Tuesday, September 01, 2009
3:28 PM

in Array class, I'd like to do this:

main()
{
    int n=10;
    Array<int> myArr(n);

    for(int i=0; i<n; i++)
        myArr[i] = ~~random()~~ ~~rand()/RAND_MAX~~;

(float)rand()/(float)RAND_MAX * 100.0

int?

man 3C rand
man 3C srand

    for(int i=0; i<n; i++)
        std::cout << myArr[i] << std::endl;

}

both are operator[](int index), one is mutator one is accessor

---

```
const T& operator[](int i) const { return arr[i]; }
      T& operator[](int i)       { return arr[i]; }
```

mutator

accessor

---

useful for matrices

```
#ifndef MATRIX_H
#define MATRIX_H
#include <vector>
template <typename T>
```

```
class matrix {
    public:
    // operator
    const std::vector<T>& operator[](int r) const   {return arr[i];}
    std::vector<T>& operator[](int r)  {return arr[i];}
    private:
    std::vector<std::vector<T> >    arr;
}
```

So when you use:

```
matrix<float>      R(4,4);
```

$$\begin{pmatrix} R[0][0] = \cos\theta; & R[0][1] = \sin\theta; & R[0][2] = 0.0; & \cdots 0 \\ R[1][0] = -\sin\theta; & R[1][1] = \cos\theta; & R[1][2] = 0. & \cdots 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & & 1 \end{pmatrix}$$

(this is "Rot Z" matrix, rotate a 1x4 vector about z-axis)

so $R[2][0] = 0$

$$\underbrace{\texttt{R.operator[](2)}}_{}\texttt{. operator[](0)}$$

vector<float>. operator[](0)

Tuesday, September 01, 2009
3:44 PM

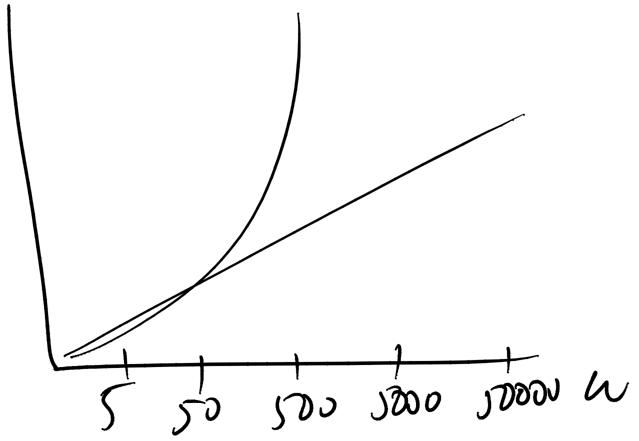
Alg Analysis (Chp. 2)

- Goal: Compare (predict) run time of algorithms
- empirical approach:
    - write a Timer class
    - use start(), stop() calls around code fragments
    - run code, vary input lengths, plot

file.dat

5	10
50	20
500	100
⋮	

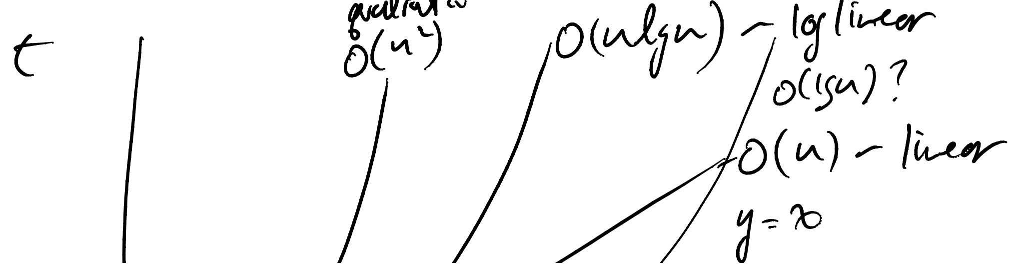
```
gnuplot
gnuplot> plot 'file.dat'
```

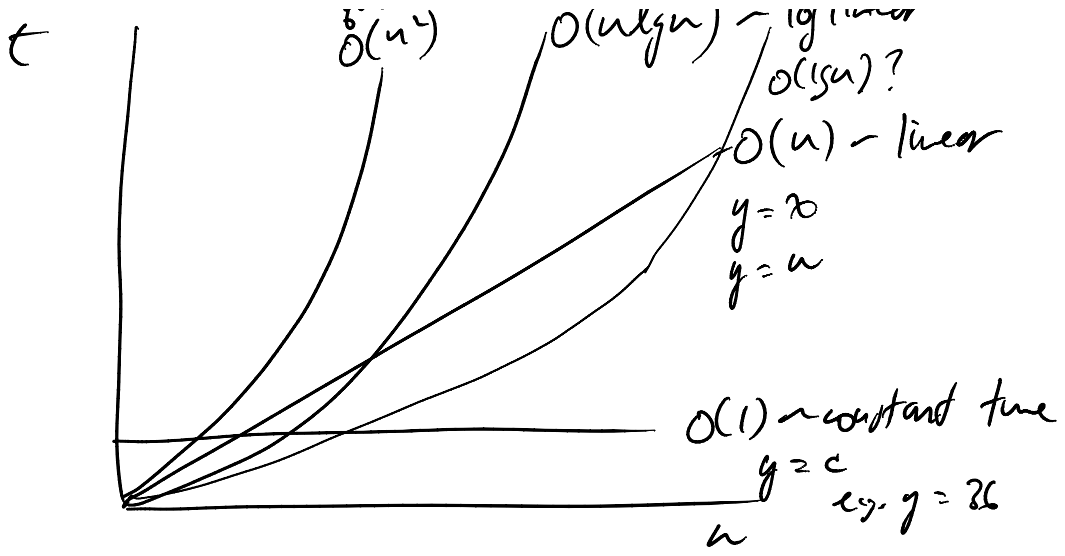


empirical approach

- & report: still done today,
  e.g., blah ray tracing code runs in 10s
  on 1,000,000 polygons on an
  Intel quad core 2.3 GHz
- problems:
    - machine-specific
    - unreliable — any other processes running on machine?

- analytical approach: big-oh notation





Comparisons:

$O(1) < O(\lg n) < O(\lg^2 n) < O(n) <$

$O(1)$	$O(\lg n)$	$O(\lg^2 n)$	$O(n)$
constant	logarithmic	iterated logarithmic	linear

$O(n \lg n) < O(n^2) < O(n^3) < O(2^n)$

$O(n \lg n)$	$O(n^2)$	$O(n^3)$	$O(2^n)$
lglinear	quadratic	cubic	exponential

what to count? $\left\{ \begin{array}{l} \text{declarations count } 0 \\ \text{statement/operations count } 1 \end{array} \right.$

Operation units	
	int sum(int n)
	{
0	int partialsum; // declaration
1	partial sum = 0; // assignment
2n+2	$\text{for}(\overset{1}{i=1};\ \overset{n+1}{i<=n};\ \overset{n}{i++})$
4n	$\text{partialsum} \underset{1}{\overset{}{+}}\underset{1}{=}\ i \underset{1}{*} i \underset{1}{*} i;$
1	return partialsum;
	}

6n+4

f(n) = n+1

- reduce all constants to 1 ⇒ 6n+4 → n+1

- by definition,

$$\boxed{T(n) = O(f(n)) \text{ if } \exists\ c, n_0 \text{ s.t. } T(n) \leq cf(n) \text{ when } n \geq n_0}$$

e.g. $f(n) = \circled{n}$

$f'(n) = n+1 \in O(n)$

$$n+1 \leq c\underset{\uparrow}{n}$$

$$T(n) \leq O(f(n))$$

$$cf(n)$$

when $n \geq n_0$

when $n \geq n_0$

pick
$n = 100$

101 $\quad$ $c(101)$ $\quad$ let $c = 2$, $\quad 101 \leq 202$ ✓

$$\boxed{n+1 \in O(n)}$$

- In general, drop lower-order terms,

$$T(n) = O(\cancel{7}n^2) = O(n^2+\cancel{4n}) = O(n^2)$$

- big-oh specifies <u>upper bound</u> – alg. can't be any worse than this $\Rightarrow$ $\boxed{O() \equiv \text{worst-case running time}}$

- $\Omega(g(n))$:

$$T(n) = \Omega(g(n)) \text{ if } \underset{\text{there exists}}{\exists} c, n_0 \underset{\text{such that}}{\Big|} T(n) \underset{\uparrow}{\textcircled{\geq}} cg(n)$$

means that g(n) must be alg's <u>lower bound</u>

alg is at least as fast/slow as this

$$\underset{\text{best case}}{cg(n)} \leq T(n) \leq \underset{\text{worst case}}{cf(n)}$$

- $T(n) = \Theta(h(n))$ iff $T(n) = O(h(n))$ & $T(n) = \Omega(h(n))$

$$\Omega(n) = T(n) = O(n)$$

upper & lower bound are the same

used to talk about <u>average run time</u>

little oh

- $T(n) = \overset{\text{little oh}}{o}(f(n))$ if $\forall c \; \exists n_0 \mid T(n) < cf(n)$ when $n > n_0$

strict inequality

(big-oh allows $\leq$)

(little-oh does not)

~~(no one uses little-oh)~~

Tuesday, September 01, 2009
4:17 PM

- program analysis (rules of thumb)

1. for loops:
   - no statements × no. iterations

2. nested loops
   - count inside out

for(i=0 ... n)

for(j=0, ... n)

O(n) O(n)

×

$O(n^2)$

3. consecutive statements: just add up

4. conditionals: use larger of 2 branches

5. recursion: the tricky one
   - need to solve recurrence relations

Tuesday, September 01, 2009
4:21 PM

e.g.,
```
long fib(int n)
{
    if (n<=1) return 1;
    else
        return fib(n-1) + fib(n-2)
}
```

$T(0) + T(1) = 1$

$T(n) = T(n-1) + T(n-2) + 2$

book says for $n > 4$, $T(n) \geq \left(\frac{3}{2}\right)^n$

inductive proof:

$T(n) = T(n-1) + T(n-2) \geq \boxed{\left(\frac{3}{2}\right)^n}$

assume this holds for $n > 4$

show $\boxed{T(n+1) \geq \left(\frac{3}{2}\right)^{n+1}}$

$T(n) = T(n-1) + T(n-2)$

$n+1 \quad n+1-1 \quad n+1-2$

$\quad\quad n \quad\quad n-1$

$T(n+1) = T(n) + T(n-1)$

$= \left(\frac{3}{2}\right)^n + \left(\frac{3}{2}\right)^{n-1}$

trying to get to here (n+1 exponent)

$\left(\frac{3}{2}\right)^{-1}\left(\frac{3}{2}\right)^{n+1} + \left(\frac{3}{2}\right)^{-2}\left(\frac{3}{2}\right)^{n-1+1+1}$

$= \left(\frac{2}{3}\right)\left(\frac{3}{2}\right)^{n+1} + \left(\frac{2}{3}\right)^2\left(\frac{3}{2}\right)^{n+1}$

$$= \left(\frac{2}{3}\right)\left(\frac{3}{2}\right)^{n+1} + \left(\frac{2}{3}\right)^2\left(\frac{3}{2}\right)^{n+1}$$

$$= \left(\frac{2}{3} + \frac{4}{9}\right)\left(\frac{3}{2}\right)^{n+1}$$

$$= \left(\frac{10}{9}\right)\left(\frac{3}{2}\right)^{n+1} \geq \left(\frac{3}{2}\right)^{n+1}$$

so we're done

- generally, the types of recursive algs we'll see involve splitting up a list then recursing on each half (divide & conquer)

e.g., insert(list)

```
{
    if (list.size() == 1) return;  ——— O(1)
    // split list in 2;  ——————— O(n)
    insert(list/2)     }  O(n/2)+O(n/2) = 2T(n/2)
    insert(list/2)     }
```

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

sub in n for O(n)

$T(1) = 1$

$T(n) = 2T\left(\frac{n}{2}\right) + n$ how to solve this?

Two methods: telescoping sum, rewrite back substitution