r-1}$. if $r \neq 1$.

Harmonic series

Let $H_n = 1 + 1/2 + 1/3 + \ldots + 1/n$.

We have the following estimation:

$$H_n = \ln n + O(1).$$

exact value

asymptotic value

Other sums

$T_1 = 1 + 2 + \ldots + n = n(n+1)/2.$ $= \Theta(n^2)$

$T_2 = 1^2 + 2^2 + \ldots + n^2 = n(n+1)(2n+1)/6.$ $= \Theta(n^3)$

$T_3 = 1^3 + 2^3 + \ldots + n^3 = [n(n+1)/2]^2.$ $= \Theta(n^4)$

In general $T_r = 1^r + 2^r + \ldots + n^r = \Theta(n^{r+1})$ (see proof below, using the method of integrals).

$1^7 + 2^7 + \ldots + n^7 = \Theta(n^8)$

$a, a+d, a+2d, \dots, a+(n-1)d$

We add d at each step

$$S_n = a + (a+d) + \dots + (a+(n-1)d) =$$

$$= n \cdot a + d(1+2+\dots+(n-1)) = n \cdot a + d \cdot \frac{(n-1)\cdot n}{2}$$

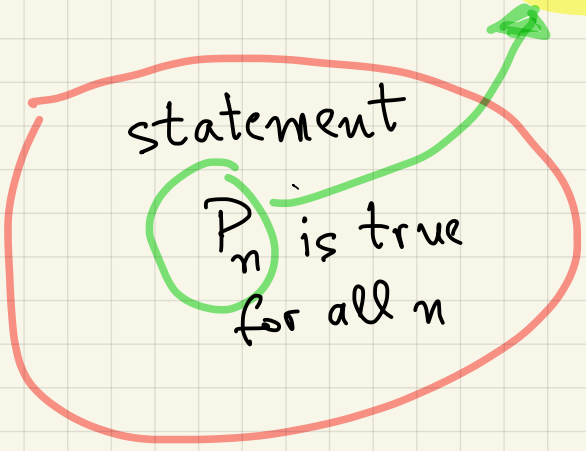
$$S = 1+2+\dots+n$$

$$S = n+(n-1)+\dots+1$$

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

$$S = 1+2+\dots+n$$

We can also prove $S = \frac{n(n+1)}{2}$ by induction



$$1+3+5+7+\dots+2n-1 =$$

$$= (2\cdot 1-1)+(2\cdot 2-1)+(2\cdot 3-1)+\dots+(2\cdot n-1)$$

$$= 2(1+2+\dots+n) - n$$

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

$$= n^2+n-n = n^2$$

$S = 1 + 2 + \cdots + n$ statement P_n

We want to prove $S = \frac{n(n+1)}{2}$ by induction

I (Base) Prove it for $n = 1$

$S = 1$

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

So P_1 is true

II (Inductive step)

We assume that statement is true for n

We prove it for n+1.

We assume: $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$ Assume P_n is true

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

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

$= \frac{(n+1)(n+2)}{2}$, DONE! P_{n+1} is true

$$a,\ a\cdot r,\ a\cdot r^2,\ \dots,\ a\cdot r^{n-1}$$

$$S = a + a\cdot r + a\cdot r^2 + \dots + a\cdot r^{n-1}$$

$$= a\left(1 + r + r^2 + \dots + r^{n-1}\right) = a\cdot\frac{r^n - 1}{r-1}$$

$$T = 1 + r + r^2 + \dots + r^{n-1}$$

$$r\cdot T = r + r^2 + r^3 + \dots + r^n$$

$$r\cdot T - T = r^n - 1$$

$$T(r-1) = r^n - 1, \quad \text{If } r \neq 1$$

$$T = \frac{r^n - 1}{r-1}$$

$$1 + r + r^2 + \dots + r^{n-1} = \frac{r^n - 1}{r-1}$$

2.1 Approximating sums with integrals

Using integrals, we can estimate sums of the form $S_n = f(1) + f(2) + \ldots + f(n)$. If f is an increasing function then

$$\int_0^n f(x)dx \leq S_n \leq \int_1^{n+1} f(x)dx$$

If f is decreasing then

$$\int_1^{n+1} f(x)dx \leq S_n \leq \int_0^n f(x)dx$$

Example: We want to estimate $T_r = 1^r + 2^r + \ldots + n^r$. We use $f(n) = n^r$, so $T_r = f(1) + \ldots + f(n)$. Since $f(n)$ is an increasing function, we obtain using integrals,

$$T_r \geq \int_0^n f(x)dx = \int_0^n x^r dx = x^{r+1}/(r+1)\Big|_0^n = n^{r+1}/(r+1).$$

and

$$T_r \leq \int_1^{n+1} f(x)dx = \int_1^{n+1} x^r dx = x^{r+1}/(r+1)\Big|_0^{n+1} = (n+1)^{r+1}/(r+1) - 1/(r+1).$$

Combining the two estimations, we see that $T_r = \Theta(n^{r+1})$.

Sometimes, the integral involved in the estimation is not defined in the domain of interest, and then we have to be a little more careful and split the sum in two parts, so that the part where we use the estimations with integrals does not present this problem.

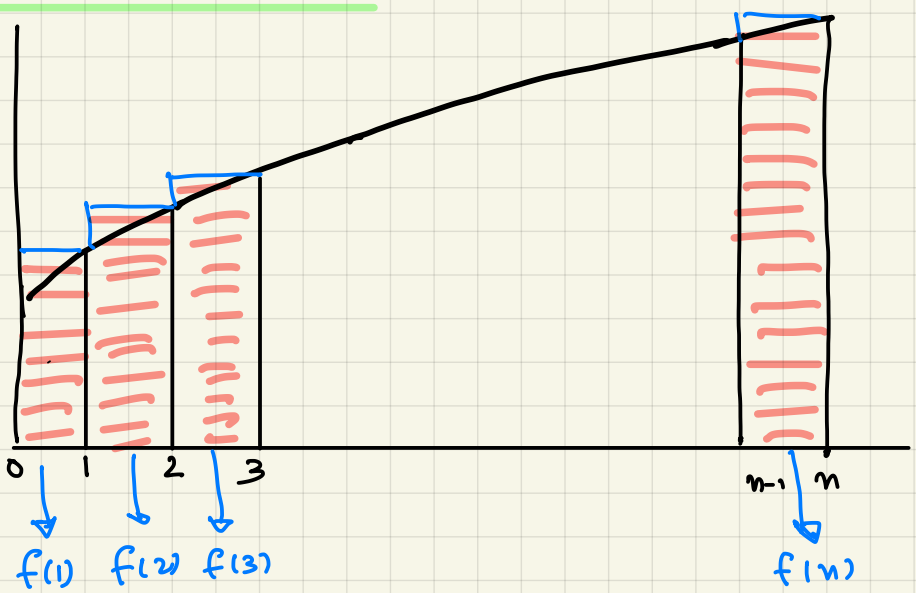
For example, suppose we want to estimate the harmonic series $H_n = 1 + 1/2 + \ldots + 1/n$. We take the function $f(x) = 1/x$, which is decreasing. The lower bound does not raise any problem:

$$H_n \geq \int_1^{n+1} 1/x \; dx = \ln x\Big|_1^{n+1} = \ln(n+1).$$

For the upper bound, we would like to say that $H_n \leq \int_0^n 1/x \; dx$ but this is wrong because $1/x$ is not defined at 0. So we split H_n into 1 and $(1/2 + \ldots + 1/n)$, that is $H_n = 1 + (1/2 + \ldots + 1/n)$.

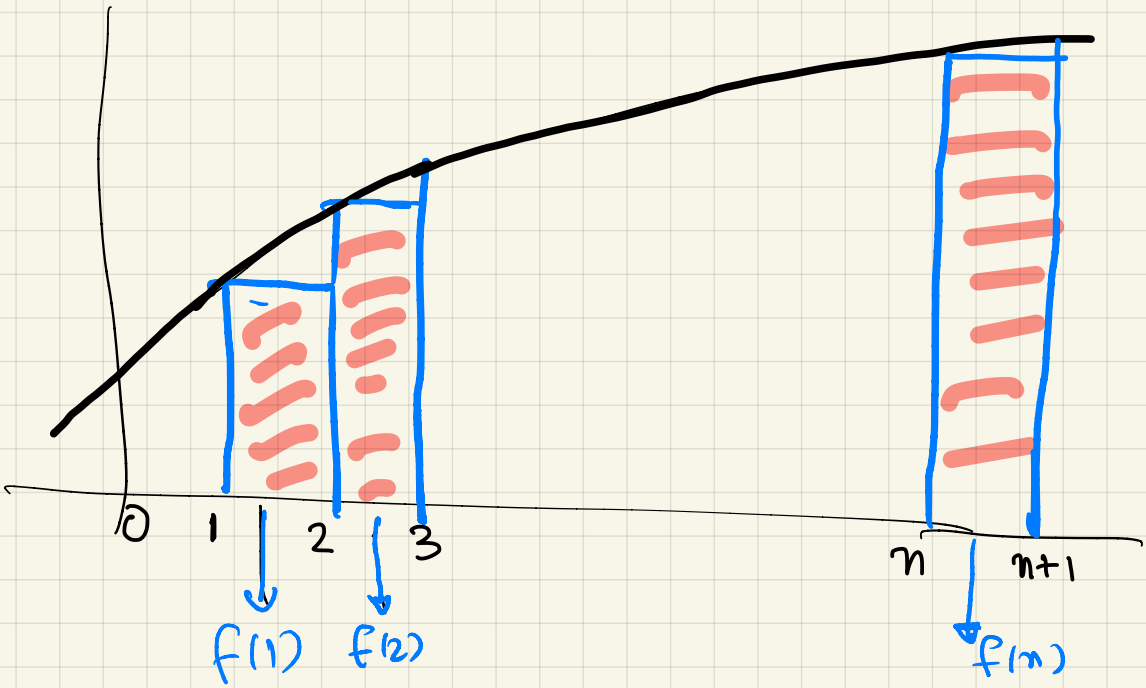
Now, for the second part we can use the integral technique and obtain $(1/2 + \ldots + 1/n) \leq \int_1^n 1/x \; dx = \ln n$. And we get that $H_n \leq 1 + \ln n$.

f is increasing



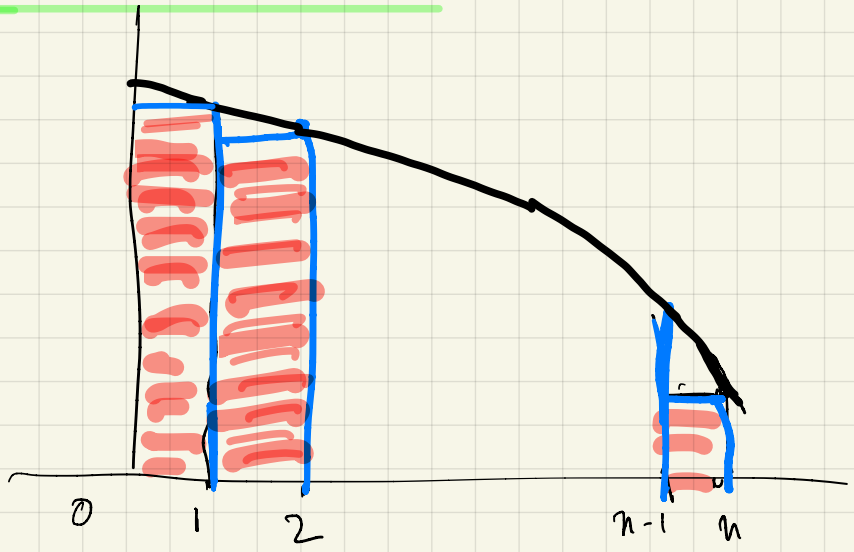
$$S_n = f(1) + f(2) + \dots + f(n)$$

$$S_n = \text{area} \geqslant \int_0^n f(x)\,dx$$



$$S_n \leq \int_1^{n+1} f(x)\,dx$$

f is decreasing



$$S_n \leq \int_0^n f(x)\,dx$$

area of rectangles

area under the curve

Similarly: $S_n \geq \int_1^{n+1} f(x)\,dx$

$$S_n = 1^r + 2^r + \dots + n^r$$, r is a constant

$r \geq 1$

$f(x) = x^r$ is ↗

$$S_n = f(1) + f(2) + \dots + f(n)$$

$$S_n \geq \int_0^n f(x)\,dx = \int_0^n x^r\,dx$$

$$= \frac{x^{r+1}}{r+1}\Big|_0^n = \frac{n^{r+1}}{r+1} - \frac{0^{r+1}}{r+1}$$ $= 0$

$$S_n \leq \int_1^{n+1} f(x)\,dx = \int_1^{n+1} x^r\,dx$$

$$= \frac{x^{r+1}}{r+1}\Big|_1^{n+1} = \frac{(n+1)^{r+1}}{r+1} - \frac{1^{r+1}}{r+1}$$

So:

$$\frac{n^{r+1}}{r+1} \leq S_n \leq \frac{(n+1)^{r+1}}{r+1} - \frac{1}{r+1}$$

So $S_n = \Theta(n^{r+1})$.

$$H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$$

$f(x) = \frac{1}{x}$; $H_n = f(1) + \cdots + f(n)$.

↓

f is decreasing

$$H_n \le \int_0^n f(x)\,dx \overset{??}{=} \int_0^n \frac{1}{x}\,dx$$

$$H_n = 1 + \underbrace{\left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)}_{S}$$

$$S \le \int_1^n f(x)\,dx = \int_1^n \frac{1}{x}\,dx$$

$$= \ln x \Big|_1^n = \ln n - \ln 1 = \ln n$$

So, $H_n \le 1 + \ln n$

$$H_n \geq \int_1^{n+1} \frac{1}{x}\,dx = \ln x \Big|_1^{n+1}$$

$$= \ln(n+1) - \ln 1 = \ln(n+1).$$

So:

$$\ln(n+1) \leq H_n \leq 1 + \ln n$$

$$H_n \simeq \ln n$$

$$H_n = \Theta(\ln n)$$

$$H_n = \ln n + \Theta(1).$$

Example with Geometric series

```
i = n
while (i ≥ 1) {
    for (j = 1 to i)
        x = x+1
    i = i/2
}
```

Find $\Theta(\cdot)$ evaluation for runtime $T(n)$.

Solution: $n + \frac{n}{2} + \frac{n}{2^2} + \cdots + \frac{n}{2^k}$

where $k = \log n$

$= n\left(1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^k}\right)$

$= n \frac{1 - \left(\frac{1}{2}\right)^{k+1}}{1 - 1/2}$

$\leq \frac{n}{1/2} = 2n$

2.2 Other techniques for estimating sums

Bounding the terms

Example:

$1^r + 2^r + \ldots + n^r \leq n^r + n^r + \ldots + n^r = n \cdot n^r = n^{r+1}$.

"CUT-AND-BOUND"

Cutting some terms and bounding

Example:

$$\begin{aligned} 1^r + 2^r + \ldots + n^r &> (n/2+1)^r + (n/2+2)^r + \ldots + n^r \\ &> (n/2)^r + (n/2)^r + \ldots + (n/2)^r \\ &= (n/2)(n/2)^r \\ &= (1/2^{r+1})n^{r+1}. \end{aligned}$$

Telescoping

Example:

$\frac{1}{1\cdot 2} + \frac{1}{2\cdot 3} + \ldots + \frac{1}{n\cdot(n+1)} = (\frac{1}{1} - \frac{1}{2}) + (\frac{1}{2} - \frac{1}{3}) + \ldots + (\frac{1}{n} - \frac{1}{n+1}) = 1 - \frac{1}{n+1}$.

$$S_n = 1^r + 2^r + \cdots + n^r$$

$$S_n \le n^r + n^r + \cdots + n^r = n \cdot n^r = n^{r+1}.$$

$$S_n \ge \left(\frac{n}{2}+1\right)^r + \left(\frac{n}{2}+2\right)^r + \cdots + \left(\frac{n}{2}+\frac{n}{2}\right)^r$$

$$\ge \left(\frac{n}{2}\right)^r + \left(\frac{n}{2}\right)^r + \cdots + \left(\frac{n}{2}\right)^r$$

$$= \frac{n}{2} \cdot \left(\frac{n}{2}\right)^r = \left(\frac{n}{2}\right)^{r+1}$$

So:

$$\left(\frac{n}{2}\right)^{r+1} \le S_n \le n^{r+1}$$

$$\Rightarrow \quad S_n = \Theta(n^{r+1}).$$

$S_n \ge 1 + 1 + \cdots + 1 = n$ → CORRECT

But we can do better.

3 Asymptotic notation:

When we analyze algorithms, many times we want to just get a rough estimate and then it is ok to ignore constants and what happens for small values of the parameters.

In what follows we assume that functions have positive integers inputs.

1. $t(n) = O(f(n))$ (read $t(n)$ is in Big-Oh of $f(n)$, or just $t(n)$ is Big-Oh of $f(n)$) means that there exist positive constants c and n_0 such that $t(n) \leq c \cdot f(n)$ for all $n \geq n_0$.

   The intuitive meaning is that $t(n)$ "$\leq$" $f(n)$ if we ignore multiplicative constants and small values of n.

   Sum rule: $f(n) + g(n) = O(\max(f(n), g(n)))$.

2. $t(n) = \Omega(f(n))$ (read $t(n)$ is in Big-Omega of $f(n)$ or just $t(n)$ is Big-Omega of $f(n)$) means that there exist positive constants c and n_0 such that $t(n) \geq c \cdot f(n)$ for all $n \geq n_0$.

   The intuitive meaning is that $t(n)$ "$\geq$" $f(n)$ if we ignore multiplicative constants and small values of n.

   Sum rule: $f(n) + g(n) = \Omega(\min(f(n), g(n)))$.

3. $t(n) = \Theta(f(n))$ (read $t(n)$ is in Big-Theta of $f(n)$ or just $t(n)$ is Theta of $f(n)$) means that $t(n) = O(f(n))$ and $t(n) = \Omega(f(n))$.

   The intuitive meaning is that $t(n)$ " = " $f(n)$ if we ignore multiplicative constants and small values of n.

4. $t(n) = o(f(n))$ (read $t(n)$ is in Little-Oh of $f(n)$ or just $t(n)$ is Little-Oh of $f(n)$) means that $\lim_{n\to\infty} \frac{t(n)}{f(n)} = 0$.

   The intuitive meaning is that $t(n)$ "< " $f(n)$ if we ignore small values of n.

5. $t(n) = \omega(f(n))$ (read $t(n)$ is in Little-Omega of $f(n)$ or just $t(n)$ is Little-Omega of $f(n)$) means that $\lim_{n\to\infty} \frac{t(n)}{f(n)} = \infty$.

   The intuitive meaning is that $t(n)$ "> " $f(n)$ if we ignore small values of n.

3.1 How to establish the relative order of growth of two functions

We want to establish the asymptotic relation between $f(n)$ and $g(n)$. For that we use:

1. What we know about some common functions. Example $n \log n = o(n^2)$ because we know that $\log n = o(n)$.

2. We calculate $\lim_{n\to\infty} f(n)/g(n)$

   If the limit is some constant $c > 0$, then $f(n) = \Theta(g(n))$.

   If the limit is $c = 0$, then $f(n) = o(g(n))$ and $g(n) = \omega(f(n))$.

   If the limit is $+\infty$, then $f(n) = \omega(g(n))$ and $g(n) = o(f(n))$

EXERCISES

- $n \log n = O(n^2)$?
- $n^2 = O(n^{1.5} \log 10n)$?
- $2^n = O(n^{1000000})$?
- $(\sqrt{2})^{\log n} = O(n^{1/3})$?
- $n^{\log \log n} = O((\log n)^{\log n})$?
- $2^n = O(4^{\log n})$?
- $n! = O(2^n)$?
- $n! = O(n^n)$?
- $n2^n = O(2^{n \log n})$?

- $\log(2n) = O(\log n)$?

$2^{2n} = O(2^n)$?

$n \cdot \log n$ vs. n^2

$\log n$ vs. n

So: $n \cdot \log n = o(n^2)$

$= O(n^2)$

---

n^2 vs. $n^{1.5} \cdot \log(10n)$

$n^{0.5}$ vs $\log(10n)$ → CLEAR

$\sqrt{2}^{\log n}$ vs $n^{1/3}$

↓

$2^{\frac{1}{2} \cdot \log n} = 2^{\log n^{1/2}} = n^{1/2}$

$n^{\log\log n}$ vs. $(\log n)^{\log n}$

$= 2^{\log(n^{\log\log n})}$

$= 2^{\log\log n \cdot \log n}$

$= 2^{\log((\log n)^{\log n})}$

$= 2^{\log n \cdot \log\log n}$

2^n vs. $4^{\log n}$

$= (2^2)^{\log n} = 2^{\log(n^2)} = n^2$

$n \cdot 2^n$ vs. $2^{n \log n}$

$\|$

$2^{\log n} \cdot 2^n$

$\|$

$2^{n + \log n}$ vs $2^{n \log n}$

$\log 2n$ vs $\log n$

$\|$

$\log 2 + \log n$ vs $\log n$

4 The Master Theorem

When we do divide-and-conquer algorithms, we typically need to solve recurrences of the form

$$t(n) = at(\frac{n}{b}) + f(n),$$

where $a \geq 1, b > 1$. This happens when we divide the problem of size n into a subproblems each of size (n/b) and do extra $f(n)$ steps.

The Master Theorem gives the asymptotic solution for the above recurrence:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $t(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $t(n) = \Theta(f(n) \log n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, then $t(n) = \Theta(f(n))$.

So, in short, we compare $f(n)$ and $n^{\log_b a}$, and see which one is bigger in the asymptotic sense. We call the bigger function, the winner, and the smaller function, the loser.

If there is a tie (case 2, above), the $t(n) = \Theta(f(n) \log n)$.

If the winner is bigger by the loser by some n^ϵ (cases 1 and 3), then $t(n) = \Theta(winner)$.