3 Analysis of Non-Recursive functions

Thursday, August 31, 2023 10:40 AM

E.G.

```
FUNCTION MaxElement ( A[0...n-1] )
    maxval ← A[0]
    for i ← 1 to n-1 do
        if A[i] ≥ maxval
            maxval ← A[i]
    return maxval
```

2 + 2 + 4 + $\sum_{i=1}^{n-1} 4$ = $2 + \sum_{i=1}^{n-1} 4$

$R_{MaxE}(n) = 2 + (n-1)4 = 2 + 4n - 4 = 4n - 2$

$= 4n - 2 \in \Theta(n)$

$2 + (4 + 4 + 4 \cdots + 4)$

i: 1 2 3 ... n-1 (n-1 terms)

Computing whole runtime functions: too much work

Instead let's concentrate on a basic operation.

∴ operation that characterizes algorithm.

$C_{MaxE}(n) = \sum_{i=1}^{n-1} 1 = n-1 \in \Theta(n)$

E.G

```
FUNCTION foo ( A[0...n-1] )
    for i ← 1 to n+n
        do something(1)
```

E.G

```
FUNCTION allUnique ( A[0...n-1] )
    for i ← 0 to n-2 do
        for j ← i+1 to n-1 do
            if A[i] = A[j] return false
    return true.
```

$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$

$= \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1]$

$= \sum_{i=0}^{n-2} (n - 1 - i)$

$= \sum_{i=0}^{n-2} (n-1) - \sum_{i=0}^{n-2} i$

$= (n-1) \cdot \sum_{i=0}^{n-2} 1 - \sum_{i=0}^{n-2} i$

$= (n-1)^2 - \frac{(n-2)(n-1)}{2}$ (0 + 1 + 2 + 3 + ... + n-2 arithmetic series.)

$= \frac{(n-1)n}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$

$\approx \frac{1}{2}n^2 \in \Theta(n^2)$

tip

$\sum_{i=L}^{U} 1 = U - L + 1$

tip

$\sum_{i=L}^{U} (a \pm b) = \sum_{i=L}^{U} a \pm \sum_{i=L}^{U} b$

tip

$\sum_{i=L}^{U} a \cdot i = a \cdot \sum_{i=L}^{U} i$

tip

$\sum_{i=0}^{U} i = \frac{U(U+1)}{2}$

E.G.

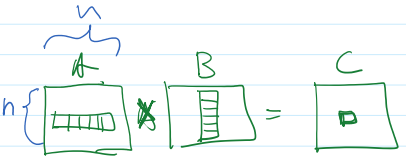
```
FUNCTION SqMatrixMult ( A B C )
```

n

A B C

E.G.

```
FUNCTION SqMatrxMult(A, B, C)
    for i ← 0 to n-1
        for j ← 0 to n-1
            C[i,j] ← 0.0
            for k ← 0 to n-1
                C[i,j] ← C[i,j] + A[i,k] * B[k,j]
    return C
```



$C[i,j] = A[i,0] \cdot B[0,j] + A[i,1] \cdot B[1,j] + \dots + A[i,n] \cdot B[n,j]$

- Used square matrices.
- basic operation

$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n$$

$$= \sum_{i=0}^{n-1} n^2$$

$$= n^3 \in \Theta(n^3)$$

EG. what about while ?

```
FUNCTION Binlen(n)
    // the number of binary digits in n's binary representation.
    count ← 1
    while n > 1 do
        count ← count + 1
        n ← ⌊n/2⌋
    return count.
```

DEV //

n = 255 → n_{bin} = 11111111 (8)

What happens to the variables involved in the end condition?
n is divided by 2.

How many times loop repeats? = how many times can you divide n?

$$D(n) = \log_2 n \in \Theta(\log n)$$

Why we don't care about base?
for bases a and b

$$\log_a x = \frac{\log_b x}{\log_b a} \quad ; \quad \log_a x = \log_a b \cdot \log_b x$$

$$\log_a x = C \cdot \log_b x$$

- GENERAL PLAN FOR ANALYZING THE TIME EFFICIENCY OF NON-RECURSIVE ALGORITMS

1.- Decide on parameter(s) indicating input size.

2.- Identify the algorithms basic_operation.
A: find it inside the innermost loop.

3.- Check whether the number of times the basic_operation is executed depends only on the size of the input or it depends on other factors,
- if so: worst-case, avg-case must be analysed separately

4.- Set up a sum expressing the number of execution of the basic_operation.

5.- Simplify sum to closed form,
by - rules of sum manipulation
- standard sum formulas,
or at least, find the order-of-growth of the count.

- EOF -