CSE 3318 Notes 2: Growth of Functions

(Last updated 8/1/22 2:58 PM)

CLRS Chapter 3

Why constants are annoying . . .

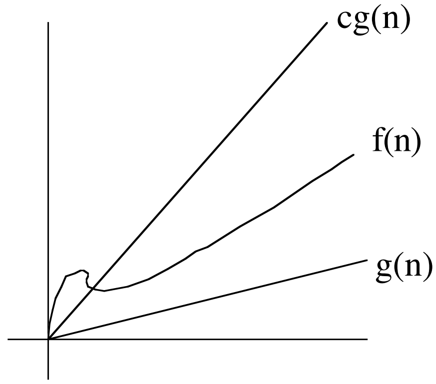
Problem Instance of Size n (n = size of data structure or input)
⇓
Algorithm for Problem
⇓
Language
⇓
Compiler
⇓
Machine
clock
memory levels
word length ($\Omega(\log n)$)
instruction set
⇓
Resources Consumed? (time, memory, processors, bandwidth)
(Asides: `https://dl-acm-org.ezproxy.uta.edu/doi/10.1145/2976758` ,
Compiler Explorer: `https://godbolt.org`)

Need to compare time and space usage of various algorithms for a problem.

2.A. Asymptotic Notation

$O(g(n))$ is a *set* of functions:

$$f(n) \in O(g(n)) \text{ iff } \exists\, c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ when } n \ge n_0$$



Theorem: If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ is a constant, then $f(n) \in O(g(n))$.

$f(n) = n^2, g(n) = n^3 \quad \lim_{n\to\infty} \frac{n^2}{n^3} = \lim_{n\to\infty} \frac{2n}{3n^2} = \lim_{n\to\infty} \frac{2}{6n} = 0 \Rightarrow n^2 \in O(n^3)$

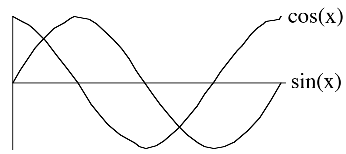
$f(n) = n^3, g(n) = n^2 \quad \lim_{n\to\infty} \frac{n^3}{n^2} = \lim_{n\to\infty} \frac{3n^2}{2n} = \lim_{n\to\infty} \frac{6n}{2} = \textit{unbounded} \Rightarrow n^3 \notin O(n^2)$

$f(n) = 3n^2 + 2n - 3, g(n) = 5n^2 - n + 2$

$$\lim_{n\to\infty} \frac{3n^2 + 2n - 3}{5n^2 - n + 2} = \lim_{n\to\infty} \frac{6n + 2}{10n - 1} = \lim_{n\to\infty} \frac{6}{10} = \frac{3}{5} \Rightarrow 3n^2 + 2n - 3 \in O(5n^2 - n + 2)$$

Conclusion: Toss out low-order terms $n^k \in O\left(n^l\right)$ if $l \geq k$.

Is $\sin(x) \in O(\cos(x))$?



Is $\ln(x) \in O(\log_{10}(x))$? [But in some situations, like Notes 04, log details are important]

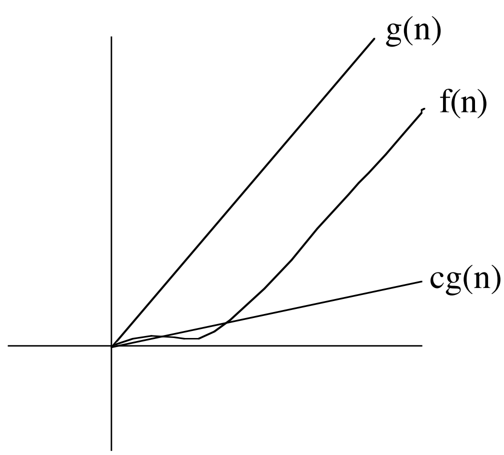
Is $2^n \in O\left(n^k\right)$ for some *fixed k*?

$n^k \in O\left(2^n\right)$ for any k. General case, $n^k \in O\left(c^n\right)$ for any $c > 1$

Aside: There are problems in the gap between polynomial and exponential:
`https://en.wikipedia.org/wiki/Graph_isomorphism_problem`

$\Omega(g(n))$ is a *set* of functions:

$f(n) \in \Omega(g(n))$ iff $\exists$ c and n_0 such that $0 \leq cg(n) \leq f(n)$ $(c > 0)$ when $n \geq n_0$



Theorem: If $\lim_{n\to\infty} \frac{g(n)}{f(n)}$ is a constant, then $f(n) \in \Omega(g(n))$.

$g(n) = n^2,\ f(n) = 3n^3 \quad \lim_{n\to\infty} \frac{n^2}{3n^3} = \lim_{n\to\infty} \frac{2n}{9n^2} = \lim_{n\to\infty} \frac{2}{18n} = 0 \Rightarrow n^3 \in \Omega(n^2)$

Upper bounds (O()) are for particular *algorithms*.

Lower bounds (Ω()) are for indicating the inherent difficulty of *problems*.

The hunt for an optimal algorithm . . .

Aside: `https://rjlipton.wpcomstaging.com/2012/02/01/a-brief-history-of-matrix-product/`

Best upper bound: $O\left(n^{2.373}\right)$ Best lower bound: $\Omega\left(n^2\right)$

Θ-notation (asymptotically *tight* bound) – Used to express time for worst-case *instances* for algorithm

$f(n) \in \Theta(g(n))$ iff $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.

Theorem: If $\lim_{n\to\infty} \frac{g(n)}{f(n)}$ is a constant > 0, then $f(n) \in \Theta(g(n))$.

$f(n) = 3n^2 + 2n - 3,\ g(n) = 5n^2 + n - 1$

$$\lim_{n\to\infty} \frac{3n^2 + 2n - 3}{5n^2 + n - 1} = \lim_{n\to\infty} \frac{6n + 2}{10n + 1} = \lim_{n\to\infty} \frac{6}{10} = \frac{3}{5} \Rightarrow 3n^2 + 2n - 3 \in \Theta(5n^2 + n - 1)$$

$\Theta(f(n))$ is an equivalence relation . . .

Symmetric: $g(n) \in \Theta(f(n)) \Leftrightarrow f(n) \in \Theta(g(n))$ [Doesn't work for O or Ω]
Reflexive: $f(n) \in \Theta(f(n))$ [Works for O and Ω]
Transitive: $f(n) \in \Theta(g(n))$ and $g(n) \in \Theta(h(n)) \Rightarrow f(n) \in \Theta(h(n))$ [Works for O and Ω]

Accepted abuses of asymptotic notation. (CLRS, p. 58)

1. $n^2 + n = \Theta\left(n^2\right)$ means $n^2 + n \in \Theta\left(n^2\right)$.

2. $3n^2 + 2n + 1 = 3n^2 + \Theta(n)$ means . . .

3. $2 + 3\Theta(n) = n + \Theta(n)$ means . . .

Aside: Indicating that a bound could/might be improved:

$o(f) = O(f) - \Theta(f)$ $\omega(f) = \Omega(f) - \Theta(f)$

$n^2 \in o(n^3)$ $\frac{1}{n^2} \in \omega\left(\frac{1}{n^3}\right)$

Example: p. 46 of Demaine & O'Rourke, *Geometric Folding Algorithms: Linkages, Origami, & Polyhedra*, Cambridge Univ. Press, 2007. (`https://www.amazon.com/dp/0521857570/`)

"Open Problem 4.1: Faster Generic Rigidity in 2D. Is there a $o\left(n^2\right)$-time algorithm to test generic rigidity of a given graph with *n* vertices?" (`https://minerva.cs.mtholyoke.edu/scholarship/rigidity-theory/getting-started-with-combinatorial-rigidity-theory`)

2.B. APPLYING ASYMPTOTIC ANALYSIS TO CODE SEGMENTS

```
for (i=0; i<n; i++)
  for (j=0; j<n; j++)
  {
    c[i][j] = 0;
    for (k=0; k<n; k++)
      c[i][j] += a[i][k]*b[k][j];
  }
```

```
min=max=a[0];
for (i=1; i<n; i++)
  if (a[i]<min)
    min=a[i];
  else if (a[i]>max)
    max=a[i];
```

```
equalCount=unequalCount=0;
for (i=0; i<n; i++)
  if (a[i]==val)
    equalCount++;
  else
    unequalCount++;
```

```
equalCount=0;
for (i=0; i<n; i++)
  if (a[i]==val)
    equalCount++;
unequalCount=0;
for (i=0; i<n; i++)
  if (a[i]!=val)
    unequalCount++;
```

Linear time or exponential time?

```
for (i=0; i<k; i++)
  ;
```

Test Problem: Perform an asymptotic analysis for the following code segment to determine an appropriate Θ set for the time used.

```
sum=0;

for (i=0; i<n; i=i+2)

  for (j=1; j<n; j=j+j)

    sum=sum + a[i]/a[j];
```

LU Decomposition (Gaussian Elimination, CSE 3380) - aside, useful as practice

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MAX_SLOTS 10
typedef char boolean;
int num_slots,i,j;
double slot[MAX_SLOTS+1];
typedef double row[MAX_SLOTS+2];
row ab[MAX_SLOTS+2],ab1[MAX_SLOTS+2];

void LU()
//SOLVES SYSTEM OF LINEAR EQUATIONS.
//TAKEN FROM M.J. Quinn,P. 147
//CODE HAS BEEN MODIFIED TO PRELOAD AB
//TO SAVE SPACE BY NOT STORING A AND B
//SOLUTION IS PLACED IN SLOT[*]

{
double amax,temp,c,sum;
int n,i,j,k,k1,ell,ell1,itemp;
row y;
int piv[MAX_SLOTS+1];

n=num_slots;
```

```
for (i=1;i<=n;i++)
  piv[i]=i;

for (k=1;k<=n-1;k++)
{
  printf("iteration %d\n",k);
  amax=0.0;
  for (ell1=k;ell1<=n;ell1++)
    if (amax<fabs(ab[ell1][k]))
    {
      amax=fabs(ab[ell1][k]);
      ell=ell1;
    }
  if (amax==0.0)
  {
    printf("singular matrix\n");
    abort();
  }
  printf("pivoting row %d\n",ell);
  itemp=piv[ell];
  piv[ell]=piv[k];
  piv[k]=itemp;
```

```
for (k1=1;k1<=k;k1++)
  {
    temp=ab[ell][k1];
    ab[ell][k1]=ab[k][k1];
    ab[k][k1]=temp;
  }

  c=1.0/ab[k][k];
  for (i=k+1;i<=n;i++)
    ab[i][k] *= c;

  for (j=k+1;j<=n;j++)
  {
    temp=ab[ell][j];
    ab[ell][j]=ab[k][j];
    ab[k][j]=temp;

    for (i=k+1;i<=n;i++)
      ab[i][j] -= ab[i][k]*ab[k][j];
  }

printf("LU matrix\n");
  for (i=1;i<=n;i++)
  {
    for (j=1;j<=n+1;j++)
      printf("%f ",ab[i][j]);
    printf("\n");
  }
}
printf("pivots\n");
for (i=1;i<=n;i++)
  printf("%d %d\n",i,piv[i]);

printf("forward substitution\n");
y[1]=ab[piv[1]][n+1];
for (i=2;i<=n;i++)
{
  sum=0.0;
  for (j=1;j<i;j++)
    sum+=ab[i][j]*y[j];
  y[i]=ab[piv[i]][n+1]-sum;
}

printf("intermediate vector y:\n");
for (i=1;i<=n;i++)
  printf("%f\n",y[i]);

printf("back substitution\n");
slot[n]=y[n]/ab[n][n];
for (i=n-1;i>=1;i--)
{
  sum=0.0;
  for (j=i+1;j<=n;j++)
    sum+=ab[i][j]*slot[j];
  slot[i]=(y[i]-sum)/ab[i][i];
}
}

int main()
{
int i,j,k;
double sum,L,U;

printf("enter n: ");
scanf("%d",&num_slots);

printf("enter matrix and column\n");
for (i=1;i<=num_slots;i++)
  for (j=1;j<=num_slots+1;j++)
    scanf("%lf",&ab[i][j]);
for (i=1;i<=num_slots;i++)
  for (j=1;j<=num_slots+1;j++)
    ab1[i][j]=ab[i][j];

printf("input:\n");
for (i=1;i<=num_slots;i++)
{
  for (j=1;j<=num_slots+1;j++)
    printf("%f ",ab[i][j]);
  printf("\n");
}

LU();

printf("solution\n");
for (i=1;i<=num_slots;i++)
  printf("%f\n",slot[i]);

printf("result of multiplying L & U\n");
for (i=1;i<=num_slots;i++)
{
  for (j=1;j<=num_slots;j++)
  {
    sum=0.0;
    for (k=1;k<=num_slots;k++)
    {
      if (i==k)
        L=1.0;
      if (i<k)
        L=0.0;
      if (i>k)
        L=ab[i][k];
      if (k==j)
        U=ab[k][j];
      if (k<j)
        U=ab[k][j];
      if (k>j)
        U=0.0;
      sum+=L*U;
    }
    printf("%f ",sum);
  }
  printf("\n");
}
printf("sub. sol. into orig. system:\n");
for (i=1;i<=num_slots;i++)
{
  sum=0.0;
  for (j=1;j<=num_slots;j++)
    sum+=ab1[i][j]*slot[j];
  printf("%f\n",sum);
}
}
```

2.C. DEMONSTRATING FUNDAMENTAL RESULTS FOR ASYMPTOTIC NOTATION

Exercise 3.2-1: $f(n)$ and $g(n)$ are positive functions. Show $\max(f(n),g(n)) = \Theta(f(n) + g(n))$

O: Since $f(n) \le f(n) + g(n)$ and $g(n) \le f(n) + g(n)$

$\max(f(n),g(n)) \le f(n) + g(n)$ for all n, so $n_0 = 0$ and $c = 1$

Ω: Must have $c(f(n) + g(n)) \le \max(f(n),g(n))$, so choose $n_0 = 0$ and $c \le 1/2$ [Consider $f(n) = g(n)$]

c	+	f	g	max
	10	1	9	9
	10	2	8	8
	10	3	7	7
	10	5	5	5

Test Problem: Prove that if $g(n) \in \Omega(f(n))$ then $f(n) \in O(g(n))$.

$cf(n) \le g(n)$ when $n \ge n_0$

So, $f(n) \le \frac{1}{c} g(n)$ when $n \ge n_0$

Need $f(n) \le dg(n)$ when $n \ge n_1$

Choose $d = \frac{1}{c}$ and $n_1 = n_0$

2.D. MISCELLANEOUS

floor(x) = $\lfloor x \rfloor$ Largest integer $\le x$ $\lfloor 2.5 \rfloor = ?$ $\lfloor -2.5 \rfloor = ?$

ceiling(x) = $\lceil x \rceil$ Smallest integer $\ge x$ $\lceil 2.5 \rceil = ?$ $\lceil -2.5 \rceil = ?$

Worst-case number of probes for binary search: $\lfloor \lg n \rfloor + 1$

Consider worst-case searches on tables with 15 elements (4) and 16 elements (5).

Review: $2^n \,?\, O(3^n)$ $2^n \,?\, \Omega(3^n)$ $\lg(2^n) = n$ $\lg(3^n) = n \lg 3$ $\lg(2^n) = \Theta(\lg(3^n))$

Factorials:

$$n! = \prod_{i=1}^{n} i = 1 \bullet 2 \bullet 3 \bullet \ldots \bullet n$$

$$\lg(n!) = \sum_{i=1}^{n} \lg i \in \Theta(n \lg n)$$

[Using Notes 03.C and `https://www.wolframalpha.com`]

n	$n!$	$\lg(n!)$	$n \lg n$
1	1	0	0
2	2	1	2
3	6	2.58	4.75
4	24	4.58	8
5	120	6.91	11.61
6	720	9.49	15.51
7	5040	12.3	19.65
8	40320	15.3	24
9	362880	18.47	28.53
10	3628800	21.79	33.22
11	39916800	25.25	38.05
12	479001600	28.84	43.02
13	6227020800	32.54	48.11
14	87178291200	36.34	53.30
15	1307674368000	40.25	58.60
20	2.43E18	61.08	86.44
200		1245	1528
2000		19053	21932
20000		256909	285754
200000		3233399	3521928
2000000		38977759	41863137

O:

$$n! \le n^n$$
$$\lg(n!) \le n \lg n$$
$$\lg(n!) \in O(n \lg n)$$

Ω:

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

$$\ge 1 \bullet \ldots \bullet \frac{n}{2} \bullet (\sqrt{n})^{\frac{n}{2}} \qquad \frac{n}{2} + k \ge \sqrt{n} \text{ for } n > 3 \text{ and } k \ge 0$$

$$\ge (\sqrt{n})^{\frac{n}{2}} \qquad 1 \bullet \ldots \bullet \frac{n}{2} \ge 1$$

$$\lg(n!) \ge \lg(\sqrt{n})^{\frac{n}{2}} = \frac{n}{2}\frac{\lg n}{2} = \frac{1}{4} n \lg n$$

(`https://en.wikipedia.org/wiki/Stirling's_approximation`)

2.E. Asymptotic Notation for Two Parameters (aside, CLRS exercise 3.2-7)

$$O(g(n,m)) = \left\{ \begin{array}{l} f(n,m) : \exists \text{positive constants } c, n_0, \text{ and } m_0 \text{ s.t.} \\ \quad \text{for every } (n,m) \text{ pair with either } n \ge n_0 \text{ or } m \ge m_0 \\ \quad 0 \le f(n,m) \le c g(n,m) \end{array} \right\}$$