Programming Assignment #1

DUE ~~02/03~~ 02/04. (FRI)

Algorithm Efficiency.

N - input size.

Efficiency measured as $f(N)$.

Let N = 1000

Ex. ① `for (i = 0; i < N; i++)` → linear loop.

```
{
    ... /* do something */ [c = a + b;]
}
```

↓

if these statements are not loops.
or f^n calls. etc.
then $f(N) = N$

② `for (i = 0; i < N; i += 2)` # iterations = $\frac{N}{2}$

$$f(N) = \frac{N}{2}$$

algorithms where $f(N) = N$ or $\frac{N}{2}$ etc.
are linear in the size of input.

[linear f^n: $f(x) = mx + c$
$f(N) = mN + c$]

③ `for (i = 1; i <= N; i *= 2)` Multiply loop.

i = 1	16	256
i = 2	32	512 ← last
i = 4	64	1024
8	128	

iterations = 10.

$f(N) = ?$ $\log(N)$

Divide loop: `for (i = N; i >= 1; i /= 2)` → 3.

iterations = 10.

$f(N) = \log N$.

Let x = # iterations.. (total)

$2^x \approx N = 1000$.

$x \approx \log_2(N)$.

(logarithmic loop)

④ Nested loops. <=

```
for (i = 0; i < N; i++)
    for (j = 1; j <= N; j *= 2)
    {
        ....
    }
```

$f(N) = N$ ← $f(N) = \log N$

$f(N)$ (for nested loop) $= N \log N$.

$f(N)=N \rightarrow$ `for (i = 0; i < N; i++)` /* Quadratic loop */

$f(N)=N \rightarrow$ `for (j = 0; j < N; j++)`

`{ .. }`.

$f(N) = N^2$

(dependent nested loop)

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

→ `for (j = 0; j < i; j++)`

for the block. → `{ .. }`.

[number of times block is exec:

iterations =

→ $i = 1$; # iterations of inner for loop = 1 [$j = 0$]

→ $i = 2$; " " " = 2

⋮

$i = N$; " " = N

→ # iterations $= 1 + 2 + \cdots + N$ ←

$= (1+N) + (2+N-1) + (3+N-2) + \cdots$

$= \frac{N(N+1)}{2} = \frac{N^2}{2} + \frac{N}{2} < N^2$

$\boxed{f(N) = \frac{N^2}{2} + \frac{N}{2}}^2$

→ $f(N)$ is of the order $O(N^2)$

[Quadratic loop $f(N) = N^2$
nested dependent loop $f(N) = \frac{N^2}{2} + \frac{N}{2}$]

↑ term which grows faster with input size (N).

Big - O Notation.

let $f(N) = \frac{1}{2}N^3 + \frac{N^2}{2} + \frac{N-1}{4} + \log N$

$f(N)$ is of order $O(N^3)$

↙ dominant factor in execution time.
[drop constants/coefficients, remove non-dominant factors]

Ex. Linear loops $f(N) = c_1 N + c_2$

Algorithm efficiency is $O(N)$.

computer. - executes 10^6 instructions per sec.

```
loop f(N) = ⋮
{ ...
}
```

Add 2 matrices: if # rows = # cols = N

algo. eff. is $O(N^2)$

Multiply 2 matrices: " " " $O(N^3)$

Summary:

pseudocode (structure)

Data structure.

Abstract Data Type. - encapsulation.

↳ implementation - array

- linked list.

↓

generic algorithms.

(can handle multiple data types)

↓

C language < void ptr

fn ptr.

Algo. efficiency - Big O notation.

Ch-2 Repetitive algorithms - two design approaches.

1) Iterative (loops)

2) Recursive - the algorithm calls itself

- does not require a loop.

Use recursion -

when algorithm appears within the definition itself.

Ex. $n!$

$$fact(n) = n\,(n-1)(n-2)\cdots 2\cdot 1 \quad \forall n \geq 1$$

$$fact(0) = 1$$

definition → $fact(n) = n \cdot fact(n-1)$ $\forall n \geq 1$