– LECTURE 23 –

- Determining precise formula for running time $f(n)$ is difficult
- Simplify by using a big O expression
  $O(g(n))$ is an upper bound on $f(n)$

PROPERTIES of big-O

① Simplicity

$f(n) = O(g(n))$

$g(n)$ is a big O bound on $f(n)$

$g(n)$ is simple if

① It is a single term AND
② Coefficient is 1

No

ex)
```
sum=0;
for (i=0, i<n, i++);
    for (j=0, j<n, j++);
        sum+ = arr[i][j];
```

is $f(n) = c'n^2 + c''n + c'''$, where

$c' = c_2 + c_3$, $c'' = 2c_2$ and $c''' = c_1 + c_2$

Let $g(n) = c'n^2 + c''n + c'''$,

$g_2(n) = c'n^2$

$g^3(n) = n^2$

Then $f(n) = O(g_i(n))$ for each $i = 1, 2, 3$

Note: g_1 & g_2 are not simple

g^3 is simple

So the running time is $O(g_3(n))$ which is $O(n^2)$

② Tightness

We want the "tightest" big-O bound we can prove. If $f(n) = O(g(n))$ then we cannot find a func. $h(n)$ that grows at least as fast as $f(n)$ but grows slower than $g(n)$

$g(n)$ is tight bound on $f(n)$ if

① $f(n) = O(g(n))$ AND

② If $f(n) = O(h(n))$ then it is also true that $g(n) = O(h(n))$

ex 2) Running time for alg. in ex(1)

$f(n) = c' n^2 + c'' n + c'''$

We had $g_3(n) = n^2$

Let $g_4(n) = n^3$

CLAIM #1 $g_3(n)$ is tight bounded on $f(n)$

CLAIM #2 $g_4(n)$ is NOT a tight bound

Proof of Claim #1

$f(n) = O(g_3(n))$

Suppose $f(n) = O(h(n))$

Then there are c and n_0 such that

$f(n) = c'n^2 + c''n + c''' \leq c\,h(n)$

for all $n \geq n_0$

Then $h(n) \geq \left(\frac{c'}{c}\right) n^2$

for all $n \geq n_0$

But $g_3(n) = n^2$ then $g_3(n) \leq \left(\frac{c}{c'}\right) h(o)$

for all $n \geq n_0$

Thus $g_3(n) = O(h(n))$

Proof of Claim #2

Pick $h(n) = n^2$

We have $f(n)$ is $O(h(n))$

but n^3 is not $O(h(n))$

FACT: show $f(n) = \Theta\, g(n)$ to prove $g(n)$ is a tight bound on $f(n)$

Ex. in previous example we showed

$f(n) = n^8 + 7n^7 - 10n^5 - 2n^4 + 3n^2 - 17$

is $\Theta(n^8)$

This means n^8 is a tight bound on $f(n)$

Analyzing Running Time of a Program

Big-O	Informal Name
$O(1)$	constant
$O(\log n)$	logarithmic
$O(n)$	linear
$O(n \log n)$	n log n
$O(n^2)$	Quadratic
$O(n^3)$	Cubic
$O(n^k)$	Polynomial
$O(2^k)$	Exponential

Algorithms with running times...

- $\Theta(n)$ have linear complexity
- $\Theta(n^2)$ have quadratic
- $\Theta(n^k)$ have polynomial complexity

Efficient usually means polynomial complexity

① Simple Statements $O(1)$

② If - Statement

③ For - Statement

④ while - Statement

⑤ Do-while statement