Big Oh Notation

This is an intuitive approach using limits. The use of limits here can also be done intuitively. Previous knowledge of limits is not assumed.

Consider the table below

$f(n)$ is the function

n	$n^{1/3}$	$n^{1/2}$	$n \log n$	n^2	n^3	n^8	2^n	4^n	$n!$
1	1	1	0	1	1	1	2	4	1
2	1.3	1.4	1.4	4	8	256	4	16	2
3	1.4	1.7	3.3	9	27	6561	8	64	6
10	2.2	3.2	23	100	10^3	10^8	1024	10^6	3.6×10^6
20	2.7	4.5	60	400	8×10^3	2.6×10^{10}	10^6	10^{12}	2.4×10^{18}
100	4.6	10	460	10^4	10^6	10^{16}	1.3×10^{30}	1.6×10^{60}	9.3×10^{157}
1000	10	31.6	6900	10^6	10^9	10^{24}	10^{301}	1.1×10^{602}	4×10^{2567}

Most entries are approximate.

The functions in the table are in increasing order (in terms of big oh) going from left to right. (Although the function n fits between $n^{1/2}$ and $n \log n$.) That is $O(n^{1/3}) < O(n^{1/2}) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(n^8) < O(2^n) < O(4^n) < O(n!)$. Note that this does not say $n^8 < 2^n$ for all values of n. (Certainly it isn't true for $n=2$.) Big Oh "measures" what happens for large values of n. (Already by $n = 100$, it takes twice as many digits to write out ~~n^8~~ to 2^n as it does to write out n^8.)

Also big oh is a "rough measure", that is $O(n^8) = O(13 n^8)$. Multiplying a function by a positive constant does not change its big oh. After all, multipling the entries of the n^8 column by 13 isn't going to help it catch up with 2^n.

There are two useful rules in the table. The first is the approximation $2^{10} = 1024 \sim 10^3$. Thus $2^{100} = (2^{10})^{10} \sim (10^3)^{10} = 10^{30}$ and $2^{24} = (2^{10})^2 \cdot 2^4 \sim 16 \times 10^6$. The second rule is hidden better. It is $O(n \log n) = O(\log(n!))$. (The log used in the ~~above~~ table is the natural log.) More on this second rule later.

Well it's time to give a way of determining when $O(f) = O(g)$ or $O(f) < O(g)$. The theorem below doesn't always do this for general functions $f(n)$ and $g(n)$. But it will work for the functions found in this book.

Theorem: Suppose $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$ and

$\lim_{n\to\infty} f(n)/g(n) = L$, then if

$0 < L < \infty$ we have $O(f) = O(g)$

or if $L = 0$ we have $O(f) < O(g)$

or if $L = \infty$ we have $O(f) > O(g)$.

To say $\lim_{n\to\infty} f(n) = \infty$, just means ~~that if n gets big, then f(n) gets l~~ intuitively if n is "infinitely large" then $f(n)$ is "infinitely large" or that $f(n)$ grows without bound as n gets big. To say $\lim_{n\to\infty} f(n)/g(n) = 1$

means $f(n)/g(n)$ is close to L as n gets big. Let's do some examples to get the idea.

Examples

1. $O(n) = O(2n)$

   Since $\lim_{n\to\infty} n = \lim_{n\to\infty} 2n = \infty$ and $\lim_{n\to\infty} \frac{n}{2n} = \lim_{n\to\infty} \frac{1}{2} =$

2. $O(n^2) = O(7n^2 + 100n + 13)$

$$\frac{7n^2 + 100n + 13}{n^2} = 7 + \frac{100}{n} + \frac{13}{n^2} \to 7$$

   note that as $n \to \infty$, both $\frac{100}{n}$ and $\frac{13}{n^2}$ get small.

3. $O(n^8 + 5n^3) = O(\frac{1}{2}n^8 + 6)$

$$\frac{n^8 + 5n^3}{\frac{1}{2}n^8 + 6}\left(\frac{\frac{1}{n^8}}{\frac{1}{n^8}}\right) = \frac{1 + 5/n^5}{\frac{1}{2} + 6/n^8} \to \frac{1}{\frac{1}{2}} = 2$$

   "The trick" is divide both top & bottom by the highest power of n.

4. $O(n) < O(n \log n) < O(n^2)$

$$\frac{n}{n \log n} = \frac{1}{\log n} \to 0$$

$$\frac{n \log n}{n^2} = \frac{\log n}{n} \to 0$$

The fact that $\log n / n \to 0$ is usually proved in calculus classes. Intuitively speaking, we see that since $n = 10^{\log n}$, it takes $\log n$ digits to write n and so n is much bigger than $\log n$

In fact $O(\log n) < O(n^k)$, for any $k > 0$. Thus $O(n \log n) < O(n^{1+k})$, for any $k > 0$ and in particular when $k = 1$.

5. $O(2^n) < O(4^n)$

$$\lim_{n\to\infty} \frac{2^n}{4^n} = \lim_{n\to\infty} \left(\frac{2}{4}\right)^n = 0 \quad \text{since } \frac{2}{4} = \frac{1}{2} < 1.$$

6. If $A > 1$, then $O(n) < O(A^n)$.

Taking the limit of n/A^n is easy if one knows enough Calculus. However we can still do it with more work.

Lemma 1 If $B > 1$ and N is large enough so that $B > 1 + \frac{1}{N}$ and let $K = N/B^N$, then for $n \geq N$ $n \leq KB^n$

proof: By induction. When $n = N$, $K = N/B^N$ or $N = KB^N$. Assume ~~$n \geq B$~~ $n \leq KB^n$ is true. Multiply by B getting $Bn \leq KB^{n+1}$. Now $B > 1 + \frac{1}{N}$ ~~or $BN > N+1$~~. so $Bn > n + \frac{n}{N}$. But $\frac{n}{N} \geq 1$, so $Bn \geq n+1$ and $n+1 \leq KB^{n+1}$. $\square$

Now $$\frac{n}{A^n} = \frac{n}{B^n}\frac{B^n}{A^n} \quad \text{for } 1 < B < A.$$

so $$\frac{n}{A^n} \leq K\left(\frac{B}{A}\right)^n \quad \text{for } n \geq N$$

and thus $$\frac{n}{A^n} \longrightarrow 0$$

7. If $A > 1$, then $O(n^3) < O(A^n)$

Let $B = A^{1/3} > 1$ thus $\frac{n}{B^n} \longrightarrow 0$ by 6.

so that $$\left(\frac{n}{B^n}\right)^3 = \frac{n^3}{B^{3n}} = \frac{n^3}{A^n} \longrightarrow 0.$$

8. $O(10^n) < O(n!)$

Let $N = 20$ and note if $n > N$
$$\frac{10^n}{n!} \leq \frac{10^N}{N!}\left(\frac{1}{2}\right)^{n-N} \longrightarrow 0$$
(prove it by induction)

Problems

1. Show $O(n+1) = O(10n+7) = O(n+\log n) = O(n)$
2. Show $O(n^3+n^2+n+1) = O(n^3-13) = O(n^3)$
3. Show $O(n^{1/2}) < O(n)$
4. Show $O(\sqrt{n^2+1}) = O(n)$
5. Show $O(n^{100}) < O(n^{101} + \sqrt{n})$
6. Show $O(2^n) < O(3^n)$
7. Show $O(n2^n) > O(2^n)$
8. Show $O(n2^n) < O(3^n)$
9. Show $O(n^{100}) < O(2^n)$
10. Show $O(100^n) < O(n!)$
11. How are the big Oh's of the followings related
    $\sqrt{n}$, $n\log n$, $n\sqrt{n}$, 2^n, $\log n$, $n2^n$, $2^n\log n$, n^2, $n^{1/5}$
12. If there are two programs P1 & P2 which do the same thing and P1 runs in $O(f)$ and P2 runs in $O(g)$ and $O(f) < O(g)$, does this mean P1 will always be faster than P2? Why or why not?

13. Approximate 2^{100}, 2^{18}, 2^{16}, 2^{32} using $2^{10} \sim 10^3$

14. The following formula is in Advance Calculus Textbooks

$$1 \le \frac{n!}{\sqrt{2n\pi}\left(\frac{n}{e}\right)^n} \le 1 + \frac{1}{12n-1}$$

use it to show $O\left(\left(\frac{n}{e}\right)^n\right) < O(n!) < O(n^n)$

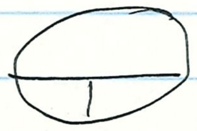
and that $O(n\log n) = O(\log n!)$

(Note $O(\log_b n) = O(\log_a n)$ if $a, b > 1$)

COP4531
DATA STRUCTS
PT 1.

1 Show $O\left(\frac{n(n+1)(2n+1)}{6}\right) = O(n^3)$
2. Arrange $O(n^3)$, $O(n^2 \log n)$, $O(n!)$, $O(2^n)$ in increasing order
3. For an input of size n, Prog A takes time $3n^2+2$ while Prog B takes time $100n$. For which values of $n \geq 1$ is A fastest (B fastest)
4. Find the Big Oh of the number of times FIRST, END, & NEXT are called in Prob. 2.11.
5. Count recursive programs [find big Oh of run time of Prob 1.12d]

6. The text's ADT list has an operation called END(L). A. What does this operation return?
B. Explain why END(L) couldn't be replaced by LAST(L). ~~C. Suppose we define L to be the empty list and explain why both~~
7. Write an array implemention of the List ADT operation DELETE(p, L) (p: position)
8 Write a singly link list implementation For LIST ADT operations FIRST & INSERT. Be sure to explain what "position" means

9. node = 

```
Procedure ABC (i, m: integer VAR P nodeptr)
if i ≤ m then  new(p)
               info(p) ← i
               ABC(2i, m, left(p))
               ABC(2i+1, m, right(p))
else P ← NIL.
```

A. Create a picture of what is create by the call ABC(1, 10, Q)
B. What is the run time ~~in terms of O(m)~~ in Big Oh in terms of m.
10. Using only LIST ADT operations, write a pascal procedure which ~~wh~~ outputs a list RESULT which is the reverse of the input list INLIST.
11. If n is a descendant of m, then n occurs (before? after? either?) m in (preorder?, inorder?, postorder?)
12. Pascal Sets!